Remember Where You Came From: On The Second-Order Random Walk Based Proximity Measures

Yubao Wu, Yuchen Bian, Xiang Zhang
Department of Electrical Engineering and Computer Science, Case Western Reserve University
{yubao.wu, yuchen.bian, xiang.zhang}@case.edu

ABSTRACT

Measuring the proximity between different nodes is a fundamental problem in graph analysis. Random walk based proximity measures have been shown to be effective and widely used. Most existing random walk measures are based on the first-order Markov model, i.e., they assume that the next step of the random surfer only depends on the current node. However, this assumption neither holds in many real-life applications nor captures the clustering structure in the graph. To address the limitation of the existing first-order measures, in this paper, we study the second-order random walk measures, which take the previously visited node into consideration. While the existing first-order measures are built on node-to-node transition probabilities, in the second-order random walk, we need to consider the edge-to-edge transition probabilities. Using incidence matrices, we develop simple and elegant matrix representations for the second-order proximity measures. A desirable property of the developed measures is that they degenerate to their original first-order forms when the effect of the previous step is zero. We further develop Monte Carlo methods to efficiently compute the second-order measures and provide theoretical performance guarantees. Experimental results show that in a variety of applications, the second-order measures can dramatically improve the performance compared to their first-order counterparts.

1. INTRODUCTION

A fundamental problem in graph analysis is to measure the *proximity* (or closeness) between different nodes. It serves as the basis of many advanced tasks such as ranking and querying [22, 25, 11, 27], community detection [2, 26], link prediction [21, 19], and graph-based semi-supervised learning [29, 28].

Designing effective proximity measures is a challenging task. The simplest notation of proximity is based on the shortest path or the network flow between two nodes [6]. Random walk based measures have recently been shown to be effective and widely used in various applications. The basic idea is to allow a surfer to randomly explore the graph. The probabilities of the nodes being visited by the random surfer are used to measure the importance of the nodes or the similarity between different nodes. The most commonly used random walk based proximity measures include PageRank [22], random walk with restart [25], and SimRank [11].

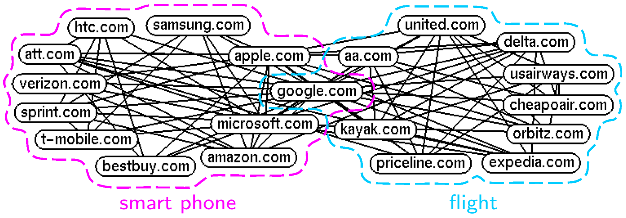


Figure 1: An example of the web domain graph

Most existing random walk measures are based on the first-order Markov model [15], i.e., they assume the next node to be visited only depends on the current node and is independent of the previous step. However, this assumption does not hold in many real-life applications. For example, consider the clickstream data which records the sequences of web domains visited by individual users [3]. The existing first-order random walk measures assume that the next page a user will visit only depends on the current page and is independent on the previous page the user has visited. This is clearly not true.

Figure 1 shows a subgraph of the real-life web domain graph[1] [17]. Each node in the graph represents a domain, and two domains share an edge if there are hyperlinks between them. The domains in the graph form two communities. The domains in the left community are about smart phones, and those in the right community are about flights. Suppose the random surfer is currently on google.com and the previously visited node is apple.com, i.e., the surfer came from the smart phone community. The existing first-order random walk measures do not consider where the surfer came from and the transition probability only depends on the edges incident to the current node. Based on this assumption and the graph topology, in the next step, the probabilities to visit att.com and delta.com are 2.4×10^{-5} and 3.1×10^{-5} respectively. That is, the surfer has a higher probability to visit a domain about flight even though she just visited a smart phone domain. However, using the real-life clickstream data (collected from comScore Inc.), given that the previous node is apple.com, the probabilities to visit att.com and delta.com are 8.5×10^{-4} and 3.7×10^{-6} respectively. That is, the probability to visit a smart phone domain is more than 200 times higher than the probability to visit a flight domain.

Proceedings of the VLDB Endowment, Vol. 10, No. 1
Copyright 2017 VLDB Endowment 2150-8097/16/07.

[1] The entire graph is publicly available at http://webdatacommons.org

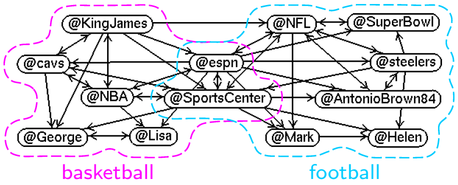


Figure 2: An example of the Twitter follower network

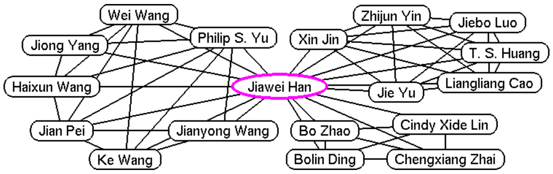


Figure 3: An example of the research collaboration network

As another example, consider the Twitter follower network. Figure 2 shows a subgraph of the real Twitter follower network. The users form two communities, the basketball community on the left and the football community on the right. If the NBA player LeBron James (@KingJames) posts a tweet, it is likely to be propagated among the users in the basketball community instead of the football community. Similarly, if the Pittsburgh Steelers (@steelers) post a tweet, it is likely to be propagated in the football community. The real tweet cascade data supports this intuition: given that the tweet is from @KingJames, the transition probabilities from @espn to @NBA and from @espn to @NFL are 5.6×10^{-2} and 2.1×10^{-4} respectively. However, using the first-order random walk, the transition probabilities are both 4.5×10^{-3}. That is, the probabilities to go to both communities are similar.

In the examples above, the visiting sequences are recorded in the network flow data. When such data is not available, it is still important to know where the surfer came from. Consider the local community detection problem, whose goal is to find a community nearby a given query node [2, 26]. Using the DBLP data[2], Figure 3 shows three different research communities involving Prof. Jiawei Han at UIUC. The authors in the left community are senior researchers in the core data mining research areas. The authors in the upper right community have published many works on social media mining. The authors in the lower right community mostly collaborate on information retrieval. Suppose that the random surfer came from the left community, e.g., from Prof. Wei Wang, and is currently at node Prof. Jiawei Han. Intuitively, in the next step, the surfer should walk to a node in the left community, since the authors in this community are more similar to Prof. Wei Wang. However, using the first-order random walk model, the probabilities of the surfer walking into the three communities are similar.

To address the limitation of the existing first-order random walk based proximity measures, in this paper, we investigate the second-order random walk measures, which take the previously visited node into consideration. We systematically study the theoretical foundations of the second-order measures. Specifically, the existing first-order measures are all built on the node-to-node transition probabilities, which can be defined using the adjacency matrix of the graph. To take the previous step into consideration, in the second-order random walk, we need to consider edge-to-edge transition probabilities. We show that such probabilities can be conveniently represented by incidence matrices of the graph [15]. Based on these mathematical tools, we develop simple and elegant matrix representations for the second-order measures including PageRank [22], random walk with restart [25], SimRank [11], and SimRank* [27], which are among the most widely used proximity measures. A desirable property of the developed second-order measures is that they can degenerate to their original first-order forms when the effect of the previous step is zero. Furthermore, to efficiently compute the second-order measures, we design Monte Carlo algorithms, which effectively simulate paths of the random surfer and estimate proximity values. We formally prove that the estimated proximity value is sharply concentrated around the exact value and converges to the exact value when the sample size is large. We perform extensive experiments to evaluate the effectiveness of the developed second-order measures and the efficiency of the Monte Carlo algorithms using both real and synthetic networks.

2. RELATED WORK

In the first-order random walk, a random surfer explores the graph according to the node-to-node transition probabilities determined by the graph topology. If the random walk on the graph is irreducible and aperiodic, there is a stationary probability for visiting each node [15]. Various random walk based proximity measures have been developed, among which PageRank [22], random walk with restart [25], SimRank [11], and SimRank* [27] have gained significant popularity and been extensively studied. In PageRank, in addition to following the transition probability, at each time point, the surfer also has a constant probability to jump to any node in the graph. Random walk with restart is the query biased version of PageRank: at each time point, the surfer has a constant probability to jump to the query node. SimRank is based on the intuition that two nodes are similar if their neighbors are similar. The SimRank value between two nodes measures the expected number of steps required before two surfers, one starting from each node, meet at the same node if they walk in lock-step. SimRank* is a variant of SimRank, which allows the two surfers not to walk in lock-step.

Very limited work has been done on the second-order random walk measure. In [24], the authors study memory-based PageRank, which considers the previously visited node. However, the developed measure does not degenerate to the original PageRank when the effect from the previous node is zero. Along the same line, multilinear PageRank [9] also tries to generalize PageRank to the second-order. It approximates the probability of visiting an edge by the product of the probabilities of visiting its two end nodes. This may not be reasonable, e.g., the probability of visiting a nonexistent edge would be non-zero. Both methods are specifically designed for PageRank and do not apply to other measures.

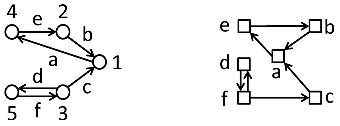
3. THE SECOND-ORDER RANDOM WALK

In this section, we study the foundation of the second-order random walk. The first-order random walk is based on node-to-node transition probabilities. In the second-order random walk, we need to consider edge-to-edge transition

[2] The data is publicly available at `http://dblp.uni-trier.de/xml/`

Table 1: 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, $[\mathbf{S}]_{u,v}=s_{u,v}$



(a) a toy graph (b) the line graph

Figure 4: An example graph and its line graph

probabilities. The main symbols used in this paper and their definitions are listed in Table 1.

3.1 The Edge-to-Edge Transition Probability

Consider the first-order random walk, where a surfer walks from node i to j with probability $p_{i,j}$. Let $\mathbb{X}_t$ be a random variable representing the node visited by the surfer at time point t. The node-to-node transition probability $p_{i,j}$ can be represented as a conditional probability $\mathbb{P}[\mathbb{X}_t=j|\mathbb{X}_{t-1}=i]$. Let $r_j^t=\mathbb{P}[\mathbb{X}_t=j]$ represent 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 set of in-neighbors of j.

Now consider the second-order random walk. We need to consider where the surfer came from, i.e., the node visited before the current node. We use $p_{i,j,k}$ to represent the transition probability from node j to k given that the previous step was from node i to j, i.e., $p_{i,j,k}=\mathbb{P}[\mathbb{X}_{t+1}=k|\mathbb{X}_{t-1}=i, \mathbb{X}_t=j]=\mathbb{P}[\mathbb{X}_t=j,\mathbb{X}_{t+1}=k|\mathbb{X}_{t-1}=i,\mathbb{X}_t=j]$.

Let $\mathbb{Y}_t=(i,j)$ represent the joint event $(\mathbb{X}_{t-1}=i,\mathbb{X}_t=j)$, i.e., the surfer is at node i at time $(t-1)$ and at node j at time t. Then, the second-order transition probability can be written as $p_{i,j,k}=\mathbb{P}[\mathbb{Y}_{t+1}=(j,k)|\mathbb{Y}_t=(i,j)]$, which can be interpreted as the transition probability between edges: let $u=(i,j)$ be the edge from node i to j, and $v=(j,k)$ be the edge from node j to k, we can rewrite $p_{i,j,k}$ as $p_{u,v}$.

Probability $p_{i,j,k}$ can be treated as the node-to-node transition probability in the line graph of the original graph. For example, Figures 4(a) and 4(b) show an example graph and its line graph. The second-order transition probability $p_{4,2,1}$ in Figure 4(a) is the same as the first-order transition probability $p_{\mathrm{e},\mathrm{b}}$ in Figure 4(b).

Let $s_{(i,j)}^t=\mathbb{P}[\mathbb{Y}_t=(i,j)]$ denote the probability of visiting edge (i,j) between time $(t-1)$ and t. We have that

$$s_{(j,k)}^{t+1}=\sum_{i\in I_j} p_{i,j,k}\cdot s_{(i,j)}^t$$

In the following, we introduce incidence matrices, which will be used as building blocks in the second-order random walk measures.

3.2 Incidence Matrices as the Basic Tool

A graph can be represented by its adjacency matrix $\mathbf{A}$, whose element $[\mathbf{A}]_{i,j}$ represents the weight of edge (i,j). Let $\mathbf{D}$ denote the diagonal matrix with $[\mathbf{D}]_{i,i}$ being the out-degree of node i. In the first-order random walk, the node-to-node transition matrix can be represented as $\mathbf{P}=\mathbf{D}^{-1}\mathbf{A}$. In the second-order random walk, instead of using the adjacency matrix, we will use incidence matrices [15].

The incidence matrices $\mathbf{B}$ and $\mathbf{E}$ represent the out-edges and in-edges of the nodes respectively. In matrix $\mathbf{B}$, each row represents a node and each column represents an edge. In matrix $\mathbf{E}$, each row represents an edge and each column represents a node. The elements in matrices $\mathbf{B}$ and $\mathbf{E}$ are defined as follows.

$$[\mathbf{B}]_{i,u}=\begin{cases}1, & \text{if edge } u \text{ is an out-edge of node } i\,,\\ 0, & \text{otherwise}.\end{cases}$$

$$[\mathbf{E}]_{u,i}=\begin{cases}1, & \text{if edge } u \text{ is an in-edge of node } i\,,\\ 0, & \text{otherwise}.\end{cases}$$

For example, the incidence matrices of the graph in Figure 4(a) are

$$\mathbf{B}=\begin{array}{c} \\ 1\\2\\3\\4\\5\end{array}\!\!\begin{array}{c}\begin{array}{cccccc}\mathsf{a}&\mathsf{b}&\mathsf{c}&\mathsf{d}&\mathsf{e}&\mathsf{f}\end{array}\\ \begin{bmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\end{bmatrix}\end{array}, \text{ and } \mathbf{E}^{\mathsf{T}}=\begin{array}{c} \\ 1\\2\\3\\4\\5\end{array}\!\!\begin{array}{c}\begin{array}{cccccc}\mathsf{a}&\mathsf{b}&\mathsf{c}&\mathsf{d}&\mathsf{e}&\mathsf{f}\end{array}\\ \begin{bmatrix}0&1&1&0&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\\1&0&0&0&0&0\\0&0&0&1&0&0\end{bmatrix}\end{array}.$$

Note that in the above definitions, the orders of nodes and edges are consistent in $\mathbf{B}$ and $\mathbf{E}$.

The incidence matrices can be conveniently used to reconstruct other commonly used matrices in graph analytics. For example, let $\mathbf{W}$ be a diagonal matrix with $[\mathbf{W}]_{u,u}$ being the weight of edge u, then the adjacency matrix can be represented by the incidence matrices as $\mathbf{A}=\mathbf{BWE}$. The out-degree matrix $\mathbf{D}$ can be represented as $\mathbf{D}=\mathbf{BWB}^{\mathsf{T}}$.

Let $\mathbf{H}$ denote the node-to-edge transition probability matrix, with $[\mathbf{H}]_{i,u}$ representing the probability that the surfer will go through an out-edge u of node i, i.e.,

$$[\mathbf{H}]_{i,u}=\begin{cases}w_u/w_i\,, & \text{if edge } u \text{ is an out-edge of node } i\,,\\ 0, & \text{otherwise},\end{cases}$$

where w_u is the weight of edge u, and w_i is the out-degree of node i. $\mathbf{H}$ can be represented using incidence matrices as $\mathbf{H}=\mathbf{D}^{-1}\mathbf{BW}$. The node-to-node transition probability matrix can then be represented as $\mathbf{P}=\mathbf{HE}$.

3.3 Obtaining Edge Transition Probabilities

In the first-order random walk, the element $p_{i,j}$ in the node-to-node transition matrix $\mathbf{P}$ is calculated as $p_{i,j}=\frac{w_{i,j}}{w_i}$, where $w_{i,j}$ and w_i are the weight of edge (i,j) and out-degree of node i respectively.

In the second-order random walk, we use $\mathbf{M}$ to represent the edge-to-edge transition matrix, with element $p_{u,v}=p_{i,j,k}$, where $u=(i,j)$ and $v=(j,k)$. Next, we discuss two different ways to obtain the edge-to-edge transition probability.

Utilizing Network Flow Data: In many applications, the information on the node visiting sequences is available. For example, as discussed in Section 1, we may know the sequences of web domains browsed by different users, or we may have the tweet cascade information. In this case, we can break each sequence into trigrams, i.e., segments consisting of two consecutive edges [24]. For example, sequence $i \to j \to k \to l$ can be broken into two trigrams, $i \to j \to k$ and $j \to k \to l$.

To obtain the second-order transition probability, recall that $p_{i,j,k}$ is the conditional probability of visiting edge (j,k) given that the previously visited edge is (i,j). Let $\gamma_{i,j,k}$ be the number of trigrams $i \to j \to k$. $p_{i,j,k}$ can be calculated as

$$p_{i,j,k} = \frac{\gamma_{i,j,k}}{\sum_{l \in O_j} \gamma_{i,j,l}} ,$$

where O_j is the set of out-neighbor nodes of j. That is, $p_{i,j,k}$ is the proportion of $i \to j \to k$ trigrams in all trigrams with (i,j) being the first edge.

When the network flow data is not available, we can use the following approach to obtain $p_{i,j,k}$.

Autoregressive Model: By taking the previous step into consideration, the autoregressive model calculates the second-order transition probability as follows [23]

$$p_{i,j,k} = \frac{p'_{i,j,k}}{\sum_{l \in O_j} p'_{i,j,l}} ,$$

where $p'_{i,j,k} = (1-\alpha)p_{j,k} + \alpha p_{i,k}$. The parameter α $(0 \leq \alpha < 1)$ is a constant to control the strength of effect from the previous step. If $\alpha = 0$, the second-order transition probability degenerates to the first-order transition probability, i.e., $p_{i,j,k} = p_{j,k}$.

The edge-to-edge transition matrix $\mathbf{M}$ based on the autoregressive model can be represented using incidence matrices. Let

$$\mathbf{M}' = (1-\alpha)\mathbf{EH} + \alpha(\mathbf{EB}) \odot (\mathbf{B}^{\mathsf{T}}\mathbf{PE}^{\mathsf{T}}) ,$$

where $\odot$ denotes the Hadamard (entry-wise) product. Then $\mathbf{M}$ is the row normalized $\mathbf{M}'$ such that $\sum_v p_{u,v} = 1$. If $\alpha = 0$, it degenerates to the first-order form and we have $\mathbf{M} = \mathbf{EH}$.

Note that in addition to the two methods discussed above, other methods, such as calculating the edge similarity based on the line graph [20], can also be applied to calculate the edge-to-edge transition probability. In this paper, we only focus on the two methods discussed here.

3.4 Matrix Form

Next we represent the second-order random walk in its matrix form. Let $\mathbf{s}^t$ denote the edge visiting probability vector between time points $(t-1)$ and t, i.e., $\mathbf{s}^t_u = s^t_{(i,j)}$ $(u = (i,j))$. We have

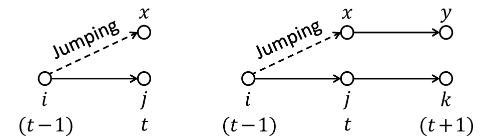
$$\mathbf{s}^{t+1} = \mathbf{M}^{\mathsf{T}}\mathbf{s}^t .$$

If $\mathbf{M}$ is primitive, $\mathbf{s}^t$ converges according to the Perron-Frobenius theorem [15]. Let $\mathbf{s} = \lim_{t \to \infty} \mathbf{s}^t$ denote the edge stationary probability. After having $\mathbf{s}$, the node stationary probability is simply the sum of all in-edge stationary probabilities, i.e., $\mathbf{r} = \mathbf{E}^{\mathsf{T}}\mathbf{s}$.

In the following, we show how to generalize the commonly used proximity measures to their second-order forms. Table 2 summarizes recursive equations of these measures in their first-order and second-order forms. In the table, RW, PR, RR, SR, and SS are shorthand notations for random walk, PageRank, random walk with restart, SimRank, and SimRank* respectively.

Table 2: Recursive equations of various measures

	first-order	second-order
RW	$\mathbf{r} = \mathbf{P}^{\mathsf{T}}\mathbf{r}$	$\mathbf{s} = \mathbf{M}^{\mathsf{T}}\mathbf{s}$
		$\mathbf{r} = \mathbf{E}^{\mathsf{T}}\mathbf{s}$
PR	$\mathbf{r} = c\mathbf{P}^{\mathsf{T}}\mathbf{r} + (1-c)\mathbf{1}/n$	$\mathbf{s} = c\mathbf{M}^{\mathsf{T}}\mathbf{s} + (1-c)\mathbf{H}^{\mathsf{T}}\mathbf{1}/n$
		$\mathbf{r} = c\mathbf{E}^{\mathsf{T}}\mathbf{s} + (1-c)\mathbf{1}/n$
RR	$\mathbf{r} = c\mathbf{P}^{\mathsf{T}}\mathbf{r} + (1-c)\mathbf{q}$	$\mathbf{s} = c\mathbf{M}^{\mathsf{T}}\mathbf{s} + (1-c)\mathbf{H}^{\mathsf{T}}\mathbf{q}$
		$\mathbf{r} = c\mathbf{E}^{\mathsf{T}}\mathbf{s} + (1-c)\mathbf{q}$
SR	$\mathbf{R} = c\mathbf{PRP}^{\mathsf{T}} + (1-c)\mathbf{I}$	$\mathbf{S} = c\mathbf{MSM}^{\mathsf{T}} + (1-c)\mathbf{EE}^{\mathsf{T}}$
		$\mathbf{R} = c\mathbf{HSH}^{\mathsf{T}} + (1-c)\mathbf{I}$
SS	$\mathbf{R} = \frac{c}{2}(\mathbf{PR} + \mathbf{RP}^{\mathsf{T}}) + (1-c)\mathbf{I}$	$\mathbf{S} = \frac{c}{2}(\mathbf{MS} + \mathbf{SM}^{\mathsf{T}}) + (1-c)\mathbf{EE}^{\mathsf{T}}$
		$\mathbf{R}' = \frac{c}{2}\mathbf{MR}' + \frac{c}{2}\mathbf{SH}^{\mathsf{T}} + (1-c)\mathbf{E}$
		$\mathbf{R} = \frac{c}{2}(\mathbf{HR}' + (\mathbf{R}')^{\mathsf{T}}\mathbf{H}^{\mathsf{T}}) + (1-c)\mathbf{I}$



(a) first-order (b) second-order

Figure 5: The jumping process in PageRank

4. THE SECOND-ORDER PAGERANK

In the first-order PageRank, the surfer has a probability of c to follow the node-to-node transition probabilities, and a probability of $(1-c)$ to randomly jump to any node in the graph. Figure 5(a) illustrates the jumping process.

The matrix form of the first-order PageRank is $\mathbf{r} = c\mathbf{P}^{\mathsf{T}}\mathbf{r} + (1-c)\mathbf{1}/n$, where $\mathbf{r}$ is the node visiting probability vector, $\mathbf{P}$ is the node-to-node transition matrix, and $\mathbf{1}$ is a vector of all 1's.

Similarly, in the second-order PageRank, the surfer has a probability of c to follow the edge-to-edge transition probabilities, and a probability of $(1-c)$ to randomly jump to any node in the graph. Its matrix form can be written as $\mathbf{s}^{t+1} = c\mathbf{M}^{\mathsf{T}}\mathbf{s}^t + (1-c)\mathbf{v}$, where $\mathbf{v}$ is the vector corresponding to the jumping process. To determine $\mathbf{v}$, we consider the jumping process in further details.

Figure 5(b) shows the jumping process in the second-order PageRank. At time point $(t-1)$, starting from node i, with probability c, the surfer first visits node j and then k by following the second-order transition probability $p_{i,j,k}$, and with probability $(1-c)$, the surfer randomly jumps to any node x first and then visits node y. After jumping, the effect of the previous step is lost, thus $p_{i,x,y} = p_{x,y}$, which is the first-order transition probability. Since the sum of probabilities to jump to node x is $(1-c)/n$, the probability of visiting edge (x,y) between time points t and $(t+1)$ is $(1-c)p_{x,y}/n$. Thus we have $\mathbf{v} = \mathbf{H}^{\mathsf{T}}\mathbf{1}/n$, where $\mathbf{H}$ is the node-to-edge transition matrix introduced in Section 3.2. Finally, we have

$$\mathbf{s}^{t+1} = c\mathbf{M}^{\mathsf{T}}\mathbf{s}^t + (1-c)\mathbf{H}^{\mathsf{T}}\mathbf{1}/n .$$

THEOREM 1. *In the second-order PageRank, if the out-degree of every node is non-zero, there is a unique edge stationary distribution, i.e.,* $\lim_{t \to \infty} s^t_u = s_u$, *where* s_u *is a constant.*

PROOF. Since the probability distribution vector $\mathbf{s}^t$ sums to 1, i.e., $\mathbf{1}^{\mathsf{T}}\mathbf{s}^t = 1$, we have

$$\mathbf{s}^{t+1} = \big(c\mathbf{M}^{\mathsf{T}} + \tfrac{(1-c)}{n}\mathbf{H}^{\mathsf{T}}\mathbf{1}^{n \times m}\big)\mathbf{s}^t ,$$

where $\mathbf{1}^{n\times m}$ is an $n\times m$ matrix of all 1's. The matrix $\mathbf{T}=c\mathbf{M}^{\mathsf{T}}+\frac{(1-c)}{n}\mathbf{H}^{\mathsf{T}}\mathbf{1}^{n\times m}$ is primitive since $\mathbf{T}$ is irreducible and has positive diagonal elements [15]. Since the out-degree of every node is non-zero, every column of $\mathbf{T}$ sums to 1. Thus, 1 is an eigenvalue of $\mathbf{T}$. By the Perron-Frobenius theorem [15], there is a unique edge stationary distribution and the power method converges. □

The node stationary distribution $\mathbf{r}$ can be obtained from the edge stationary distribution $\mathbf{s}$. The stationary probability of node i equals c times the sum of the edge stationary probabilities on the in-edges of i, plus an additional jumping probability $(1-c)/n$. The formula of $\mathbf{r}$ is given in Table 2.

Random walk with restart is the query biased version of PageRank. In random walk with restart, instead of jumping to every node uniformly, the surfer jumps to the given query node q. Thus, for random walk with restart, the jumping vector is $\mathbf{v}=\mathbf{H}^{\mathsf{T}}\mathbf{q}$, where $\mathbf{q}_q=1$, and $\mathbf{q}_i=0$ if $i\neq q$.

The developed second-order PageRank and random walk with restart degenerate to their original first-order forms when the second-order transition probability is the same as the first-order transition probability, i.e., when $p_{i,j,k}=p_{j,k}$. Please see Appendix A [1] for the proofs.

5. THE SECOND-ORDER SIMRANK

In SimRank, the random walk process involves two random surfers [11]. Next, we first give the preliminary of SimRank and discuss its representation based on meeting paths.

5.1 SimRank and Meeting Paths

The intuition behind SimRank is that two nodes are similar if their in-neighbors are also similar. Let $r_{i,j}$ denote the SimRank proximity value between nodes i and j. SimRank is defined as

$$r_{i,j}=\begin{cases}1\,, & \text{if } i=j\,,\\ \frac{c}{|I_i|\cdot|I_j|}\sum_{k\in I_i}\sum_{l\in I_j} r_{k,l}\,, & \text{if } i\neq j\,,\end{cases}$$

where I_i denotes the set of in-neighbors of node i, and $c\in(0,1)$ is a constant.

The SimRank value $r_{i,j}$ measures the expected number of steps required before two surfers, one starting at node i and the other at node j, meet at the same node if they randomly walk backward, i.e., from a node to one of its in-neighbor nodes, in lock-step [11].

Since walking backward is counter-intuitive, in the following, we study SimRank in the reverse graph, which is obtained by reversing the direction of every edge in the original graph. In the reverse graph, the two random surfers walk forward to a meeting node, and SimRank can be defined as

$$r_{i,j}=\begin{cases}1\,, & \text{if } i=j\,,\\ \frac{c}{|O_i|\cdot|O_j|}\sum_{k\in O_i}\sum_{l\in O_j} r_{k,l}\,, & \text{if } i\neq j\,,\end{cases}$$

where O_i denotes the set of out-neighbors of node i in the reverse graph.

In matrix form, the above recursive definition can be denoted as $\mathbf{R}=c\mathbf{P}\mathbf{R}\mathbf{P}^{\mathsf{T}}+(1-c)\mathbf{I}$, where matrix $\mathbf{R}$ records proximity values for all node pairs with $[\mathbf{R}]_{i,j}=r_{i,j}$ [18, 27, 13].

SimRank values can also be represented as the weighted sum of probabilities of visiting all meeting paths [27].

Definition 1. [Meeting Path] *A meeting path ϕ of length $\{a,b\}$ between nodes i and j in a graph $G(V,E)$ is a sequence of nodes, denoted as $z_0\rightarrow z_1\rightarrow\cdots\rightarrow z_a\leftarrow\cdots\leftarrow z_{b-1}\leftarrow z_b$, such that $i=z_0$, $j=z_b$, $(z_{t-1},z_t)\in E$ for $t=1,2,\cdots,a$, and $(z_t,z_{t-1})\in E$ for $t=a+1,a+2,\cdots,b$.*

A meeting path of length $\{a,b\}$ is symmetric if $b=2a$, such as the meeting path $4\rightarrow2\rightarrow1\leftarrow3\leftarrow5$ in Figure 4(a).

A meeting path $\phi: i=z_0\rightarrow z_1\rightarrow\cdots\rightarrow z_a\leftarrow\cdots\leftarrow z_{b-1}\leftarrow z_b=j$ can be decomposed into two paths $\rho_1: i=z_0\rightarrow z_1\rightarrow\cdots\rightarrow z_a$ and $\rho_2: j=z_b\rightarrow z_{b-1}\rightarrow\cdots\rightarrow z_a$. In the first-order random walk, starting from i, the probability to visit path ρ_1 is $\mathbb{P}[\rho_1]=\prod_{t=1}^{a}p_{z_{t-1},z_t}$. Similarly, starting from j, the probability to visit path ρ_2 is $\mathbb{P}[\rho_2]=\prod_{t=a+1}^{b}p_{z_t,z_{t-1}}$. The probability for the two surfers to visit ϕ and meet at node z_a is thus $\mathbb{P}[\phi]=\mathbb{P}[\rho_1]\cdot\mathbb{P}[\rho_2]$.

Let $\Phi_{i,j}^{a,b}$ denote the set of all meeting paths of length $\{a,b\}$ between nodes i and j, and $\mathbb{P}[\Phi_{i,j}^{a,b}]$ be the sum of probabilities of visiting the meeting paths in $\Phi_{i,j}^{a,b}$. We have the following lemma [27].

Lemma 1. $\mathbb{P}[\Phi_{i,j}^{a,b}]=[\mathbf{P}^a(\mathbf{P}^{\mathsf{T}})^{b-a}]_{i,j}$

Thus the SimRank value $r_{i,j}$ can be represented as

$$r_{i,j}=(1-c)\sum_{t=0}^{\infty}c^t\mathbb{P}[\Phi_{i,j}^{t,2t}]=(1-c)\sum_{t=0}^{\infty}c^t[\mathbf{P}^t(\mathbf{P}^{\mathsf{T}})^t]_{i,j} \quad (1)$$

That is, $r_{i,j}$ is the weighted sum of probabilities of visiting all symmetric meeting paths between nodes i and j. In matrix form, we have

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

5.2 Visiting Meeting Paths in the Second-Order

To develop the second-order SimRank, we need to know the probability of visiting the meeting paths in the second-order. Consider the second-order visiting probability of path $\rho_1: i=z_0\rightarrow z_1\rightarrow\cdots\rightarrow z_a$. Starting from node i, in the first step, the surfer follows the first-order transition probability, and then in the subsequent steps, the surfer follows the second-order transition probabilities. Thus in the second-order random walk, starting from i, the probability to visit path ρ_1 is $\mathbb{M}[\rho_1]=p_{z_0,z_1}\prod_{t=1}^{a-1}p_{z_{t-1},z_t,z_{t+1}}$. Similarly, the probability to visit path ρ_2 is $\mathbb{M}[\rho_2]=p_{z_b,z_{b-1}}\prod_{t=a+1}^{b-1}p_{z_{t+1},z_t,z_{t-1}}$. The probability for the two surfers to visit the meeting path ϕ and meet at node z_a is $\mathbb{M}[\phi]=\mathbb{M}[\rho_1]\cdot\mathbb{M}[\rho_2]$.

Let $\mathbb{M}[\Phi_{i,j}^{a,b}]=\sum_{\phi\in\Phi_{i,j}^{a,b}}\mathbb{M}[\phi]$ be the sum of probabilities of visiting the meeting paths in $\Phi_{i,j}^{a,b}$ in the second-order. The following lemma shows how to compute $\mathbb{M}[\Phi_{i,j}^{a,b}]$ for different cases.

Lemma 2.

$$\mathbb{M}[\Phi_{i,j}^{a,b}]=\begin{cases}\mathbf{I}_{i,j}\,, & \text{if } 0=a=b\,,\\ [\mathbf{H}\mathbf{M}^{a-1}\mathbf{E}]_{i,j}\,, & \text{if } 0<a=b\,,\\ [\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b-1}\mathbf{H}^{\mathsf{T}}]_{i,j}\,, & \text{if } 0=a<b\,,\\ [\mathbf{H}\mathbf{M}^{a-1}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b-a-1}\mathbf{H}^{\mathsf{T}}]_{i,j}\,, & \text{if } 0<a<b\,.\end{cases}$$

Please see Appendix B [1] for the proof.

Lemma 1 for the first-order random walk is a special case of Lemma 2 when the second-order transition probability is the same as the first-order transition probability.

Lemma 3. *If $p_{i,j,k}=p_{j,k}$, we have that $\mathbb{M}[\Phi_{i,j}^{a,b}]=\mathbb{P}[\Phi_{i,j}^{a,b}]$.*

Please see Appendix B [1] for the proof.

Replacing $\mathbb{P}[\Phi_{i,j}^{t,2t}]$ by $\mathbb{M}[\Phi_{i,j}^{t,2t}]$ in Equation (1), we have the second-order SimRank proximity

$$r_{i,j}=(1-c)\sum_{t=0}^{\infty}c^t\mathbb{M}[\Phi_{i,j}^{t,2t}]\,.$$

THEOREM 2. *The matrix form for the second-order SimRank is*

$$\left\{\begin{array}{l} \mathbf{S} = c\mathbf{M}\mathbf{S}\mathbf{M}^{\intercal} + (1-c)\mathbf{E}\mathbf{E}^{\intercal} \\ \mathbf{R} = c\mathbf{H}\mathbf{S}\mathbf{H}^{\intercal} + (1-c)\mathbf{I} \end{array}\right.$$

Please see Appendix C [1] for the proof.

THEOREM 3. *There exists a unique solution to the second-order SimRank.*

PROOF. From the recursive equation of $\mathbf{S}$, we have

$$\mathbf{S} = (1-c)\sum_{t=0}^{\infty} c^t \mathbf{M}^t \mathbf{E}\mathbf{E}^{\intercal}(\mathbf{M}^{\intercal})^t$$

Let $\mathbf{S}^{(\eta)} = (1-c)\sum_{t=0}^{\eta} c^t \mathbf{M}^t \mathbf{E}\mathbf{E}^{\intercal}(\mathbf{M}^{\intercal})^t$. Lemma 5 in Appendix C shows that $\|\mathbf{S}-\mathbf{S}^{(\eta)}\|_{\max} \leq c^{\eta+1}$ for any η $(\eta \geq 0)$. The convergence of the series follows directly from Lemma 5 and $\lim_{\eta\to\infty} c^{\eta+1} = 0$ $(0<c<1)$. Thus, $\mathbf{S}$ exists.

Next, we prove that $\mathbf{S}$ is unique. Suppose that $\mathbf{S}$ and $\mathbf{S}'$ are two solutions and we have

$$\left\{\begin{array}{l} \mathbf{S} = c\mathbf{M}\mathbf{S}\mathbf{M}^{\intercal} + (1-c)\mathbf{E}\mathbf{E}^{\intercal} \\ \mathbf{S}' = c\mathbf{M}\mathbf{S}'\mathbf{M}^{\intercal} + (1-c)\mathbf{E}\mathbf{E}^{\intercal} \end{array}\right.$$

Let $\Delta = \mathbf{S}-\mathbf{S}'$ be the difference. We have $\Delta = c\mathbf{M}\Delta\mathbf{M}^{\intercal}$. Let $|\Delta_{u,v}| = \|\Delta\|_{\max}$ for some $u, v \in E$. We have

$$\begin{aligned} \|\Delta\|_{\max} &= |\Delta_{u,v}| = c\cdot\left|[\mathbf{M}]_{u,:}\cdot\Delta\cdot([\mathbf{M}]_{v,:})^{\intercal}\right| \\ &\leq c\sum_{x\in O_u}\sum_{y\in O_v} p_{u,x}\cdot p_{v,y}\cdot\left|[\Delta]_{x,y}\right| \\ &\leq c\sum_{x\in O_u}\sum_{y\in O_v} p_{u,x}\cdot p_{v,y}\cdot\|\Delta\|_{\max} = c\cdot\|\Delta\|_{\max}\ , \end{aligned}$$

where O_u denotes the set of out-neighbor edges of edge u. Since $0<c<1$, we have that $\|\Delta\|_{\max} = 0$ and $\mathbf{S} = \mathbf{S}'$. Thus, $\mathbf{S}$ is unique.

Given that $\mathbf{S}$ exists and is unique, $\mathbf{R}$ also exists and is unique. □

SimRank* [27] is a variant of SimRank that considers non-symmetric meeting paths. Following a similar approach, we can develop the matrix form for the second-order SimRank*. The equations are summarized in Table 2.

The second-order SimRank degenerates to its original first-order form when the second-order transition probability is the same as the first-order transition probability. Please see Appendix A [1] for the proof.

6. COMPUTING ALGORITHMS

In this section, we discuss how to efficiently compute the developed second-order measures. We first study the power iteration method which utilizes the recursive definitions to compute the exact proximity values. This method needs to iterate over the entire graph thus the complexity is high. To speed up the computation, we develop Monte Carlo methods, which are randomized algorithms and provide a trade-off between accuracy and efficiency. We formally prove that the estimated value (1) converges to the exact proximity value when the sample size is large, and (2) is sharply concentrated around the exact value.

6.1 The Power Iteration Method

Given the recursive equations in Table 2, we can apply the power iteration methods to compute the second-order measures. For example, the power method computes the second-order PageRank as follows

$$\mathbf{s}^t = \begin{cases} \mathbf{H}^{\intercal}\mathbf{1}/n\,, & \text{if } t=0\,, \\ c\mathbf{M}^{\intercal}\mathbf{s}^{t-1} + (1-c)\mathbf{H}^{\intercal}\mathbf{1}/n\,, & \text{if } t>0\,. \end{cases}$$

Algorithm 1 The MC algorithm for the first-order RR [8]

```
Input: $G(V,E)$, query node $q$, decay factor $c$, sample size $\pi$
Output: estimated proximity vector $\tilde{\mathbf{r}}$
1: for each node $i \in V$ do $\tilde{r}_i \leftarrow 0$;                // initialization
2: repeat $\pi$ times
3:     $a \leftarrow$ generate a random number following the geometric
           distribution $\mathbb{P}[\mathbb{A}=a] = (1-c)\cdot c^a$;
4:     $z_0 \leftarrow q$; bSuccess $\leftarrow$ true;
5:     for $t \leftarrow 1$ to $a$ do
6:         if $|O_{z_{t-1}}| = 0$ then bSuccess $\leftarrow$ false, break;
7:         $z_t \leftarrow$ randomly pick a node from $O_{z_{t-1}}$ according to
               the first-order transition probability;
8:     if bSuccess = true then $\tilde{r}_{z_a} \leftarrow \tilde{r}_{z_a} + 1$;
9: for each node $i \in V$ do $\tilde{r}_i \leftarrow \tilde{r}_i/\pi$;          // normalization
```

Let $\mathbf{s}$ be the converged edge stationary vector. We can then compute node stationary vector $\mathbf{r} = c\mathbf{E}^{\intercal}\mathbf{s} + (1-c)\mathbf{1}/n$.

Time Complexity : Let $\sigma = \sum_{i\in V}|I_i|\cdot|O_i|$ denote the number of second-order transition probabilities, i.e., the number of non-zero elements in matrix $\mathbf{M}$. In each iteration, the matrix-vector product $\mathbf{M}^{\intercal}\mathbf{s}^{t-1}$ needs $O(\sigma)$ time. Suppose that the power method needs β iteration to converge. It runs in $O(\beta\sigma)$ time for the second-order PageRank. Similarly, the power method for the second-order random walk with restart also runs in $O(\beta\sigma)$ time.

The power iteration method computes the second-order SimRank as follows

$$\mathbf{S}^{(t)} = \begin{cases} (1-c)\mathbf{E}\mathbf{E}^{\intercal}\,, & \text{if } t=0\,, \\ c\mathbf{M}\mathbf{S}^{(t-1)}\mathbf{M}^{\intercal} + (1-c)\mathbf{E}\mathbf{E}^{\intercal}\,, & \text{if } t>0\,. \end{cases}$$

Let $\mathbf{S}$ be the converged edge proximity matrix. We then compute the node proximity matrix as $\mathbf{R} = c\mathbf{H}\mathbf{S}\mathbf{H}^{\intercal} + (1-c)\mathbf{I}$.

Time Complexity: In each iteration, the matrix-matrix products $\mathbf{M}\mathbf{S}^{(t-1)}\mathbf{M}^{\intercal}$ need $O(m\sigma)$ time, where m is the number of edges in the graph and σ is the number of non-zero elements in $\mathbf{M}$. Suppose that the power method needs β iteration to converge. It runs in $O(\beta m\sigma)$ time for the second-order SimRank. Similarly, the power method for the second-order SimRank* also runs in $O(\beta m\sigma)$.

6.2 The Monte Carlo Method

Monte Carlo (MC) methods have been recently studied to compute the first-order random walk with restart [8] and SimRank [7]. Next, we develop MC methods to compute the second-order random walk with restart and SimRank, and provide the theoretical analysis for the developed methods.

6.2.1 Computing Random Walk with Restart

To illustrate the basic idea, we begin with the MC algorithm for the first-order random walk with restart [8], which is shown in Algorithm 1. It is based on the following series expansion of random walk with restart

$$r_i = (1-c)\sum_{t=0}^{\infty} c^t \mathbb{P}[\Phi_{i,q}^{0,t}] = (1-c)\sum_{t=0}^{\infty} c^t \mathbb{P}[\Phi_{q,i}^{t,t}]$$

That is, the proximity r_i can be represented as the weighted sum of probabilities of visiting all paths from node q to i. The longer the path length t, the smaller the weight $(1-c)\cdot c^t$. Based on this interpretation, in line 3 of Algorithm 1, the MC method determines the path length a based on the geometric distribution. Then, starting from the query node q, the algorithm simulates a path of length a in lines 4∼7. When simulating a path, at each time point t $(1\leq t\leq a)$, the algorithm randomly picks an out-neighbor

Algorithm 2 The MC algorithm for the second-order RR

```
Input: $G(V,E)$, query node $q$, decay factor $c$, sample size $\pi$
Output: estimated proximity vector $\tilde{\mathbf{r}}$
1: for each node $i \in V$ do $\tilde{r}_i \leftarrow 0$;        // initialization
2: repeat $\pi$ times
3:     $a \leftarrow$ generate a random number following the geometric
           distribution $\mathbb{P}[\mathbb{A}=a]=(1-c)\cdot c^a$;
4:     $z_0 \leftarrow q$; bSuccess $\leftarrow$ true;
5:     for $t \leftarrow 1$ to $a$ do
6:         if $|O_{z_{t-1}}|=0$ then bSuccess $\leftarrow$ false, break;
7:         if $t=1$ then $z_t \leftarrow$ randomly pick a node from $O_{z_{t-1}}$
               according to the first-order transition probability;
8:         else $z_t \leftarrow$ randomly pick a node from $O_{z_{t-1}}$ according
               to the second-order transition probability;
9:     if bSuccess = true then $\tilde{r}_{z_a} \leftarrow \tilde{r}_{z_a}+1$;
10: for each node $i \in V$ do $\tilde{r}_i \leftarrow \tilde{r}_i/\pi$;        // normalization
```

of the previous node z_{t-1} to visit according to the first-order transition probability. The algorithm stops when the simulated path reaches length a or there is no out-neighbor to pick. The algorithm repeats this process π times, where π is the number of paths to be simulated. The proximity value of node i is estimated as the fraction of the π paths that end at i.

We can extend this MC algorithm to compute the second-order random walk with restart as shown in Algorithm 2. It is based on the series expansion

$$r_i = (1-c)\sum_{t=0}^{\infty} c^t \mathbb{M}[\Phi_{i,q}^{0,t}] = (1-c)\sum_{t=0}^{\infty} c^t \mathbb{M}[\Phi_{q,i}^{t,t}]$$

That is, the proximity r_i can be represented as the weighted sum of probabilities of visiting all paths from node q to i in the second-order random walk. The difference between Algorithm 1 and Algorithm 2 is how to sample a path. In lines $5 \sim 7$ of Algorithm 1, at each step, the algorithm picks an out-neighbor with the first-order transition probability. In lines $5 \sim 8$ of Algorithm 2, only in the first step, i.e., when $t=1$, the algorithm picks an out-neighbor with the first-order transition probability. Then the algorithm picks out-neighbors with the second-order transition probabilities when simulating subsequent steps.

Theorem 4 shows that when the sample size is large, the estimated proximity $\tilde{r}_i$ converges to the exact proximity r_i. Theorem 5 shows that the error is bounded by a term that is exponentially small in terms of the sample size.

Theorem 4. *The estimated proximity $\tilde{r}_i$ converges to the exact proximity r_i when $\pi \to \infty$.*

Proof. In Algorithm 2, if we successfully sample a path ending at node i, we will increase $\tilde{r}_i$ by 1; otherwise, $\tilde{r}_i$ is unchanged. Let $\mathbb{S}_i^{(d)}$ be a Bernoulli random variable denoting the incremental value of $\tilde{r}_i$ at the d-th iteration (lines $3 \sim 9$). Random variables $\mathbb{S}_i^{(1)}, \mathbb{S}_i^{(2)}, \cdots, \mathbb{S}_i^{(\pi)}$ are independent and identically distributed. Let $\mathbb{S}_i$ be a Bernoulli random variable following the same distribution as $\mathbb{S}_i^{(d)}$'s. Lemma 6 in Appendix D [1] shows that the expected value of $\mathbb{S}_i$ equals the exact proximity r_i, i.e., $\mathbb{E}[\mathbb{S}_i] = r_i$.

Let $\overline{\mathbb{S}}_i = \frac{1}{\pi}\sum_{d=1}^{\pi} \mathbb{S}_i^{(d)}$ be the sample average, which represents the estimated proximity $\tilde{r}_i$. By the law of large numbers, if the sample size $\pi \to \infty$, $\overline{\mathbb{S}}_i$ converges to the expected value $\mathbb{E}[\mathbb{S}_i] = r_i$. □

Theorem 5. *For any $\epsilon > 0$, we have that*

$$\mathbb{P}[|\tilde{r}_i - r_i| \geq \epsilon] \leq 2 \cdot \exp(-2\pi\epsilon^2)$$

Proof. Following the notations defined in the proof of Theorem 4, random variables $\mathbb{S}_i^{(1)}, \mathbb{S}_i^{(2)}, \cdots, \mathbb{S}_i^{(\pi)}$ are independent and bounded by interval $[0,1]$. By Hoeffding's inequality [10], we can prove this theorem. □

Algorithm 3 The basic MC algorithm for the first-order SR [7]

```
Input: $G(V,E)$, query node $q$, decay factor $c$, sample size $\pi$,
       maximum length $\eta$
Output: estimated proximity vector $\tilde{\mathbf{r}}$
for each node $i \in V$ do                    // process each node individually
    $\tilde{r}_i \leftarrow 0$;                        // initialization
    repeat $\pi$ times                        // sample $\pi$ pairs of paths
        sample a path $q = z_0 \to z_1 \to \cdots \to z_\eta$ starting from $q$;
        sample a path $i = z'_0 \to z'_1 \to \cdots \to z'_\eta$ starting from $i$;
        // find the common node with the smallest offset
        for $t=0$ to $\eta$ do   if $z_t = z'_t$ then $\tilde{r}_i \leftarrow \tilde{r}_i + c^t$, break;
    $\tilde{r}_i \leftarrow \tilde{r}_i/\pi$;                     // normalization
```

Time Complexity : Generating a random number costs $O(1)$ time. Since the path length a follows the geometric distribution, the average length is $(1-c)\sum_{a=0}^{\infty} a\, c^a = c/(1-c)$. When sampling a path, at each step, the algorithm randomly picks an out-neighbor, which costs $O(\xi)$ on average, where $\xi = \frac{1}{n}\sum_{i \in V}|O_i|$ denotes the average out-degree. Thus, on average, sampling a path costs $O(\xi c/(1-c))$ time. Sampling π paths costs $O(\pi \xi c/(1-c))$ time. Initialization and normalization cost $O(n)$ time. In total, Algorithm 2 runs in $O(\pi \xi c/(1-c)+n)$ time.

The MC algorithm developed here is readily applicable to compute the second-order PageRank. Let $\mathrm{PR}(i)$ denote the PageRank value of node i, and $\mathrm{RR}_j(i)$ denote the random walk with restart proximity of node i when j is the query.

Theorem 6. $\mathrm{PR}(i) = \frac{1}{n}\sum_{j \in V} \mathrm{RR}_j(i)$

Proof. The proof is similar to that of the linearity theorem [12]. We omit the proof here due to the space limit. □

Based on this theorem, we can use Algorithm 2 to compute the random walk with restart proximity vector for each node. The average of all vectors is the estimated PageRank vector.

6.2.2 Computing SimRank

The basic MC algorithm for computing the first-order SimRank is proposed in [7], which is shown in Algorithm 3. It is based on the original interpretation of SimRank [11], i.e., the proximity r_i measures the expected number of steps required before two surfers, one starting at the query node q and the other at node i, meet at the same node if they randomly walk on the reverse graph in lock-step.

As shown in Algorithm 3, the algorithm proposed in [7] directly simulates the meeting paths of the two surfers. For each node i, it simulates two paths of length η, one starting from node q and one from i. It then scans these two paths to determine whether there is a common node. The fraction of the sampled paths that do have a common node is used as the estimated proximity value for node i.

Algorithm 3 estimates the proximity of each node i individually and samples a fixed number of paths for each node. This method usually needs to simulate a large number of meeting paths to achieve an accurate estimation since not all simulated paths may have common nodes. The paths that do not have common nodes can only be used in the denominator in the estimated proximity value.

Algorithm 4 The proposed MC algorithm for the first-order SR

```
Input: $G(V,E)$, query node $q$, decay factor $c$, sample size $\pi$,
       maximum length $\eta$, matrix $\mathbf{X}$
Output: estimated proximity vector $\tilde{\mathbf{r}}$
1: for each node $i \in V$ do $\tilde{r}_i \leftarrow 0$;                // initialization
2: repeat $\pi$ times                                        // sample $\pi$ meeting paths
3:     $a \leftarrow$ generate a random number following the geometric
           distribution $\mathbb{P}[\mathbb{A}=a]=(1-c)\cdot c^a$;
4:     if $a > \eta$ then continue;
5:     [bSuccess, $z_{2a}$, $\delta$] $\leftarrow$ SampleOneMeetingPath($q, a, \mathbf{X}$);
6:     if bSuccess = true then $\tilde{r}_{z_{2a}} \leftarrow \tilde{r}_{z_{2a}} + \delta$;
7: for each node $i \in V$ do $\tilde{r}_i \leftarrow \tilde{r}_i/\pi$;          // normalization
```

Algorithm 5 [bSuccess, z_{2a}, δ] $\leftarrow$ SampleOneMeetingPath($q, a, \mathbf{X}$)

```
1: $z_0 \leftarrow q$; bSuccess $\leftarrow$ true;                    // start from the query node
2: for $t \leftarrow 1$ to $a$ do                                  // sample the first half
3:     if $|O_{z_{t-1}}| = 0$ then bSuccess $\leftarrow$ false, return;
4:     $z_t \leftarrow$ randomly pick a node from $O_{z_{t-1}}$ according to the
           first-order transition probability;
5: for $t \leftarrow (a+1)$ to $2a$ do                             // sample the second half
6:     if $|I_{z_{t-1}}| = 0$ or $[\mathbf{X}]_{z_{t-1},2a-t+1} = 0$ then bSuccess $\leftarrow$ false, return;
7:     $z_t \leftarrow$ randomly pick a node from $I_{z_{t-1}}$ according to the
           probability $p_{z_t,z_{t-1}} \cdot [\mathbf{X}]_{z_t,2a-t}/[\mathbf{X}]_{z_{t-1},2a-t+1}$;
8: $\delta \leftarrow [\mathbf{X}]_{z_a,a}/[\mathbf{X}]_{z_{2a},0}$;
```

Algorithm 6 ComputeNodeVisitingProbabilities()

```
Input: $G(V,E)$, transition matrix $\mathbf{P}$, maximum length $\eta$
Output: $n \times (\eta+1)$ matrix $\mathbf{X}$
1: $[\mathbf{X}]_{:,0} \leftarrow \mathbf{1}/n$;       // begin from the uniform probability distribution
2: for $t \leftarrow 1$ to $\eta$ do $[\mathbf{X}]_{:,t} \leftarrow \mathbf{P}^{\mathsf{T}}[\mathbf{X}]_{:,t-1}$;
```

Next, we propose a new sampling strategy to compute the SimRank values. Our sampling method estimates the proximity values for all the nodes at the same time. Every simulated path is guaranteed to contribute to the numerator of some node and to the denominators of all nodes. Experimental results show that compared to the previous method, our sampling method needs several orders of magnitude less simulated paths to achieve the same accuracy.

For simplicity, next, we illustrate the key idea of the developed sampling strategy for the first-order SimRank. It can be easily extended to the second-order SimRank.

Algorithm 4 shows the overall procedure of the proposed algorithm, which is based on the series expansion

$$r_i = (1-c)\sum_{t=0}^{\infty} c^t \mathbb{P}[\Phi_{i,q}^{t;2t}] = (1-c)\sum_{t=0}^{\infty} c^t \mathbb{P}[\Phi_{q,i}^{t;2t}]$$

That is, the proximity r_i can be represented as the weighted sum of probabilities of visiting all meeting paths of length $\{t, 2t\}$ between nodes q and i.

Instead of simulating meeting paths starting from both nodes q and i, we only simulate paths starting from q. Algorithm 5 shows the procedure to sample a meeting path. For a meeting path $\phi: q=z_0 \to z_1 \to \cdots \to z_a \leftarrow \cdots \leftarrow z_{2a-1} \leftarrow z_{2a}=i$, when simulating the first half of the path, we simply follow the first-order transition probability. Since the second half of the meeting path is in reverse order, we need to pick in-neighbors of the visited nodes. To do that, we need to know the probability of visiting in-neighbors. We can use Bayes' theorem to calculate these probabilities.

Let $\mathbb{X}_t$ be a random variable representing the node visited by the surfer at time t. Suppose that node j is an in-neighbor of k, i.e., $j \in I_k$. We have

$$\mathbb{P}[\mathbb{X}_{t-1}{=}j|\mathbb{X}_t{=}k] = \frac{\mathbb{P}[\mathbb{X}_{t-1}{=}j]}{\mathbb{P}[\mathbb{X}_t=k]} \cdot \mathbb{P}[\mathbb{X}_t{=}k|\mathbb{X}_{t-1}{=}j] = \frac{\mathbb{P}[\mathbb{X}_{t-1}{=}j]}{\mathbb{P}[\mathbb{X}_t=k]} \cdot p_{j,k}$$

Thus to calculate the probability of visiting in-neighbors, we only need the prior probability of visiting each node. Algorithm 6 computes these probabilities and stores them in an $n \times (\eta+1)$ matrix $\mathbf{X}$ in the preprocessing stage, with $[\mathbf{X}]_{j,t} = \mathbb{P}[\mathbb{X}_t = j]$, where η is the maximum length of the simulated paths.

Let $\hat{r}_i = (1-c)\sum_{t=0}^{\eta} c^t \mathbb{P}[\Phi_{q,i}^{t;2t}]$ be the SimRank value we try to estimate. Next, we show that our sampling strategy gives accurate estimation of $\hat{r}_i$.

Theorem 7. *The estimated proximity $\tilde{r}_i$ converges to $\hat{r}_i$ when $\pi \to \infty$.*

Proof. In Algorithm 4, if we successfully sample a path ending at node i, we will increase $\tilde{r}_i$ by δ; otherwise, $\tilde{r}_i$ is unchanged. For different sampled paths, the corresponding value δ may be different. Let $\mathbb{R}_i^{(d)}$ be a random variable denoting the incremental value of $\tilde{r}_i$ at the d-th iteration (lines $3 \sim 6$). Random variables $\mathbb{R}_i^{(1)}, \mathbb{R}_i^{(2)}, \cdots, \mathbb{R}_i^{(\pi)}$ are independent and identically distributed. Let $\mathbb{R}_i$ be a random variable following the same distribution as $\mathbb{R}_i^{(d)}$'s. Lemma 7 in Appendix D [1] shows that the expected value of $\mathbb{R}_i$ equals the truncated proximity $\hat{r}_i$, i.e., $\mathbb{E}[\mathbb{R}_i] = \hat{r}_i$.

Let $\overline{\mathbb{R}}_i = \frac{1}{\pi}\sum_{d=1}^{\pi} \mathbb{R}_i^{(d)}$ be the sample average, which represents the estimated proximity $\tilde{r}_i$. By the law of large numbers, if the sample size $\pi \to \infty$, $\overline{\mathbb{R}}_i$ converges to the expected value $\mathbb{E}[\mathbb{R}_i] = \hat{r}_i$. □

Theorem 8. *For any $\epsilon > 0$, we have that* $\mathbb{P}[\tilde{r}_i - \hat{r}_i \le -\epsilon] \le \exp(\frac{-\pi\epsilon^2}{2nr_i})$ *and* $\mathbb{P}[\tilde{r}_i - \hat{r}_i \ge \epsilon] \le \exp(\frac{-\pi\epsilon^2}{2nr_i + 2n\epsilon/3})$.

Proof. Following the notations defined in the proof of Theorem 7, random variables $\mathbb{R}_i^{(1)}, \mathbb{R}_i^{(2)}, \cdots, \mathbb{R}_i^{(\pi)}$ are independent and bounded by interval $[0, \delta] \subseteq [0, n]$. Lemma 7 in Appendix D [1] shows that the expected value of $\mathbb{R}_i^2$ is bounded from above by nr_i, i.e., $\mathbb{E}[\mathbb{R}_i^2] \le nr_i$. Thus, we have that $\sum_{d=1}^{\pi} \mathbb{E}[(\mathbb{R}_i^{(d)})^2] \le \pi n r_i$. By Theorem 14 in Appendix D [1], we can prove this theorem. □

Time Complexity : Since the path length a follows the geometric distribution, the average length is $(1-c)\sum_{a=0}^{\infty} 2ac^a = 2c/(1-c)$. When sampling a path in Algorithm 5, at each step, the algorithm randomly picks an out-neighbor or in-neighbor, which costs $O(\psi)$ time on average, where $\psi = \frac{1}{2n}\sum_{i \in V}\big(|I_i| + |O_i|\big)$ denotes the average degree. Thus, on average, sampling a path costs $O(\psi c/(1-c))$. Sampling π paths costs $O(\pi\psi c/(1-c))$. Initialization and normalization cost $O(n)$. In total, Algorithm 4 runs in $O(\pi\psi c/(1-c) + n)$ time. Algorithm 6 runs in $O(m\eta)$ time.

The proposed MC algorithm is readily applicable to the second-order SimRank. The only difference is that in the second-order SimRank, we need to follow the second-order transition probability when sampling meeting paths. Theorems 7 and 8 also apply when we follow the second-order transition probability.

7. EXPERIMENTAL RESULTS

In this section, we perform comprehensive experimental evaluations on the developed methods. To evaluate the effectiveness of the developed second-order proximity measures,

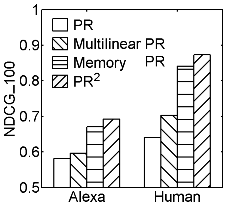


Figure 6: Ranking accuracy

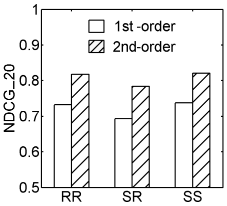
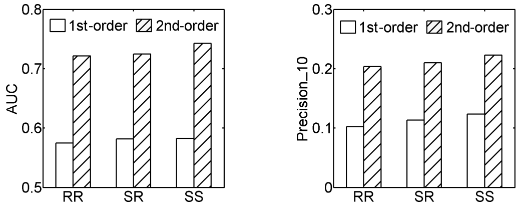


Figure 7: Query accuracy



(a) AUC (b) Precision_{10}

Figure 8: Link prediction on the Twitter follower network

we use both networks with and without the data flow information. We also evaluate the efficiency of proposed computing methods on large real and synthetic networks.

All programs are written in C++, and all experiments are performed on a server with 32G memory, Intel Xeon 3.2GHz CPU, and Redhat OS.

7.1 Networks with Data Flow Information

We first evaluate the effectiveness of the second-order measures using networks with data flow information.

7.1.1 Web Domain Network with Clickstream Data

In the web domain network, each node represents a domain, and an edge is weighted by the number of hyperlinks between pages contained in the two connected domains. The web graph was gathered in 2012 and is publicly available at `http://webdatacommons.org/hyperlinkgraph/` [17]. It contains $463,824$ domains and $6,285,354$ edges.

Clickstream data records the sequences of domains visited by different users [3]. The clickstream data is obtained from comScore Inc. It contains $5,000$ users' clickstreams recorded over 6 months in 2012. The total number of visits is 62.4 million. Each domain was visited 135 times on average.

We first evaluate the domain ranking results of PageRank (PR). The original first-order PR [22], multilinear PR [9], memory PR [24], and our second-order PR (PR^2) are used for comparison. PR uses the first-order transition probability. Multilinear PR approximates the stationary probability of an edge by the product of stationary probabilities of its two end nodes. Memory PR and PR^2 use the second-order transition probabilities based on the frequencies of the trigrams in the clickstream data as discussed in Section 3.3.

We use Alexa's top domains (`http://alexa.com/topsites`) as the reference to evaluate the ranking results of the selected measures. The *Normalized Discounted Cumulative Gain* (NDCG) is used as the evaluation metric [27]. The NDCG value at position k is $\text{NDCG}_k = \beta \sum_{i=1}^{k} (2^{s(i)} - 1)/\log_2(1+i)$, where $s(i)$ is the score of the i-th node and β is a normalizing factor to ensure the NDCG value of the ground-truth ordering to be 1. We use NDCG_{100} to evaluate the top-100 ranked domains of the selected measures. A retrieved domain gets a score of 1 if it appears in the top-100 domains in Alexa's top domains; otherwise, it gets a score of 0.

In addition to using Alexa's top domains as the reference, we also hired 10 human evaluators to manually evaluate the retrieved domains. An evaluator gives an importance score (ranging from 1 to 5, with 5 being the most important) to each retrieved domain. For each domain, the average score of all evaluators is used as its final score. NDCG_{100} is then calculated as the evaluation metric.

Figure 6 shows the NDCG scores of the selected methods. As we can see, using both Alexa's top domains and human evaluators as references, the second-order measures, PR^2, memory PR and multilinear PR, perform better than the original first-order PR. Among the second-order measures, PR^2 and memory PR have higher accuracy than multilinear PR, since the second-order transition probabilities used in PR^2 and memory PR are obtained from the network flow data while the probabilities used in multilinear PR are estimated from the first-order transition probabilities. The better performance of PR^2 over memory PR indicates that the jumping strategy in PR^2 is more effective than the uniform jumping strategy in memory PR.

Next, we evaluate the effectiveness of the proposed measures for the top-k query problem. The evaluated measures include the first-order random walk with restart (RR), SimRank (SR) and SimRank* (SS) and their second-order forms developed in this paper. We randomly select a web domain as the query node, and retrieve the top-20 most relevant domains using the selected measures. We repeat the experiment 100 times. Since there is no ground truth about the proximities between the query nodes and the retrieved nodes, we use the 10 human evaluators to evaluate the relevance of retrieved domains. The relevance score ranges from 1 to 5 with 5 being the most relevant. The average score of all evaluators is used as the final score for a domain. NDCG_{20} then is calculated as the evaluation metric.

Figure 7 shows the accuracy of the selected measures in their first-order and second-order forms. We can see that each second-order measure is more accurate than its first-order counterpart. The second-order measures utilize the real clickstream data to compute the second-order transition probabilities. Since the clickstream data faithfully reflects the similarity among the domains, by leveraging such information, the second-order measures can dramatically improve the accuracy of the results.

7.1.2 Twitter Network with Tweet Cascade Data

A node in the Twitter follower network represents a user and an edge (i,j) represents that user j follows user i. The Twitter follower network used in our experiments was crawled on November 2014. The network contains $231,624$ nodes and $3,214,581$ edges. We query the timeline from December 2014 to February 2015 of each user once per day to monitor the tweet cascades. The second-order transition probabilities are computed based on frequency of the trigrams in these tweet cascades.

We use the link prediction accuracy to evaluate the effectiveness of the second-order measures. For a given query node, the top ranked nodes that are not followers of the query node are predicted to follow the query node. The followers newly emerged from March to May 2015 are used as the ground truth to evaluate the predicted results.

We use AUC (area under the ROC curve) and Precision to evaluate the accuracy [21]. AUC can be interpreted as the probability that a randomly chosen user that newly follows the query node is given a higher score than a randomly chosen user that does not follow the query node. Precision is defined as $\text{Precision}_k = k'/k$, where k is the total number of predicted users, and k' is the number of users that actually started to follow the query node. We randomly pick 10^3 query nodes and report the average.

Figure 8(a) shows the AUC values of the first-order and second-order RR, SR, and SS. We can observe that the second-order measures improve the AUC values by $23 \sim 28\%$ compared to their first-order counterparts. Figure 8(b) shows the Precision_{10} values. Similarly, the second-order measures outperform the first-order measures consistently. The real tweet cascade data reflects how tweets propagate among different users and provides more accurate transition information than the network topology alone does. The second-order measures use such information thus have better performance.

7.2 Networks without Data Flow Information

When the network data flow information is not available, we use three different applications, including local community detection, link prediction, and graph-based semi-supervised learning, to evaluate the effectiveness of the developed second-order proximity measures. We use the autoregressive model discussed in Section 3.3 to obtain the second-order transition probabilities.

7.2.1 Local Community Detection

The goal of local community detection is to find the community near a given query node [2, 26]. Intuitively, the identified local community should contain the nodes having large proximity to the query node. We use the query biased densest connected subgraph (QB) method [26] and the PageRank-Nibble (NB) method [2] to evaluate the developed second-order measure. For a given query node, both QB and NB compute the node proximity values and use them to find a set of top-ranked nodes as the identified local community. Both methods use the first-order random walk with restart (RR) as their proximity measure. We simply replace the first-order RR with the proposed second-order RR (RR^2) in QB and NB. All other parts in QB and NB remain the same as the original algorithms. The second-order transition probability is computed by the autoregressive model and the default setting for α is 0.2.

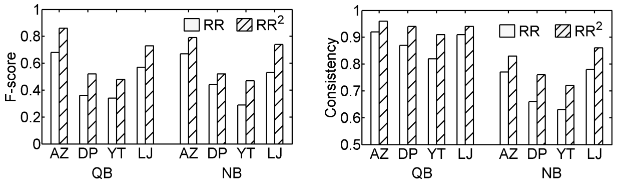
We use F-score and consistency [26] as the evaluation metrics. F-score measures the accuracy of the detected community with regard to the ground-truth community labels. Consistency measures the standard deviation of F-scores of the identified communities when different nodes in the same community are used as the query nodes. A high consistency value indicates that the method tends to find the same local community no matter which node in it is used as the query. We randomly pick 10^3 query nodes and report the average.

We first use real networks to evaluate the performance of the second-order measure. Table 3 shows the statistics of real networks. These datasets are provided with ground-truth community memberships and are publicly available at `http://snap.stanford.edu`.

Figure 9(a) shows the F-scores on these networks. QB with RR^2 outperforms QB with RR for $26 \sim 44\%$. NB with RR^2 outperforms NB with RR for $18 \sim 62\%$. Using RR^2, the random surfer is more likely to be trapped within the local community containing the query node, since it takes the previous step of the surfer into consideration. This results that the nodes in the local community have larger proximity values than the nodes outside the community. It helps improve the accuracy of the local community detection methods.

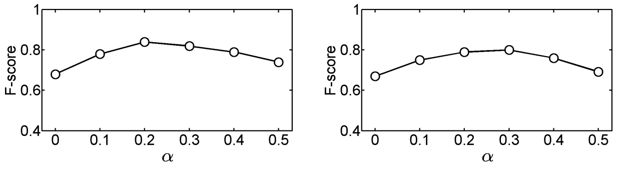
Table 3: Statistics of real networks

datasets	abbr.	#nodes	#edges	#communities
Amazon	AZ	334,863	925,872	151,037
DBLP	DP	317,080	1,049,866	13,477
Youtube	YT	1,134,890	2,987,624	8,385
LiveJournal	LJ	3,997,962	34,681,189	287,512



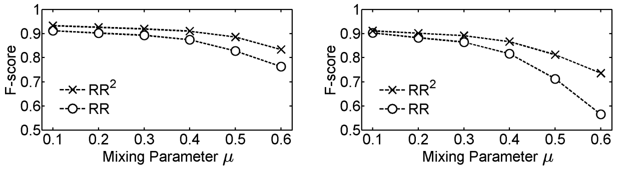
(a) F-score (b) consistency

Figure 9: F-scores and consistency values on real networks



(a) the QB method (b) the NB method

Figure 10: Tuning the parameter α (Amazon network, RR^2)



(a) the QB method (b) the NB method

Figure 11: F-scores on synthetic networks

Figure 9(b) shows the consistency results. QB with RR^2 outperforms QB with RR for $3 \sim 11\%$. NB with RR^2 outperforms NB with RR for $8 \sim 15\%$. High consistency is important for local community detection, since the identified communities should be similar even if different nodes in the same community are used as the query. The higher consistency value of RR^2 demonstrates that it better captures the community structures.

Next we evaluate the sensitivity of RR^2 with respect to the tuning parameter α. Figure 10(a) shows the F-scores of QB with RR^2 on the Amazon network for different α values. We can see that the performance is stable when varying α. When $\alpha = 0.2$, QB has the best performance. Figure 10(b) shows the results of NB with RR^2. A similar trend can be observed. Note that when $\alpha = 0$, RR^2 degenerates to RR and has the same performance as RR.

In addition to real networks, we also generate a collection of synthetic networks using the graph generator in [14] to evaluate the developed second-order measures. The number of nodes in the network is 2^{20} and the number of edges is 10^7. The network generating model contains a mixing parameter μ, which indicates the proportion of a node's neighbors that reside in other communities. By tuning μ, we can vary the

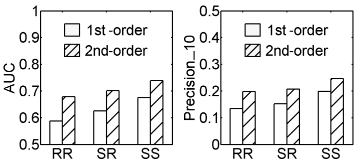


Figure 12: Link prediction on the co-author network

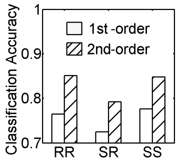


Figure 13: Graph-based semi-supervised learning

clearness of the community structure: the boundaries between different communities become less clear for larger μ values.

Figure 11 shows the F-scores on synthetic networks when using QB and NB to detect the local communities. As we can see, RR^2 achieves better performance than RR consistently. Moreover, the performance gap between RR^2 and RR becomes larger for larger μ. This demonstrates that RR^2 is more robust to the noise in the networks than RR. The reason is that in RR^2 the random surfer is likely to stay in the same community rather than to walk across the boundary of the community.

7.2.2 Link Prediction

We further evaluate the link prediction accuracy of the second-order measures using the DBLP co-author network. A node in the network represents an author and the edge weight represents the number of papers that two connected authors have co-authored. We select the papers published in the database conferences including SIGMOD, VLDB, ICDE, EDBT, ICDT, and PODS, and data mining conferences including KDD, ICDM, SDM, PKDD, and PAKDD, from 2004 to 2013. The papers published in 2004 to 2008 are used to construct the training network, which contains 146,527 nodes and 426,835 edges. The papers published in 2009 to 2013 are used to obtain the newly emerged links among the authors, which are used as the ground truth for testing.

The left figure in Figure 12 shows the AUC values. We can see that the second-order measures improve the AUC value by 9$\sim$16% compared to the first-order measures. The right figure shows the Precision$_{10}$ values. The second-order measures improve the Precision$_{10}$ value by 24$\sim$47%. Since the second-order measures better capture the community structure in the network, they can significantly improve the link prediction accuracy.

7.2.3 Graph-Based Semi-Supervised Learning

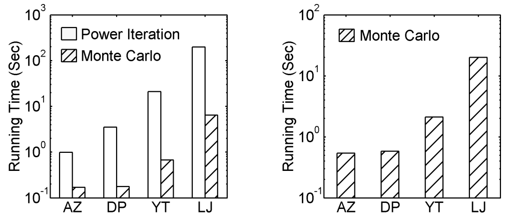
In graph-based semi-supervised learning, a graph is constructed to connect similar data objects [29]. The goal is to predict the unknown class labels using the partially labeled data.

We use the USPS dataset, which contains 9,298 images of handwritten digits from $0\sim 9$ [16, 28]. A weighted k-NN graph is constructed with $k=20$. We use the Gaussian kernel [29] to compute the edge weight $w_{i,j}$ if i is within j's k nearest neighbors or vice versa. We randomly pick 20 nodes as labeled nodes and make sure that there is at least one labeled node for each class. The label of the nearest neighbor is used as the predicted class label for unlabeled nodes. We repeat this process 10^3 times and report the average classification accuracy.

Figure 13 shows the classification accuracy using RR, SR, and SS in the first-order and second-order forms. We can see that the second-order measures outperform their first-order counterparts. The second-order measures take the community structure in the k-NN graph into account thus have better performance than the first-order measures.

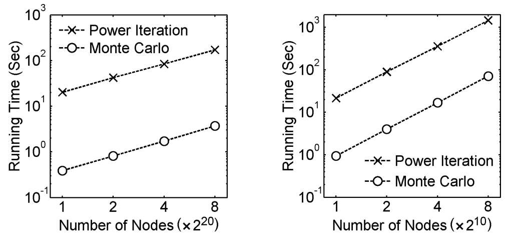
Table 4: Statistics of synthetic networks

large	#nodes	1×2^{20}	2×2^{20}	4×2^{20}	8×2^{20}
	#edges	1×10^{7}	2×10^{7}	4×10^{7}	8×10^{7}
small	#nodes	1×2^{10}	2×2^{10}	4×2^{10}	8×2^{10}
	#edges	1×10^{4}	2×10^{4}	4×10^{4}	8×10^{4}



(a) RR^2 (b) SR^2

Figure 14: Running time on real networks



(a) RR^2 (b) SR^2

Figure 15: Running time on synthetic networks

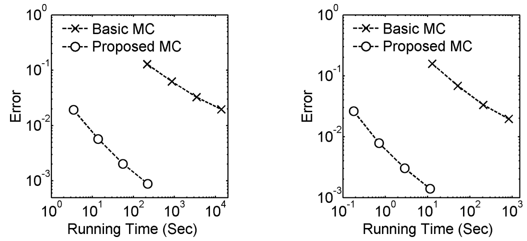
7.3 Efficiency Evaluation

We evaluate the efficiency of the proposed Monte Carlo (MC) methods on real and synthetic networks. Table 3 shows the statistics of real networks. The synthetic graphs are based on the R-MAT model [4]. We use the graph generator available at `https://github.com/dhruvbird/GTgraph` and its default parameters to generate two series of graphs with different sizes. Table 4 shows the statistics. The algorithms for PR and SS have similar performance as those for RR and SR respectively. Thus, we focus on RR and SR.

In the MC methods, we set the parameter $\eta=20$ and $\pi=4n$, where n denotes the number of nodes in the graph. Let $\mathbf{r}$ and $\tilde{\mathbf{r}}$ denote the exact and estimated proximity vectors respectively. The error of the MC method is defined as $\mathsf{Error}=\|\mathbf{r}-\tilde{\mathbf{r}}\|_1/\|\mathbf{r}\|_1$, where $\|\mathbf{r}\|_1=\sum_i |r_i|$ denotes the sum of absolute values.

Figure 14(a) shows the running time of the power iteration method and the developed MC algorithm for RR^2 on real networks. We can see that the MC method is $1\sim 2$ orders of magnitude faster than the power method. Note that in these experiments, the error of the MC method is less than 10^{-2}. Thus with little loss in accuracy, the MC algorithm can dramatically improve the running time. Figure 14(b) shows the running time of the MC method for SR^2. The power method for SR^2 is prohibitive on these large networks. Thus, its running time is omitted. We can see that the MC method can process large graphs within seconds.

Figure 15(a) shows the running time of the algorithms for RR^2 on large synthetic networks. Similarly, the MC method is $1\sim 2$ orders of magnitude faster than the power method.



(a) real network, LiveJournal (b) synthetic network, 2^{20} nodes

Figure 16: Error versus running time (first-order SimRank)

The power method for SR^2 is prohibitive on large networks, thus we report the results on small networks. Note that the proposed MC method is applicable on large networks. Since the power method computes all-pairs proximity, to compare with the power method, we use each node as the query node and call the MC method. In this way, we also compute all-pairs proximity using the MC method. We then report the overall running time. Figure 15(b) shows the running time on synthetic networks. We can see that the MC method is $1\sim2$ orders of magnitude faster than the power method. The error of the MC method is also less than 10^{-2} in all these experiments.

We further compare the sampling strategy in the MC method developed in [7] and our sampling strategy described in Algorithm 5. Recall that the method in [7] samples meeting paths starting from the query node q and every other node i, while our method samples the meeting paths all starting from the query node. We compare the running time that the two methods need to take to achieve the same accuracy. When varying the number of sampled paths, the running time and accuracy of the two methods will change correspondingly. For each setting, we repeat the query 10^3 times with randomly picked query nodes and report the average running time and error.

Figure 16(a) shows the error versus running time on the LiveJournal network. We can see that to achieve the same accuracy, the proposed method is about 3 orders of magnitude faster than the previous method. This demonstrates the advantage of the proposed sampling strategy. Figure 16(b) shows the error versus running time on the synthetic graph with 2^{20} nodes. A similar trend can be observed.

8. CONCLUSIONS

Designing effective proximity measures for large graphs is an important and challenging task. Most existing random walk based measures only use the first-order transition probability. In this paper, we investigate the second-order random walk measures which can capture the cluster structures in the graph and better model real-life applications. We provide rigorous theoretical foundations for the second-order random walk and develop second-order forms for commonly used measures. We further develop effective Monte Carlo methods to compute these measures. Extensive experimental results demonstrate that the second-order measures can effectively improve the accuracy in various applications, and the developed Monte Carlo methods can significantly speed up the computation with little loss in accuracy.

Acknowledgements. This work was partially supported by the National Science Foundation grants IIS-1162374, CAREER, and the NIH grant R01GM115833.

9. REFERENCES

[1] http://www.robwu.net .

[2] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using PageRank vectors. In *FOCS*, pp. 475–486, 2006.

[3] R. E. Bucklin and C. Sismeiro. Click here for internet insight: Advances in clickstream data analysis in marketing. *Journal of Interactive Marketing*, 23(1):35–48, 2009.

[4] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A recursive model for graph mining. In *SDM*, pages 442–446, 2004.

[5] F. Chung and L. Lu. *Complex graphs and networks*, chapter Old and new concentration inequalities. AMS, 2006.

[6] S. Cohen, B. Kimelfeld, and G. Koutrika. A survey on proximity measures for social networks. In *Search Computing*, pages 191–206, 2012.

[7] D. Fogaras and B. Rácz. Scaling link-based similarity search. In *WWW*, pages 641–650, 2005.

[8] D. Fogaras, B. Rácz, K. Csalogány, et al. Towards scaling fully personalized PageRank: Algorithms, lower bounds, and experiments. *Internet Mathematics*, 2(3):333–358, 2005.

[9] D. F. Gleich, L.-H. Lim, and Y. Yu. Multilinear PageRank. *SIAM Journal on Matrix Analysis and Applications*, 2015.

[10] W. Hoeffding. Probability inequalities for sums of bounded random variables. *JASA*, 58(301):13–30, 1963.

[11] G. Jeh and J. Widom. SimRank: A measure of structural-context similarity. In *KDD*, pages 538–543, 2002.

[12] G. Jeh and J. Widom. Scaling personalized web search. In *WWW*, pages 271–279, 2003.

[13] M. Kusumoto, T. Maehara, and K. Kawarabayashi. Scalable similarity search for SimRank. In *SIGMOD*, pp.325-336,2014.

[14] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110, 2008.

[15] A. N. Langville and C. D. Meyer. *Google's PageRank and beyond: The science of search engine rankings*, chapter The mathematics guide. Princeton University Press, 2006.

[16] Y. LeCun, B. E. Boser, et al. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1990.

[17] O. Lehmberg, R. Meusel, and C. Bizer. Graph structure in the web: Aggregated by pay-level domain. In *WebSci*, pages 119–128, 2014.

[18] C. Li, J. Han, G. He, X. Jin, Y. Sun, Y. Yu, and T. Wu. Fast computation of SimRank for static and dynamic information networks. In *EDBT*, pages 465–476, 2010.

[19] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *JASIST*, 58(7):1019–1031, 2007.

[20] S. Lim, S. Ryu, S. Kwon, K. Jung, and J.-G. Lee. LinkSCAN*: Overlapping community detection using the link-space transformation. In *ICDE*, pages 292–303, 2014.

[21] L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A*, 390(6):1150–1170, 2011.

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

[23] A. E. Raftery. A model for high-order Markov chains. *J. of the Royal Statistical Society: Series B*, pages 528–539, 1985.

[24] M. Rosvall, A. V. Esquivel, A. Lancichinetti, et al. Memory in network flows and its effects on spreading dynamics and community detection. *Nature Commun.*, 5(4630), 2014.

[25] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. In *ICDM*, pages 613–622, 2006.

[26] Y. Wu, R. Jin, J. Li, and X. Zhang. Robust local community detection: On free rider effect and its elimination. *PVLDB*, 8(7):798–809, 2015.

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

[28] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.

[29] X. Zhu and A. Goldberg. *Introduction to semi-supervised learning*, chapter Graph-based semi-supervised learning. Morgan & Claypool Publishers, 2009.

APPENDIX

A. DEGENERATION

In this appendix, we show that if the second-order transition probability is the same as the first-order transition probability, i.e., if $p_{i,j,k}=p_{j,k}$, each developed second-order measure degenerates to its original first-order form.

Lemma 4. *If $p_{i,j,k}=p_{j,k}$, we have that $\mathbf{M}=\mathbf{E}\mathbf{H}$.*

Proof. Let $u=(i,j)$ and $v=(j,k)$. There is only one non-zero element in the row vector $[\mathbf{E}]_{u,:}$, i.e., $[\mathbf{E}]_{u,j}=1$. There is only one non-zero element in the column vector $[\mathbf{H}]_{:,v}$, i.e., $[\mathbf{H}]_{j,v}=p_{j,k}$. Thus, we have $p_{j,k}=[\mathbf{E}]_{u,j}\cdot[\mathbf{H}]_{j,v}=[\mathbf{E}]_{u,:}\cdot[\mathbf{H}]_{:,v}$. We also have that $p_{i,j,k}=[\mathbf{M}]_{u,v}$. Since $p_{i,j,k}=p_{j,k}$, we have that $[\mathbf{M}]_{u,v}=[\mathbf{E}]_{u,:}\cdot[\mathbf{H}]_{:,v}$ for any two edges u and v. Thus, we have that $\mathbf{M}=\mathbf{E}\mathbf{H}$. □

Theorem 9. *If $p_{i,j,k}=p_{j,k}$, the second-order random walk degenerates to the first-order random walk.*

Proof. In the first-order random walk, the recursive equation is

$$\mathbf{r}=\mathbf{P}^{\mathsf{T}}\mathbf{r}=\mathbf{E}^{\mathsf{T}}\mathbf{H}^{\mathsf{T}}\mathbf{r}$$

Multiplying $\mathbf{H}^{\mathsf{T}}$ from left to both sides, we have that

$$\mathbf{H}^{\mathsf{T}}\mathbf{r}=\mathbf{H}^{\mathsf{T}}\mathbf{E}^{\mathsf{T}}\mathbf{H}^{\mathsf{T}}\mathbf{r}$$

By Lemma 4, if $p_{i,j,k}=p_{j,k}$, we have that $\mathbf{M}=\mathbf{E}\mathbf{H}$. Let $\mathbf{s}=\mathbf{H}^{\mathsf{T}}\mathbf{r}$. We have that

$$\mathbf{s}=\mathbf{H}^{\mathsf{T}}\mathbf{E}^{\mathsf{T}}\mathbf{s}=\mathbf{M}^{\mathsf{T}}\mathbf{s}\quad\text{and}\quad\mathbf{r}=\mathbf{E}^{\mathsf{T}}\mathbf{H}^{\mathsf{T}}\mathbf{r}=\mathbf{E}^{\mathsf{T}}\mathbf{s}$$

Thus, we get the equations for the second-order random walk. That is, the solution to the first-order random walk is also a solution to the second-order random walk. Since the solutions are unique, we can complete the proof. □

Theorem 10. *If $p_{i,j,k}=p_{j,k}$, the second-order PageRank degenerates to the first-order PageRank.*

Proof. In the first-order PageRank, the recursive equation is

$$\mathbf{r}=c\mathbf{P}^{\mathsf{T}}\mathbf{r}+(1-c)\mathbf{1}/n=c\mathbf{E}^{\mathsf{T}}\mathbf{H}^{\mathsf{T}}\mathbf{r}+(1-c)\mathbf{1}/n$$

Multiplying $\mathbf{H}^{\mathsf{T}}$ from left to both sides, we have that

$$\mathbf{H}^{\mathsf{T}}\mathbf{r}=c\mathbf{H}^{\mathsf{T}}\mathbf{E}^{\mathsf{T}}\mathbf{H}^{\mathsf{T}}\mathbf{r}+(1-c)\mathbf{H}^{\mathsf{T}}\mathbf{1}/n$$

By Lemma 4, if $p_{i,j,k}=p_{j,k}$, we have that $\mathbf{M}=\mathbf{E}\mathbf{H}$. Let $\mathbf{s}=\mathbf{H}^{\mathsf{T}}\mathbf{r}$. We have that

$$\mathbf{s}=c\mathbf{H}^{\mathsf{T}}\mathbf{E}^{\mathsf{T}}\mathbf{s}+(1-c)\mathbf{H}^{\mathsf{T}}\mathbf{1}/n=c\mathbf{M}^{\mathsf{T}}\mathbf{s}+(1-c)\mathbf{H}^{\mathsf{T}}\mathbf{1}/n$$

$$\text{and}\quad\mathbf{r}=c\mathbf{E}^{\mathsf{T}}\mathbf{H}^{\mathsf{T}}\mathbf{r}+(1-c)\mathbf{1}/n=c\mathbf{E}^{\mathsf{T}}\mathbf{s}+(1-c)\mathbf{1}/n$$

Thus, we get the equations for the second-order PageRank. That is, the solution to the first-order PageRank is also a solution to the second-order PageRank. Since the solutions are unique, we can complete the proof. □

Theorem 11. *If $p_{i,j,k}=p_{j,k}$, the second-order random walk with restart degenerates to the first-order random walk with restart.*

Proof. The proof is similar to that of Theorem 10. □

Theorem 12. *If $p_{i,j,k}=p_{j,k}$, the second-order SimRank degenerates to the first-order SimRank.*

Proof. In the first-order SimRank, we have that

$$r_{i,j}=(1-c)\textstyle\sum_{t=0}^{\infty}c^t\mathbb{P}[\Phi_{i,j}^{t,2t}]$$

In the second-order SimRank, we have that

$$r_{i,j}=(1-c)\textstyle\sum_{t=0}^{\infty}c^t\mathbb{M}[\Phi_{i,j}^{t,2t}]$$

By Lemma 3, if $p_{i,j,k}=p_{j,k}$, we have that $\mathbb{M}[\Phi_{i,j}^{t,2t}]=\mathbb{P}[\Phi_{i,j}^{t,2t}]$. This completes the proof. □

Theorem 13. *If $p_{i,j,k}=p_{j,k}$, the second-order SimRank* degenerates to the first-order SimRank*.*

Proof. In the first-order SimRank*, we have that

$$r_{i,j}=(1-c)\textstyle\sum_{t=0}^{\infty}\frac{c^t}{2^t}\sum_{a=0}^{t}\binom{t}{a}\mathbb{P}[\Phi_{i,j}^{a,t}]$$

In the second-order SimRank*, we have that

$$r_{i,j}=(1-c)\textstyle\sum_{t=0}^{\infty}\frac{c^t}{2^t}\sum_{a=0}^{t}\binom{t}{a}\mathbb{M}[\Phi_{i,j}^{a,t}]$$

By Lemma 3, if $p_{i,j,k}=p_{j,k}$, we have that $\mathbb{M}[\Phi_{i,j}^{a,t}]=\mathbb{P}[\Phi_{i,j}^{a,t}]$. This completes the proof. □

B. VISITING PROBABILITY

The proof of Lemma 2 is as follows.

Proof. We prove each of the four cases individually including $0=a=b$, $0<a=b$, $0=a<b$, and $0<a<b$.

In the first case, the lemma trivially holds. The probability of visiting a meeting path of length $\{0,0\}$ between nodes i and j is 1 if $i=j$ and 0 if $i\neq j$, i.e., $\mathbb{M}[\Phi_{i,j}^{0,0}]=\mathbf{I}_{i,j}$.

In the second case, we have that $0<a=b$. We proceed by induction on a. If $a=1$, we have that

$$[\mathbf{H}\mathbf{M}^{a-1}\mathbf{E}]_{x,j}=[\mathbf{H}\mathbf{E}]_{x,j}=[\mathbf{P}]_{x,j}=p_{x,j}$$

Since there is a unique path of length 1 from node x to j, which is the directed edge (x,j), we have that $\mathbb{M}[\Phi_{x,j}^{1,1}]=p_{x,j}$. Therefore, we have $\mathbb{M}[\Phi_{x,j}^{1,1}]=[\mathbf{H}\mathbf{E}]_{x,j}$ thus the lemma holds for $a=1$. Now assume that the lemma holds for $a\,(a\geq 1)$. By the assumption, we have

$$\begin{aligned}\mathbb{M}[\Phi_{x,j}^{a,a}]=[\mathbf{H}\mathbf{M}^{a-1}\mathbf{E}]_{x,j}&=\textstyle\sum_{i\in I_j}[\mathbf{H}\mathbf{M}^{a-1}]_{x,(i,j)}\cdot[\mathbf{E}]_{(i,j),j}\\&=\textstyle\sum_{i\in I_j}[\mathbf{H}\mathbf{M}^{a-1}]_{x,(i,j)}\end{aligned}$$

Each term $[\mathbf{H}\mathbf{M}^{a-1}]_{x,(i,j)}$ represents the sum of probabilities of visiting the paths of length a from node x to j whose last edge is (i,j). Next, we prove that the lemma holds for $(a+1)$. Each term $[\mathbf{H}\mathbf{M}^{a}\mathbf{E}]_{x,k}$ can be expanded as

$$\begin{aligned}[\mathbf{H}\mathbf{M}^{a}\mathbf{E}]_{x,k}&=\textstyle\sum_{j\in I_k}[\mathbf{H}\mathbf{M}^{a}]_{x,(j,k)}\cdot[\mathbf{E}]_{(j,k),k}\\&=\textstyle\sum_{j\in I_k}[\mathbf{H}\mathbf{M}^{a}]_{x,(j,k)}\\&=\textstyle\sum_{j\in I_k}\sum_{i\in I_j}[\mathbf{H}\mathbf{M}^{a-1}]_{x,(i,j)}\cdot[\mathbf{M}]_{(i,j),(j,k)}\\&=\textstyle\sum_{j\in I_k}\sum_{i\in I_j}[\mathbf{H}\mathbf{M}^{a-1}]_{x,(i,j)}\cdot p_{i,j,k}\end{aligned}$$

Consider a path ρ of length $(a+1)$ from node x to k whose last two edges are (i,j) and (j,k). The path ρ consists of a path ρ' of length a from x to j whose last edge is (i,j), followed by the edge (j,k). The probability of visiting ρ equals the probability of visiting path ρ' times the transition probability $p_{i,j,k}$. It follows that $[\mathbf{H}\mathbf{M}^{a-1}]_{x,(i,j)}\cdot p_{i,j,k}$ equals the sum of probabilities of visiting the paths of length $(a+1)$ from node x to k whose last two edges are (i,j) and (j,k). Thus, $\sum_{j\in I_k}\sum_{i\in I_j}[\mathbf{H}\mathbf{M}^{a-1}]_{x,(i,j)}\cdot p_{i,j,k}$ is the sum of probabilities of visiting all paths of length $(a+1)$ from x to k. Therefore, we have that $\mathbb{M}[\Phi_{x,k}^{a+1,a+1}]=[\mathbf{H}\mathbf{M}^{a}\mathbf{E}]_{x,k}$. This completes the proof for the second case.

In the third case, we have that $0=a<b$. We proceed by induction on b. If $b=1$, we have that

$$[\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b-1}\mathbf{H}^{\mathsf{T}}]_{j,y}=[\mathbf{E}^{\mathsf{T}}\mathbf{H}^{\mathsf{T}}]_{j,y}=[\mathbf{P}^{\mathsf{T}}]_{j,y}=p_{y,j}$$

Since there is a unique path of length 1 from node y to j, which is the directed edge (y,j), we have that $\mathbb{M}[\Phi^{0,1}_{j,y}]=p_{y,j}$. Therefore, we have $\mathbb{M}[\Phi^{0,1}_{j,y}]=[\mathbf{E}^{\mathsf{T}}\mathbf{H}^{\mathsf{T}}]_{j,y}$ thus the lemma holds for $b=1$. Now assume that the lemma holds for $b\,(b\geq 1)$. By the assumption, we have

$$\begin{aligned}\mathbb{M}[\Phi^{0,b}_{j,y}]=[\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b-1}\mathbf{H}^{\mathsf{T}}]_{j,y}&=\textstyle\sum_{i\in I_j}[\mathbf{E}^{\mathsf{T}}]_{j,(i,j)}\cdot[(\mathbf{M}^{\mathsf{T}})^{b-1}\mathbf{H}^{\mathsf{T}}]_{(i,j),y}\\&=\textstyle\sum_{i\in I_j}[(\mathbf{M}^{\mathsf{T}})^{b-1}\mathbf{H}^{\mathsf{T}}]_{(i,j),y}\end{aligned}$$

Each term $[(\mathbf{M}^{\mathsf{T}})^{b-1}\mathbf{H}^{\mathsf{T}}]_{(i,j),y}$ represents the sum of probabilities of visiting the paths of length b from node y to j whose last edge is (i,j). Next, we prove that the lemma holds for $(b+1)$. Each term $[\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b}\mathbf{H}^{\mathsf{T}}]_{k,y}$ can be expanded as

$$\begin{aligned}[\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b}\mathbf{H}^{\mathsf{T}}]_{k,y}&=\textstyle\sum_{j\in I_k}[\mathbf{E}^{\mathsf{T}}]_{k,(j,k)}\cdot[(\mathbf{M}^{\mathsf{T}})^{b}\mathbf{H}^{\mathsf{T}}]_{(j,k),y}\\&=\textstyle\sum_{j\in I_k}[(\mathbf{M}^{\mathsf{T}})^{b}\mathbf{H}^{\mathsf{T}}]_{(j,k),y}\\&=\textstyle\sum_{j\in I_k}\sum_{i\in I_j}[\mathbf{M}^{\mathsf{T}}]_{(j,k),(i,j)}\cdot[(\mathbf{M}^{\mathsf{T}})^{b-1}\mathbf{H}^{\mathsf{T}}]_{(i,j),y}\\&=\textstyle\sum_{j\in I_k}\sum_{i\in I_j}p_{i,j,k}\cdot[(\mathbf{M}^{\mathsf{T}})^{b-1}\mathbf{H}^{\mathsf{T}}]_{(i,j),y}\end{aligned}$$

Consider a path ρ of length $(b+1)$ from node y to k whose last two edges are (i,j) and (j,k). The path ρ consists of a path ρ' of length b from y to j whose last edge is (i,j), followed by the edge (j,k). The probability of visiting ρ equals the probability of visiting path ρ' times the transition probability $p_{i,j,k}$. It follows that $p_{i,j,k}\cdot[(\mathbf{M}^{\mathsf{T}})^{b-1}\mathbf{H}^{\mathsf{T}}]_{(i,j),y}$ equals the sum of probabilities of visiting the paths of length $(b+1)$ from node y to k whose last two edges are (i,j) and (j,k). Thus, $\sum_{j\in I_k}\sum_{i\in I_j}p_{i,j,k}\cdot[(\mathbf{M}^{\mathsf{T}})^{b-1}\mathbf{H}^{\mathsf{T}}]_{(i,j),y}$ is the sum of probabilities of visiting all paths of length $(b+1)$ from y to k. Therefore, we have that $\mathbb{M}[\Phi^{0,b+1}_{k,y}]=[\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b}\mathbf{H}^{\mathsf{T}}]_{k,y}$. This completes the proof for the third case.

In the fourth case, we have that $0<a<b$. We proceed by induction on both a and b. If $a=1$ and $b=2$, we have that

$$\begin{aligned}&[\mathbf{H}\mathbf{M}^{a-1}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b-a-1}\mathbf{H}^{\mathsf{T}}]_{x,y}\\&=[\mathbf{H}\mathbf{E}\mathbf{E}^{\mathsf{T}}\mathbf{H}^{\mathsf{T}}]_{x,y}=[\mathbf{P}\mathbf{P}^{\mathsf{T}}]_{x,y}=\textstyle\sum_{i\in V}p_{x,i}\cdot p_{y,i}\end{aligned}$$

Each term $(p_{x,i}\cdot p_{y,i})$ represents the probability of visiting the meeting path $x\rightarrow i\leftarrow y$. Thus, $\sum_{i\in V}p_{x,i}\cdot p_{y,i}$ represents the sum of probabilities of visiting the paths in $\Phi^{1,2}_{x,y}$. Therefore, we have that $\mathbb{M}[\Phi^{1,2}_{x,y}]=[\mathbf{H}\mathbf{E}\mathbf{E}^{\mathsf{T}}\mathbf{H}^{\mathsf{T}}]_{x,y}$ thus the lemma holds for $\{a=1,b=2\}$. Now assume that the lemma holds for $\{a,b\}\,(0<a<b)$. We will prove that the lemma holds for both $\{a+1,b+1\}$ and $\{a,b+1\}$. By the assumption, we have

$$\begin{aligned}\mathbb{M}[\Phi^{a,b}_{x,y}]&=[\mathbf{H}\mathbf{M}^{a-1}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b-a-1}\mathbf{H}^{\mathsf{T}}]_{x,y}\\&=\textstyle\sum_{j\in V}[\mathbf{H}\mathbf{M}^{a-1}\mathbf{E}]_{x,j}\cdot[\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b-a-1}\mathbf{H}^{\mathsf{T}}]_{j,y}\end{aligned}$$

We first prove that the lemma holds for $\{a+1,b+1\}$. As discussed in the second case, each term $[\mathbf{H}\mathbf{M}^{a-1}\mathbf{E}]_{x,j}=\mathbb{M}[\Phi^{a,a}_{x,j}]$ represents the sum of probabilities of visiting the paths of length a from node x to j. Following the same discussion in the second case, we can prove that $[\mathbf{H}\mathbf{M}^{a}\mathbf{E}]_{x,j}=\mathbb{M}[\Phi^{a+1,a+1}_{x,j}]$ represents the sum of probabilities of visiting the paths of length $(a+1)$ from node x to j. Thus, we have that

$$\begin{aligned}&\textstyle\sum_{j\in V}[\mathbf{H}\mathbf{M}^{a}\mathbf{E}]_{x,j}\cdot[\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b-a-1}\mathbf{H}^{\mathsf{T}}]_{j,y}\\&=[\mathbf{H}\mathbf{M}^{a}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b-a-1}\mathbf{H}^{\mathsf{T}}]_{x,y}=\mathbb{M}[\Phi^{a+1,b+1}_{x,y}]\end{aligned}$$

represents the sum of probabilities of visiting the meeting paths of length $\{a+1,b+1\}$ between nodes x and y. Thus, the lemma holds for $\{a+1,b+1\}$.

We then prove that the lemma holds for $\{a,b+1\}$. As discussed in the third case, each term $[\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b-a-1}\mathbf{H}^{\mathsf{T}}]_{j,y}=\mathbb{M}[\Phi^{0,b-a}_{j,y}]$ represents the sum of probabilities of visiting the paths of length $(b-a)$ from node y to j. Following the same discussion in the third case, we can prove that $[\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b-a}\mathbf{H}^{\mathsf{T}}]_{j,y}=\mathbb{M}[\Phi^{0,b-a+1}_{j,y}]$ represents the sum of probabilities of visiting the paths of length $(b-a+1)$ from node y to j. Thus, we have that

$$\begin{aligned}&\textstyle\sum_{j\in V}[\mathbf{H}\mathbf{M}^{a-1}\mathbf{E}]_{x,j}\cdot[\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b-a}\mathbf{H}^{\mathsf{T}}]_{j,y}\\&=[\mathbf{H}\mathbf{M}^{a-1}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b-a}\mathbf{H}^{\mathsf{T}}]_{x,y}=\mathbb{M}[\Phi^{a,b+1}_{x,y}]\end{aligned}$$

represents the sum of probabilities of visiting the meeting paths of length $\{a,b+1\}$ between nodes x and y. Thus, the lemma holds for $\{a,b+1\}$. This completes the proof for the fourth case. □

The proof of Lemma 3 is as follows.

Proof. If $0=a=b$, the lemma trivially holds, i.e., $\mathbb{M}[\Phi^{0,0}_{i,j}]=\mathbb{P}[\Phi^{0,0}_{i,j}]=\mathbf{I}_{i,j}$. By Lemma 4, if $p_{i,j,k}=p_{j,k}$, we have that $\mathbf{M}=\mathbf{E}\mathbf{H}$. If $0<a=b$, we have that

$$\begin{aligned}\mathbb{M}[\Phi^{a,a}_{i,j}]&=[\mathbf{H}\mathbf{M}^{a-1}\mathbf{E}]_{i,j}=[\mathbf{H}(\mathbf{E}\mathbf{H})^{a-1}\mathbf{E}]_{i,j}\\&=[(\mathbf{H}\mathbf{E})^{a}]_{i,j}=[\mathbf{P}^{a}]_{i,j}=\mathbb{P}[\Phi^{a,a}_{i,j}]\end{aligned}$$

If $0=a<b$, we have that

$$\begin{aligned}\mathbb{M}[\Phi^{0,b}_{i,j}]&=[\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b-1}\mathbf{H}^{\mathsf{T}}]_{i,j}=[\mathbf{E}^{\mathsf{T}}(\mathbf{H}^{\mathsf{T}}\mathbf{E}^{\mathsf{T}})^{b-1}\mathbf{H}^{\mathsf{T}}]_{i,j}\\&=[(\mathbf{E}^{\mathsf{T}}\mathbf{H}^{\mathsf{T}})^{b}]_{i,j}=[(\mathbf{P}^{\mathsf{T}})^{b}]_{i,j}=\mathbb{P}[\Phi^{0,b}_{i,j}]\end{aligned}$$

If $0<a<b$, we have that

$$\begin{aligned}\mathbb{M}[\Phi^{a,b}_{i,j}]&=[\mathbf{H}\mathbf{M}^{a-1}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{b-a-1}\mathbf{H}^{\mathsf{T}}]_{i,j}\\&=[\mathbf{H}(\mathbf{E}\mathbf{H})^{a-1}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{H}^{\mathsf{T}}\mathbf{E}^{\mathsf{T}})^{b-a-1}\mathbf{H}^{\mathsf{T}}]_{i,j}\\&=[(\mathbf{H}\mathbf{E})^{a}(\mathbf{E}^{\mathsf{T}}\mathbf{H}^{\mathsf{T}})^{b-a}]_{i,j}=[\mathbf{P}^{a}(\mathbf{P}^{\mathsf{T}})^{b-a}]_{i,j}=\mathbb{P}[\Phi^{a,b}_{i,j}]\end{aligned}$$

This completes the proof. □

C. THE SECOND-ORDER SIMRANK

The proof of Theorem 2 is as follows.

Proof. The second-order SimRank is defined as

$$r_{i,j}=(1-c)\textstyle\sum_{t=0}^{\infty}c^{t}\mathbb{M}[\Phi^{t,2t}_{i,j}]$$

By Lemma 2, we have that $\mathbb{M}[\Phi^{0,0}_{i,j}]=\mathbf{I}_{i,j}$ and $\mathbb{M}[\Phi^{t,2t}_{i,j}]=[\mathbf{H}\mathbf{M}^{t-1}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{t-1}\mathbf{H}^{\mathsf{T}}]_{i,j}$ if $t>0$. Thus, the node proximity matrix $\mathbf{R}$ can be expressed as

$$\begin{aligned}\mathbf{R}&=(1-c)\textstyle\sum_{t=1}^{\infty}c^{t}\mathbf{H}\mathbf{M}^{t-1}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{t-1}\mathbf{H}^{\mathsf{T}}+(1-c)\mathbf{I}\\&=c\mathbf{H}\big((1-c)\textstyle\sum_{t=1}^{\infty}c^{t-1}\mathbf{M}^{t-1}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{t-1}\big)\mathbf{H}^{\mathsf{T}}+(1-c)\mathbf{I}\\&=c\mathbf{H}\big((1-c)\textstyle\sum_{t=0}^{\infty}c^{t}\mathbf{M}^{t}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{t}\big)\mathbf{H}^{\mathsf{T}}+(1-c)\mathbf{I}\end{aligned}$$

Let $\mathbf{S}=(1-c)\sum_{t=0}^{\infty}c^{t}\mathbf{M}^{t}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{t}$. Thus, we have that $\mathbf{R}=c\mathbf{H}\mathbf{S}\mathbf{H}^{\mathsf{T}}+(1-c)\mathbf{I}$. Matrix $\mathbf{S}$ can be written as

$$\begin{aligned}&c\mathbf{M}\mathbf{S}\mathbf{M}^{\mathsf{T}}+(1-c)\mathbf{E}\mathbf{E}^{\mathsf{T}}\\&=(1-c)\textstyle\sum_{t=0}^{\infty}c^{t+1}\mathbf{M}^{t+1}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{t+1}+(1-c)\mathbf{E}\mathbf{E}^{\mathsf{T}}\\&=(1-c)\textstyle\sum_{t=1}^{\infty}c^{t}\mathbf{M}^{t}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{t}+(1-c)\mathbf{E}\mathbf{E}^{\mathsf{T}}\\&=(1-c)\textstyle\sum_{t=0}^{\infty}c^{t}\mathbf{M}^{t}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{t}\;=\;\mathbf{S}\end{aligned}$$

□

Lemma 5 is needed in the proof of Theorem 3.

Lemma 5. *The gap between* $\mathbf{S}$ *and* $\mathbf{S}^{(\eta)}$ *is bounded by* $\big\|\mathbf{S}-\mathbf{S}^{(\eta)}\big\|_{\max}\leq c^{\eta+1}$ *for any* $\eta\,(\eta\geq 0)$.

Proof. For each $\eta=0,1,\cdots$, we subtract $\mathbf{S}^{(\eta)}$ from $\mathbf{S}$, and then take $\|\cdot\|_{\max}$ norms on both sides to get

Table 5: The sample space when the length a is given

sample space		$\mathbb{S}_i$
bSuccess	node z_a	
successfully sample a path of length a starting from node q (bSuccess = true)	$z_a = i$	1
	$z_a \neq i$	0
fail to sample a path of length a starting from node q (bSuccess = false)	–	0

$$\left\|\mathbf{S}-\mathbf{S}^{(\eta)}\right\|_{\max} \leq (1-c)\sum_{t=\eta+1}^{\infty} c^t \left\|\mathbf{M}^t\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^t\right\|_{\max}$$

Note that matrix $\mathbf{E}\mathbf{E}^{\mathsf{T}}$ is binary. Each element $[\mathbf{E}\mathbf{E}^{\mathsf{T}}]_{u,v}=1$ if edge u and v end at the same node and $[\mathbf{E}\mathbf{E}^{\mathsf{T}}]_{u,v}=0$ otherwise. Thus, we have that $\left\|\mathbf{M}^t\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^t\right\|_{\max} \leq 1$. Plugging this into the above inequality, we have that

$$\left\|\mathbf{S}-\mathbf{S}^{(\eta)}\right\|_{\max} \leq (1-c)\sum_{t=\eta+1}^{\infty} c^t = c^{\eta+1} \qquad \square$$

D. MONTE CARLO METHODS

Lemma 6 is needed in the proof of Theorem 4.

Lemma 6. $\mathbb{E}[\mathbb{S}_i] = r_i$

Proof. The set of all possible outcomes of the sampling process is called the sample space, which contains the following events.

1) The algorithm generates a random number a and successfully samples a path of length a starting from the query node q.
2) The algorithm generates a random number a but fails to sample a path of length a starting from the query node q because some node has no out-neighbors.

Note that this sample space is different from the sample space of the random variable $\mathbb{S}_i$, which is the set of two integers $\{0,1\}$.

The whole sample space can be partitioned based on the length a. By the law of total expectation, the expectation of $\mathbb{S}_i$ can be written as

$$\mathbb{E}[\mathbb{S}_i] = \sum_{a=0}^{\infty} \mathbb{P}[\mathbb{A}=a] \cdot \mathbb{E}[\mathbb{S}_i|\mathbb{A}=a], \qquad (2)$$

where the random variable $\mathbb{A}$ representing the length a follows the geometric distribution $\mathbb{P}[\mathbb{A}=a]=(1-c)\cdot c^a$. Next, we consider the conditional expectation $\mathbb{E}[\mathbb{S}_i|\mathbb{A}=a]$ of $\mathbb{S}_i$ given the event $\mathbb{A}=a$.

Table 5 shows the sample space given the length a. Given the length a, if the algorithm successfully samples a path of length a from node q to i, the random variable $\mathbb{S}_i=1$; otherwise, $\mathbb{S}_i=0$. Let $\underline{\mathbb{M}}[\rho]$ represent the probability of successfully sampling a path ρ given the length a. Since $\mathbb{S}_i=1$ if and only if the algorithm successfully samples a path ρ of length a from node q to i, the conditional expectation $\mathbb{E}[\mathbb{S}_i|\mathbb{A}=a]$ can be written as

$$\mathbb{E}[\mathbb{S}_i|\mathbb{A}=a] = \sum_{\rho\in\Phi_{q,i}^{a,a}} 1\cdot\underline{\mathbb{M}}[\rho] = \sum_{\rho\in\Phi_{q,i}^{a,a}} \underline{\mathbb{M}}[\rho],$$

where $\Phi_{q,i}^{a,a}$ denotes the set of all paths of length a from node q to i.

Given the length a, the probability of successfully sampling a path $\rho: q=z_0\rightarrow\cdots\rightarrow z_a$ is $\underline{\mathbb{M}}[\rho]=p_{z_0,z_1}\prod_{t=1}^{a-1}p_{z_{t-1},z_t,z_{t+1}}$. We can see that the probabilities of sampling and visiting a path ρ are equal, i.e., $\underline{\mathbb{M}}[\rho]=\mathbb{M}[\rho]$. Thus, we have that

$$\mathbb{E}[\mathbb{S}_i|\mathbb{A}=a] = \sum_{\rho\in\Phi_{q,i}^{a,a}} \mathbb{M}[\rho] = \mathbb{M}[\Phi_{q,i}^{a,a}]$$

Table 6: The sample space when the length a is given

a	sample space		$\mathbb{R}_i$
	bSuccess	node z_{2a}	
$0\leq a\leq\eta$	successfully sample a path of length a starting from node q (bSuccess = true)	$z_{2a}=i$	1
		$z_{2a}\neq i$	0
	fail to sample a path of length a starting from node q (bSuccess = false)	–	0
$\eta<a$	–	–	0

Plugging this into Equation (2), we have that

$$\mathbb{E}[\mathbb{S}_i] = (1-c)\sum_{a=0}^{\infty} c^a \mathbb{M}[\Phi_{q,i}^{a,a}] = r_i \qquad \square$$

Lemma 7 and Theorem 14 are needed in the proofs of Theorems 7 and 8.

Lemma 7. $\mathbb{E}[\mathbb{R}_i]=\hat{r}_i$ *and* $\mathbb{E}[\mathbb{R}_i^2]\leq nr_i$

Proof. The set of all possible outcomes of the sampling process is called the sample space, which contains the following events.

1) The algorithm generates a random number $a\,(0\leq a\leq\eta)$ and successfully samples a meeting path of length $\{a,2a\}$ starting from the query node q.
2) The algorithm generates a random number $a\,(0\leq a\leq\eta)$ but fails to sample a meeting path of length $\{a,2a\}$ starting from the query node q because some node has no out-neighbors or in-neighbors.
3) The algorithm generates a random number $a\,(\eta<a)$ and does nothing.

Note that this sample space is different from the sample space of the random variable $\mathbb{R}_i$, which is the set of real values $\{\delta\}$.

The whole sample space can be partitioned based on the length a. By the law of total expectation, the expectation of $\mathbb{R}_i$ can be written as

$$\mathbb{E}[\mathbb{R}_i] = \sum_{a=0}^{\infty} \mathbb{P}[\mathbb{A}=a]\cdot\mathbb{E}[\mathbb{R}_i|\mathbb{A}=a], \qquad (3)$$

where the random variable $\mathbb{A}$ representing the length a follows the geometric distribution $\mathbb{P}[\mathbb{A}=a]=(1-c)\cdot c^a$. Next, we consider the conditional expectation $\mathbb{E}[\mathbb{R}_i|\mathbb{A}=a]$ of $\mathbb{R}_i$ given the event $\mathbb{A}=a$.

Table 6 shows the sample space given the length a. Given the length $a\,(0\leq a\leq\eta)$, if the algorithm successfully samples a meeting path of length $\{a,2a\}$ between nodes q and i, the random variable $\mathbb{R}_i=\delta$; otherwise, $\mathbb{R}_i=0$. Note that $\delta=[\mathbf{X}]_{z_a,a}/[\mathbf{X}]_{z_{2a},0}$ changes for different sampled meeting paths. Let $\underline{\mathbb{P}}[\phi]$ represent the probability of successfully sampling a meeting path ϕ given the length a. Since $\mathbb{R}_i=\delta$ if and only if the algorithm successfully samples a meeting path ϕ of length $\{a,2a\}$ between nodes q and i, the conditional expectation $\mathbb{E}[\mathbb{R}_i|\mathbb{A}=a]$ can be written as

$$\mathbb{E}[\mathbb{R}_i|\mathbb{A}=a] = \sum_{\phi\in\Phi_{q,i}^{a,2a}} \delta\cdot\underline{\mathbb{P}}[\phi],$$

where $\Phi_{q,i}^{a,2a}$ denotes the set of all meeting paths of length $\{a,2a\}$ between nodes q and i.

Consider the probability of successfully sampling a meeting path $\phi: q=z_0\rightarrow\cdots\rightarrow z_a\leftarrow\cdots\leftarrow z_{2a}$ given the length a. The probability of sampling the first half is $\underline{\mathbb{P}}[\rho_1]=\prod_{t=1}^{a}p_{z_{t-1},z_t}$. The probability of sampling the second half is

$$\underline{\mathbb{P}}[\rho_2] = \prod_{t=a+1}^{2a}\left(p_{z_t,z_{t-1}}\cdot\frac{[\mathbf{X}]_{z_t,2a-t}}{[\mathbf{X}]_{z_{t-1},2a-t+1}}\right) = \frac{[\mathbf{X}]_{z_{2a},0}}{[\mathbf{X}]_{z_a,a}}\cdot\prod_{t=a+1}^{2a}p_{z_t,z_{t-1}}$$

The probability of sampling the meeting path ϕ then is $\underline{\mathbb{P}}[\phi]=\underline{\mathbb{P}}[\rho_1]\cdot\underline{\mathbb{P}}[\rho_2]$. Note that $\delta=[\mathbf{X}]_{z_a,a}/[\mathbf{X}]_{z_{2a},0}$. We can see that the probabilities of sampling and visiting a meeting path ϕ have a relationship, i.e., $\delta\cdot\underline{\mathbb{P}}[\phi]=\mathbb{P}[\phi]$. Thus, if $0\leq a\leq\eta$, we have

$$\mathbb{E}[\mathbb{R}_i|\mathbb{A}=a]=\sum_{\phi\in\Phi_{q,i}^{a,2a}}\mathbb{P}[\phi]=\mathbb{P}[\Phi_{q,i}^{a,2a}]$$

If $\eta<a$, we have $\mathbb{E}[\mathbb{R}_i|\mathbb{A}=a]=0$. Plugging this into Equation (3), we have that

$$\mathbb{E}[\mathbb{R}_i]=(1-c)\sum_{a=0}^{\eta}c^a\mathbb{P}[\Phi_{q,i}^{a,2a}]=\hat{r}_i\ ,$$

where $\hat{r}_i$ is the truncated SimRank proximity.

Next, we prove that $\mathbb{E}[\mathbb{R}_i^2]\leq nr_i$. By the law of total expectation, we have that

$$\mathbb{E}[\mathbb{R}_i^2]=\sum_{a=0}^{\infty}\mathbb{P}[\mathbb{A}=a]\cdot\mathbb{E}[\mathbb{R}_i^2|\mathbb{A}=a] \tag{4}$$

Since $[\mathbf{X}]_{z_a,a}\in[0,1]$ and $[\mathbf{X}]_{z_{2a},0}=\frac{1}{n}$, where n is the number of nodes in the graph, we have that $\delta=[\mathbf{X}]_{z_a,a}/[\mathbf{X}]_{z_{2a},0}\leq n$. Thus, if $0\leq a\leq\eta$, we have

$$\mathbb{E}[\mathbb{R}_i^2|\mathbb{A}=a]=\sum_{\phi\in\Phi_{q,i}^{a,2a}}\delta^2\,\underline{\mathbb{P}}[\phi]=\sum_{\phi\in\Phi_{q,i}^{a,2a}}\delta\,\mathbb{P}[\phi]\leq n\mathbb{P}[\Phi_{q,i}^{a,2a}]$$

If $\eta<a$, we have $\mathbb{E}[\mathbb{R}_i^2|\mathbb{A}=a]=0$. Plugging this into Equation (4), we have that

$$\mathbb{E}[\mathbb{R}_i^2]\leq(1-c)\sum_{a=0}^{\infty}c^a n\mathbb{P}[\Phi_{q,i}^{a,2a}]=nr_i \qquad \square$$

Theorem 14 is based on Theorems 2.8 and 2.9 in [5].

THEOREM 14. [Concentration Inequality] *Let $\mathbb{U}_1,\cdots,\mathbb{U}_\pi$ be independent random variables bounded by interval $[-\alpha,\beta]$, where α and β are non-negative constants, i.e., $-\alpha\leq\mathbb{U}_d\leq\beta$ for each $d=1,\cdots,\pi$. Let $\mathbb{V}=\frac{1}{\pi}\sum_{d=1}^{\pi}\mathbb{U}_d$ and $\theta=\sum_{d=1}^{\pi}\mathbb{E}[\mathbb{U}_d^2]$. For any $\epsilon>0$, we have that*

$$\begin{cases}\mathbb{P}[\mathbb{V}-\mathbb{E}[\mathbb{V}]\leq-\epsilon]\leq\exp\left(\dfrac{-\pi^2\epsilon^2}{2\theta+2\pi\alpha\epsilon/3}\right)\\[2ex]\mathbb{P}[\mathbb{V}-\mathbb{E}[\mathbb{V}]\geq\ \epsilon\ ]\leq\exp\left(\dfrac{-\pi^2\epsilon^2}{2\theta+2\pi\beta\epsilon/3}\right)\end{cases}$$