Fast De-anonymization of Social Networks with Structural Information

Yingxia Shao[1] · Jialin Liu[2] · Shuyang Shi[2] · Yuemei Zhang[2] · Bin Cui[2]

Received: 14 August 2018 / Revised: 11 March 2019 / Accepted: 16 March 2019 / Published online: 27 March 2019

Abstract
Ever since the social networks became the focus of a great number of researches, the privacy risks of published network data have also raised considerable concerns. To evaluate users' privacy risks, researchers have developed methods to de-anonymize the networks and identify the same person in the different networks. However, the existing solutions either require high-quality seed mappings for cold start, or exhibit low accuracy without fully exploiting the structural information, and entail high computation expense. In this paper, we propose a fast and effective seedless network de-anonymization approach simply relying on structural information, named RoleMatch. RoleMatch equips with a new pairwise node similarity measure and an efficient node matching algorithm. Through testing RoleMatch with both real and synthesized social networks, which are anonymized by several popular anonymization algorithms, we demonstrate that the RoleMatch receives superior performance compared with existing de-anonymization algorithms.

Keywords Social network · De-anonymization · Privacy risk · Node similarity

1 Introduction

Online social networks have been a popular and important topic for many years, and a lot of successful companies (e.g., Facebook, Twitter and Tencent) emerge for providing the social network service. Users in such social networks are represented as nodes with multiple attributes, including name, gender, interests, location, etc., and the interactivities between users are abstracted as unidirectional or bidirectional edges between the user nodes. In other words, online social networks can be modeled as a network (or graph) with the necessary information of user relations.

Considering the fact that social network truly represents the relationships in human society, it has drawn a lot of attentions from researchers and advertisers. In order to satisfy the need of analysis, social network companies provide services for sharing the information of network. Nevertheless, user privacy can possibly be breached in the process of sharing more and more information for analysis. A popular problem is identity disclosure, where the real identities of nodes in the social networks are revealed [4, 14]. Therefore, the network to be published has to go through the anonymization processes. And many anonymization approaches have been developed, such as edge sparsification and edge perturbation [3]. The sparsification approach deletes edges in the network randomly, and the perturbation approach randomly deletes and adds back the same number of edges.

In order to find the weakness of anonymization approaches, it is rather crucial to explore the inverse process of anonymization, called de-anonymization. Let us explain the de-anonymization problem through an example. Figure 1a shows a complete social network, which is maintained by a company. Figure 1c shows an anonymized network with location information, and it is published by the company. Attackers usually use crawler to get a crawled network for de-anonymization. Here Fig. 1b shows an example

✉ Yingxia Shao
shaoyx@bupt.edu.cn

Jialin Liu
russellspurs@gmail.com

Shuyang Shi
shuyang790@gmail.com

Yuemei Zhang
zhangyuemei@pku.edu.cn

Bin Cui
bin.cui@pku.edu.cn

[1] School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China

[2] Key Lab of High Confidence Software Technologies (MOE), School of Electronics Engineering and Computer Science, Peking University, Beijing, China

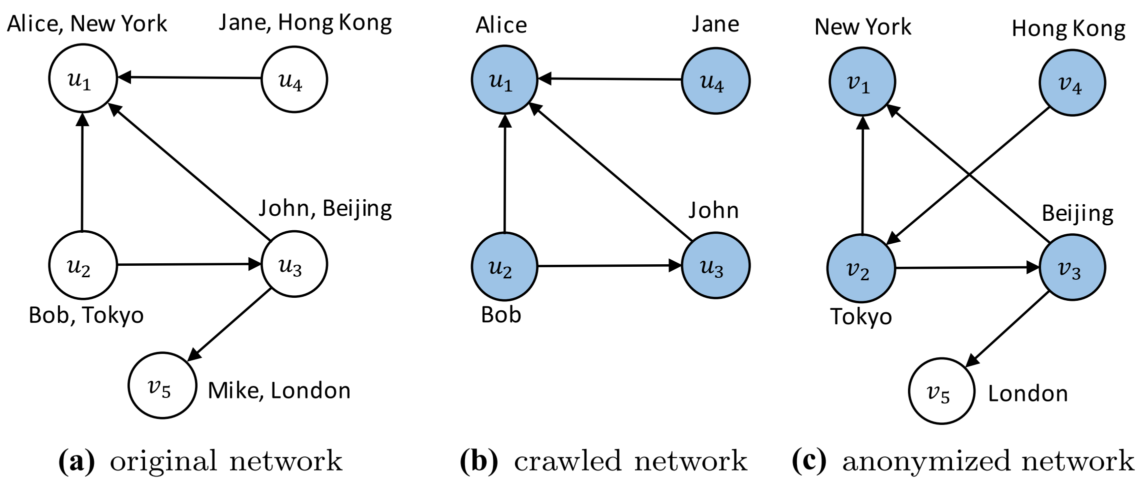


Fig. 1 Three small networks. The crawled network is obtained from the original network, and it has public information (name). The anonymized network is published with the disclosed information (locations). In addition, the blue vertices are the overlap between the crawled network and anonymized network

with name information. The de-anonymization process is to match nodes between Fig. 1b, c to identify each person in the anonymized network and obtain the corresponding location information. Specifically, the de-anonymization is to find node mapping pairs (u_1, v_1), (u_2, v_2), (u_3, v_3), (u_4, v_4).

The existing solutions to the de-anonymization problem can be classified into two major types. The first one is to de-anonymize the network based on seed mappings which are propagated across the network to match other nodes [10, 17]. This approach heavily depends on not only the quality of seed mappings, but the number of seed mappings. However, because the original networks are always highly private, it is difficult to collect a set of seed mappings with high quality. As a result, some researches aim to find satisfactory seed mappings [16]. The second one directly processes the networks without seed mappings and can still obtain satisfied matching results [4]. This approach makes use of node signatures (e.g., degrees, subgraphs) or structural features (e.g., node similarities, descriptive information) to de-anonymize networks. Without involving the seed mappings, this type of solutions is more general and is easy to setup for de-anonymization. However, because of the expensive computation cost of node similarity (or other descriptive features), they suffer from poor efficiency.

In this paper, we develop a fast seedless de-anonymization approach called RoleMatch. The RoleMatch consists of two phases, node similarity computation and node matching. During the node similarity computation phase, we propose a new similarity measure, named RoleSim++, which is extended from RoleSim [9]. To improve the precision of similarity estimation, RoleSim++ fully exploits the structural information by aggregating both incoming and outgoing neighbors' similarity. Furthermore, based on the observation that correct node mappings tend to have high node similarity, we develop an efficient iterative algorithm, α-RoleSim++, by pruning node pairs with low similarity. In the node matching phase, we introduce a new matching algorithm, named NeighborMatch, which takes advantage of both node similarities and the structural information of neighborhood matches, to efficiently obtain high-quality de-anonymization results.

In addition, previous works only study global de-anonymization, in which the anonymized network and the crawled network are of the similar size with some overlap sub-network. In this paper, we further study the local de-anonymization. In this new situation, the crawled network has much smaller size than the one of the anonymized networks does, and it basically corresponds to a sub-network of the anonymized network. The local de-anonymization is much closer to the real-world applications because usually the crawled network with certain initial node sets is much smaller than the anonymized network. The detailed definitions of the problems are presented in Sect. 2.

Finally, by conducting experiments on three real-world networks (i.e., LiveJournal, Twitter, Enron), the results demonstrate that the precision of RoleMatch can be twice better than the one of the existing solutions. In summary, our contributions are listed as below:

- We propose an efficient and seedless approach, RoleMatch, for de-anonymization.
- We propose a new node similarity measure, RoleSim++, which fully exploits the structural information and improve the de-anonymization performance.
- We develop an efficient iterative algorithm to compute RoleSim++, and introduce a fast node matching algo-

Table 1 Frequently used notations

Notation	Description
$G = (V, E)$	A network G with node set V and edge set E We use G_1 to represent the crawled network with real topology and G_2 to represent the anonymized network
$N_i^{\text{out}}(u)$	Outgoing neighbor set of node u in network G_i
$N_i^{\text{in}}(u)$	Incoming neighbor set of node u in network G_i
$\text{Sim}(u, v)$	Similarity score of the two nodes u and v
$\text{Sim}^k(u, v)$	Similarity score of the two nodes u and v after the kth iteration
$\Delta^{\text{out}}(u, v)$ $(\Delta^{\text{in}}(u, v))$	The larger one between the number of u's outgoing (incoming) neighbors and the number of v's outgoing (incoming) neighbors
$M^{\text{out}}(u, v)$ $(M^{\text{in}}(u, v))$	Node matching between u's outgoing (incoming) neighbors and v's outgoing (incoming) neighbors
$\Gamma^{\text{out}}(u, v)$ $(\Gamma^{\text{in}}(u, v))$	Then the maximum outgoing (incoming) similarity scores of all possible matchings between $N_1(u)$ and $N_2(v)$
α	Parameter to prune unnecessary computations in node similarity computation
β	Decay factor for node similarity RoleSim++
δ	Parameter for the degree of anonymization

rithm by utilizing both the node similarity and neighborhood structural information. The two algorithms effectively reduce the computation cost of RoleMatch.
- We study both global and local de-anonymizations and conduct comprehensive experiments to demonstrate the effectiveness and efficiency of the RoleMatch algorithm on real datasets.

The remains of this paper is organized as follows. In Sect. 2 we present the definition of general de-anonymization problem. Then we give an overview of RoleMatch algorithm in Sect. 3 followed by the elaboration of the node similarity and node matching algorithms used by RoleMatch in Sects. 4–6. The experimental results are presented in Sect. 7. Finally, in Sects. 8 and 9, we introduce the related works and conclude the paper.

2 Preliminaries

We introduce the formal definition of de-anonymization problem and the related metrics of evaluating the complexity of the problem. Then, we describe two variants of the de-anonymization problem, and they are global de-anonymization and local de-anonymization. Finally, we briefly review the two types of de-anonymization algorithms.

2.1 The Definition of De-anonymization Problem

We use $G = (V, E)$ to represent a directed network G where V is the node set and E is the edge set. A single node in the network is denoted by a small letter, such as v. Table 1 summarizes the frequently used notations in this paper. Then the de-anonymization problem is defined as follows.

Definition 1 (*De-anonymization problem*) Given two directed networks $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where G_1 is a crawled network from the original network and G_2 is an anonymized network, and assuming that there exist sub-networks $G_c \subset G_1$ and $G'_c \subset G_2$ such that $G_c.V = G'_c.V$, then the de-anonymization $\mathcal{D}(G_1, G_2)$ is the process to match the nodes between G_c and G'_c as many as possible.

For simplicity, we use G_1 to represent the *crawled network* and G_2 to represent the *anonymized network* in the rest of this paper. Furthermore, G_c implies the overlap between the crawled network and anonymized network, and we called it as *overlap network*.

To measure the difficulty of de-anonymization, we define the noise of a anonymized networks.

Definition 2 (*Noise*) In a problem $\mathcal{D}(G_1, G_2)$, noise is the set of nodes in the networks that do not belong to the overlap network G_c, i.e., $V_1 \cup V_2 \backslash V_c$. To quantify the noise, we introduce an overlap rate $\lambda = \frac{|V_c|}{|V_1 \cup V_2|}$; then, the noise ratio is $1 - \lambda$.

2.2 Global and Local De-anonymization

According to the different situations of real-world activities of de-anonymization, there are two variants of de-anonymization, which are global de-anonymization and local de-anonymization.

To improve the precision of de-anonymization, we would like to get a crawled network from the original network as large as the anonymized one. This is because, in such case, the noise is relatively low, and it has almost no negative impact on de-anonymization. Then the de-anonymization becomes easy. We define this kind of de-anonymization as global de-anonymization below.

Definition 3 (*Global De-anonymization*) Global de-anonymization is the de-anonymization situation where the crawled network and anonymized network are similar in size, i.e., $|G_1| \approx |G_2|$.

Next considering that there are a lot of applications (e.g., community leader detection, influential person identification.), where we are only interested in the information of a part of nodes in the network, this leads to attackers only de-anonymizing the sub-network containing our interested nodes. In such case, we only crawl nodes that are near our targets and build a sub-network as the crawled network for de-anonymization. We define this problem as local de-anonymization as below.

Definition 4 (*Local De-anonymization*) Local de-anonymization is the de-anonymization situation where the crawled network is far smaller than the anonymized one, i.e., $|G_1| \ll |G_2|$.

The local de-anonymization brings the benefit of saving considerable time and space for attackers, but also brings some challenges for de-anonymization algorithms because of the high noise ratio. To be more specific, since in this case the overlapped nodes take up only a small percentage of the anonymized network, all the remaining nodes in the anonymized network become noise which makes it difficult to figure out nodes that are actually matched. Previous researches rarely focused on sub-network attack, and therefore, the state-of-the-art de-anonymization algorithms cannot perform well for the local de-anonymization problem. Our solution will address this drawback.

2.3 Seed-Based and Seedless De-anonymization Algorithm

As briefly mentioned in Introduction, existing de-anonymization algorithms can be divided into two categories: seed-based de-anonymization algorithm and seedless one. The seed-based algorithm requires seed mappings with high quality and extends them to find more mappings. The performance of seed-based algorithms is sensitive to the amount of accurate seed mappings [10]. And the seedless one just uses the properties of network to de-anonymize. Considering the difficulty of obtaining a set of seed mappings, the seedless algorithm has its advantage of simplicity. However, as far as we know, there is no single algorithm that can handle both the situations with/without seed mappings.

In this paper, we mainly discuss the seedless situation and propose a de-anonymization algorithm only replying on the structural information of networks. But we will show that our new solution is able to handle the cases with seed mappings as well, and the performance is improved compared to other classical seed-based algorithms.

3 Overview of RoleMatch

RoleMatch is a fast de-anonymization algorithm, and it supports de-anonymization both with and without initial seed mappings. RoleMatch de-anonymizes the node mappings only based on the structural information of the crawled network and anonymized network.

RoleMatch mainly takes two networks G_1 and G_2 as inputs, and it can accept initial seed mappings if provided. After initializing a similarity matrix *score*, it iteratively computes the all pairs of node similarity according to the structural information. Higher similarity score indicates the higher probability of being a correct node mapping. To improve the effectiveness of de-anonymization algorithm, we propose a new similarity measure, called RoleSim++, which will be introduced in Sect. 4. The new measure captures the information of both outgoing and incoming neighbors, reflecting the structural similarity between a pair of nodes. RoleSim++ is computed iteratively. During each iteration, the score of a pair of nodes is aggregated from the similarities of maximum matching between their neighboring pairs. To reduce the computation cost of RoleSim++, we also develop a threshold-based variant, called α-RoleSim++.

Then, based on the similarity scores calculated in the previous stage, RoleMatch calls function *findNodeMatching* to generate final node mappings. In this function, we apply a matching algorithm called NeighborMatch, which synthetically combines the node similarity and neighborhood feedbacks. Refer to Sect. 6 for the details of NeighborMatch approach.

Furthermore, it is easy for RoleMatch to accept initial seed mappings. This is because RoleMatch computes similarity purely based on the structural information, and seed mappings just provides explicit structural information. The only difference between seedless and seed-based for RoleMatch is that, during the computation of node similarity, if seed mappings are provided, the similarity scores of all the seed pairs remain as one throughout the iterations, and during the node matching phase, the seed pairs are matched ahead of other nodes.

In summary, RoleMatch is a de-anonymization approach based on network structure and works correctly no matter whether high-quality seed mappings are provided or not. Due to the minor influence of the seed, in the following discussions, we mainly focus on the seedless version of RoleMatch.

4 RoleSim++: A New Node Similarity Measure

In this section, we introduce the details of the new node similarity metric, RoleSim++. First, we give the definition of RoleSim++ and its properties. Then we propose an efficient algorithm to compute the new measure.

4.1 Definition of RoleSim++

Many similarity measures of node pairs have been proposed; however, they cannot be applied to the de-anonymization problem directly. For instance, SimRank [7] is a popular similarity measure based on network structure, but it is designed for single network only. Another popular measure [4] is computed based on neighborhood similarity. It uses unnormalized values for iteration so that there is a skew between similarity scores of small-degree nodes and large-degree nodes. After several iterations, the scores of node pairs with small degree drop to almost zero and have no contributions for de-anonymization. In addition, most of the previous similarity measures between two networks mainly discuss about undirected networks, and the directed ones are neglected.

To improve the effectiveness of de-anonymization algorithm, we pay much attention on the structural information including the direction. As a result, we propose a new similarity measure called RoleSim++. RoleSim++ is extended from RoleSim [9] in two aspects: (1) RoleSim++ can model the similarity between two networks and (2) RoleSim++ utilizes the direction information of both incoming edges and outgoing edges.

Before introducing the formal definition of RoleSim++, we clarify some basic notations. Given two vertices $u \in G_1$ and $v \in G_2$, we use $N_1^{\text{out}}(u)$ and $N_1^{\text{in}}(u)$ ($N_2^{\text{out}}(v)$ and $N_2^{\text{in}}(v)$) to denote u's (v's) outgoing and incoming neighbors, respectively. The corresponding degrees are $|N_1^{\text{out}}(u)|$, $|N_1^{\text{in}}(u)|$ ($|N_2^{\text{out}}(v)|$, $|N_2^{\text{in}}(v)|$). Then the maximal outgoing and incoming degrees between u and v are

$$\Delta^{\text{out}}(u,v) = \max\{|N_1^{\text{out}}(u)|, |N_2^{\text{out}}(v)|\},$$
$$\Delta^{\text{in}}(u,v) = \max\{|N_1^{\text{in}}(u)|, |N_2^{\text{in}}(v)|\}.$$

Assume $M^{\text{out}}(u,v)$ is a matching between $N_1^{\text{out}}(u)$ and $N_2^{\text{out}}(v)$, i.e., $M^{\text{out}}(u,v) = \{(x,y) | x \in N_1^{\text{out}}(u), y \in N_2^{\text{out}}(v)$, and no other $(x',y') \in M^{\text{out}}(u,v)$, s.t., $x = x'$ or $y = y'\}$. Similarly, $M^{\text{in}}(u,v)$ is a matching between $N_1^{\text{in}}(u)$ and $N_2^{\text{in}}(v)$. Then the maximum outgoing and incoming similarity scores of all possible matchings between $N_1(u)$ and $N_2(v)$ are, respectively,

$$\Gamma^{\text{out}}(u,v) = \max_{\{M^{\text{out}}(u,v)\}} \sum_{(x,y)\in M^{\text{out}}(u,v)} \text{Sim}(x,y),$$
$$\Gamma^{\text{in}}(u,v) = \max_{\{M^{\text{in}}(u,v)\}} \sum_{(x,y)\in M^{\text{in}}(u,v)} \text{Sim}(x,y),$$

where $\text{Sim}(x,y)$ is the similarity between nodes x and y.

Now the formal definition of RoleSim++ is described as below.

Definition 5 (*RoleSim++*) For a node pair (u, v), its similarity score is computed as

$$\text{Sim}(u,v) = (1-\beta)\frac{\Gamma^{\text{out}}(u,v) + \Gamma^{\text{in}}(u,v)}{\Delta^{\text{out}}(u,v) + \Delta^{\text{in}}(u,v)} + \beta, \tag{1}$$

where the parameter β is a decay factor with the boundary $0 < \beta < 1$.

According to the definition of $\Gamma^{\text{out}}(u,v)$ and $\Gamma^{\text{in}}(u,v)$, we can easily infer that the RoleSim++ can be calculated iteratively. In this paper, we initialize the score matrix as an all-one matrix, i.e., $\text{Sim}(u,v) = 1$ for all node pairs.

4.2 Properties of RoleSim++

To show that RoleSim++ is a valid similarity measure, we prove that RoleSim++ is converged and its tolerance to the noise of anonymization is bounded.

First, the following lemma shows that

Lemma 1 (Non-Increasing) *Let* $\text{Sim}^k(u,v)$ *be the similarity score of* (u, v) *after k iterations. Then* $\text{Sim}^{k-1}(u,v) \geq \text{Sim}^k(u,v)$ *for all k and all node pairs* (u, v).

Proof See "The Proof of Lemma 1" of Appendix. □

With the non-increasing property and $\text{Sim}^k(u,v) \geq \beta$, the following convergence property can be derived immediately. We have:

Proposition 1 (Convergence) *The similarity measure in Definition 5 converges for every pair of nodes* (u, v), *i.e.,* $\lim_{k\to\infty} \text{Sim}^k(u,v) = \text{Sim}(u,v)$.

Next we show the impact of β on the convergence rate of the RoleSim++ score, and the result is that the difference between $\text{Sim}^k(u,v)$ and $\text{Sim}(u,v)$ decreases exponentially with $(1-\beta)$.

Proposition 2 *For every pair of nodes* (u, v), *let* $\epsilon^k(u,v)$ *be* $\text{Sim}^k(u,v) - \text{Sim}(u,v)$, *then*

$$\epsilon^k(u,v) \leq (1-\beta)^{k+1}.$$

Proof See "The Proof of Proposition 2" of Appendix. □

When computing similarity scores on real-world networks, because of the small diameter of social networks, it shows that 5 rounds of iteration will be enough for de-anonymization accuracy. We will discuss this further in Sect. 7.

The Tolerance of RoleSim++ to the Noise of Anonymization When applying RoleSim++ to de-anonymize networks, there is a lower bound for expected similarity scores of correct matches, which shows its effectiveness for network de-anonymization. The lower bound is influenced by the complexity of how the network is anonymized. In other words, the lower bound is related to the noise in an anonymized network.

First, we introduce a parameter δ to describe the complexity of anonymization algorithms on the network. Two networks G_1 and G_2 are δ-anonymized if the following three conditions are satisfied:

1. for each node u in G_1, there exists exactly one node v in G_2 such that u and v are originally the same,
2. u and v have at least a proportion $(1-\delta)$ of common neighbors,[1] i.e., $\frac{|N^{\text{out}}(u)\cap N^{\text{out}}(v)|}{\Delta^{\text{out}}(u,v)} \geq 1-\delta$ and $\frac{|N^{\text{in}}(u)\cap N^{\text{in}}(v)|}{\Delta^{\text{in}}(u,v)\}} \geq 1-\delta$,
3. the ratio of incoming neighbors of u and v, and the ratio of outgoing ones are both between $(1-\delta)$ and $1/(1-\delta)$.

Through studying the previous anonymization algorithms, we found that most of the commonly used network anonymization algorithms have the above three properties. For example, *Sparsify* anonymizes a network by deleting $p\%$ of the edges, and therefore, δ equals $p\%$. *Perturb* and *Switch* also have these properties.

Then the following proposition gives the estimation of the lower bound of $\text{Sim}^k(u,v)$ with parameter δ.

Proposition 3 *Let u and v be a correct match where G_1 and G_2 are δ-anonymized, then $\text{Sim}^k(u,v) \geq c_k$, where $c_k = a^k + \beta\frac{1-a^k}{1-a}$, and $a = (1-\beta)(1-\delta)$.*

Proof See "The Proof of Proposition 3" of Appendix. □

For example, when β is set to 0.15, and the parameter δ in anonymization algorithms is set to 10%, after five iterations, the similarity score of each correct matched node pair satisfies that $\text{Sim}^k(u,v) > 0.73$. In particular, when G_1 and G_2 are isomorphic, which means that $\delta = 0$ and $a = 1-\beta$, then $1-(1-\beta)^k = \beta\sum_{i=0}^{k-1}(1-\beta)^i$.

Consequently, for each k we have $c_k = 1$ and $\text{Sim}^k(u,v) = 1$. This is consistent with the fact that two networks are isomorphic.

[1] The case of no incoming or outgoing neighbors is omitted.

5 Solutions of Computing RoleSim++

5.1 Basic Solution for RoleSim++

To compute the similarity score of RoleSim++, the basic solution calculates all pairwise score iteratively in brute-force way. Algorithm 1 describes the procedure of the solution. First, the similarity matrix *score* is initialized as an all-one matrix (Line 1). In each iteration, the similarity scores of node pairs are updated according to Eq. (1). The function $\gamma(N(u),N(v))$ (Line 4) computes the maximum matching between neighboring pairs of u and v (i.e., Γ^{out} and Γ^{in}). Considering that the exact maximum matching for bipartite network is computationally expensive, we adopt a greedy approximation algorithm in our implementation, just as Fu et al. [4] and Jing et al. [9] did. However, the basic solution is expensive, and the computation complexity for each round is at least $\Omega(|V_1||V_2|d^2)$, where d is the average degree of the nodes. The algorithm can only handle networks with thousands of nodes.

Algorithm 1 Naive RoleSim++ Computation

```
Input: $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$
Output: $score[][]$
1: $score[][] \leftarrow$ MatrixAllOne
2: for $i = 1 \rightarrow nRounds$ do
3:     for each $(u,v) \in V_1 \times V_2$ do
4:         $r \leftarrow \frac{\gamma(N_1^{out}(u),N_2^{out}(v))+\gamma(N_1^{in}(u),N_2^{in}(v))}{\Delta^{out}(u,v)+\Delta^{in}(u,v)}$
5:         $score'(u,v) \leftarrow (1-\beta)\cdot r + \beta$
6:     end for
7:     $score[][] \leftarrow score'[][]$
8: end for
9: return $score[][]$
```

Here we illustrate the computation of Algorithm 1 through a simple example. Consider a specific node u_1 in Fig. 1, and set β to be 0.15. In the first iteration, we have

$$\text{Sim}(u_1,v_1) = (1-\beta)\frac{0+2}{0+3} + \beta = 0.72,$$
$$\text{Sim}(u_1,v_2) = (1-\beta)\frac{0+1}{2+3} + \beta = 0.32,$$
$$\text{Sim}(u_1,v_3) = (1-\beta)\frac{0+1}{2+3} + \beta = 0.32,$$
$$\text{Sim}(u_1,v_4) = (1-\beta)\frac{0+0}{1+3} + \beta = 0.15,$$
$$\text{Sim}(u_1,v_5) = (1-\beta)\frac{0+1}{0+3} + \beta = 0.43.$$

It is clear that even if the first round uses only the one-hop neighbors, the similarity between node u_1 and v_1 still dominates all other possible pairs related to u_1. The change in similarity scores related to node u_1 over iterations is shown in Table 2. After five iterations, similarity scores of correct matchings stand out, as is shown in Table 3.

Table 2 Change in similarity scores related to node u_1 in Fig. 1 over five iterations

	(u_1, v_1)	(u_1, v_2)	(u_1, v_3)	(u_1, v_4)	(u_1, v_5)
First	0.72	0.32	0.32	0.15	0.43
Second	0.56	0.32	0.27	0.15	0.35
Third	0.47	0.23	0.24	0.15	0.30
Fourth	0.41	0.22	0.23	0.15	0.28
Fifth	0.38	0.21	0.22	0.15	0.27

Table 3 Similarity scores of each node pair in Fig. 1 after five iterations

	v_1	v_2	v_3	v_4	v_5
u_1	0.38	0.20	0.22	0.15	0.26
u_2	0.15	0.38	0.36	0.31	0.15
u_3	0.27	0.36	0.38	0.24	0.31
u_4	0.15	0.26	0.26	0.34	0.15

5.2 α-RoleSim++: A Fast Solution

To scale our de-anonymization approach to large networks, we design a fast solution of computing RoleSim++, called α-RoleSim++. For de-anonymization, each node in G_1 has at most one correspondence (correct match) in G_2. This means that our main concerns are those node pairs with higher similarity scores, which are more likely to be correct matches.

Based on this observation, we propose a heuristic rule to speed up the computation.

Heuristic 1 *In each iteration, only the similar node pairs with high similarity are reserved and others can be discarded.*

Following the heuristic rule, we propose a new efficient computation method, α-RoleSim++. The α-RoleSim++ can substantially reduce the computational cost but still retain the accuracy. In α-RoleSim++, the similarity formula is revised as follows. Let $\mathrm{Sim}_k^\theta(u,v)$ denote the threshold-sieved similarity score of (u, v) on the kth iteration, where the threshold $\theta = \theta(u,\alpha)$ relies on parameter α and node u, and $0 < \alpha, \theta < 1$.

The iterative version of $\mathrm{Sim}_k^\theta(u,v)$ is given as follows:

$$\begin{aligned}
&\mathrm{Sim}_0^\theta(u,v) = 1; \\
&\mathrm{Sim}_{k+1}^\theta(u,v) = (1-\beta)\tfrac{\Gamma^{\mathrm{out}}(u,v)+\Gamma^{\mathrm{in}}(u,v)}{\Delta^{\mathrm{out}}(u,v)+\Delta^{\mathrm{in}}(u,v)} + \beta, && \text{if } \mathrm{Sim}_k^\theta(u,v) \geq \theta(u,\alpha); \\
&\mathrm{Sim}_{k+1}^\theta(u,v) = \beta, && \text{otherwise.}
\end{aligned}$$

More specifically, to reduce space and time consumption, in each iteration we keep the node pairs with similarity above θ and discard other node pairs. In this way, we only need to maintain a relatively small set of similarity scores, while dissimilar pairs are gradually filtered out.

Since the goal of de-anonymization is to identify each node in G_1, we need to keep a portion of candidates (nodes from G_2) for each node u in G_1. Consequently, the threshold θ should be related to the node u in each iteration for dynamically maintaining a proper list of candidates. We define θ as $\theta(u,\alpha) = \alpha \cdot top(u)$, where $top(u)$ is the highest similarity score related to u in the last round, and it is easy to figure out that θ is dynamically determined by the similarities with respect to node u.

Algorithm 2 describes the details of α-RoleSim++ computation. The main framework still remains the same as Algorithm 1. The differences are below. At Line 6, the threshold θ is decided by both parameter α and the similarity scores related to node u. Later when visiting candidate pairs (Line 7), those with similarity score below the threshold are filtered out and others are updated for the iteration (Lines 8–11).

Algorithm 2 A fast solution, α-RoleSim++

```
Input: $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$
Output: $score[][]$
 1: $score[][] \leftarrow$ the similarity scores of $1^{st}$ iteration in the basic solution.
 2: for $i = 1 \rightarrow nRounds$ do
 3:     $score' \leftarrow$ EmptyHashMap
 4:     for each $u \in V_1$ do
 5:         $top(u) \leftarrow$ highest score related to $u$
 6:         $\theta \leftarrow \alpha top(u)$
 7:         for each $v \in V_2$ s.t. $(u,v) \in score[][]$ do
 8:             if $score(u,v) \geq \theta$ then
 9:                 $r \leftarrow \frac{\gamma(N_1^+(u),N_2^+(v))+\gamma(N_1^-(u),N_2^-(v))}{\Delta^+(u,v)+\Delta^-(u,v)}$
10:                 $score'(u,v) \leftarrow (1-\beta)\cdot r + \beta$
11:             end if
12:         end for
13:     end for
14:     $score[][] \leftarrow score'[][]$
15: end for
16: return $score[][]$
```

Properties of α-RoleSim++ We study the property through analyzing the threshold-sieved similarity measure. First, by induction on the number of iterations, we have:

Property 1 *The threshold-sieved similarity score of each pair of nodes (u, v) is non-increasing, i.e., $\mathrm{Sim}_k^\theta(u,v) \geq \mathrm{Sim}_{k+1}^\theta(u,v)$ for each k.*

Property 2 *The value of threshold-sieved similarity scores is lower than the standard RoleSim++ scores, i.e., $\mathrm{Sim}_k^\theta(u,v) \leq \mathrm{Sim}_{k+1}(u,v)$ holds for all pairs of nodes.*

From Property 1 we know that the iterative computation of α-RoleSim++ converges. The convergent similarity score of (u, v) is denoted as $\mathrm{Sim}^{\theta}(u, v)$.

In addition, to choose the value of parameter α, there is a trade-off between the accuracy of similarity scores and the computational cost. If α is set to a relatively low value, fewer node pairs will be filtered out in each iteration, resulting in higher computational cost, while the de-anonymization accuracy will be closer to that of standard RoleSim++. In particular, if α is set to zero then it will be exactly the same as the standard RoleSim++. We will study the influence of α on accuracy in Sect. 7.

6 NeighborMatch: An Effective Node Matching Algorithm

In this part, we introduce our method to find a good mapping between the anonymized network and the crawled one, based on the pre-computed similarity scores.

Intuitively, in order to find the mapping based on node similarity, the maximum weighted matching for bipartite network is a good option. By using Karnik–Mendel (KM) algorithm [12], the maximum matching can be computed in $O(n^3)$, where n is the number of nodes. Since the maximum matching is computationally expensive, it can hardly be applied to large networks. Another solution proposed by Fu et al. [4] is a greedy algorithm, which offers an approximation of the globally optimal matching in $O(n^2 \log n)$, with less accuracy than KM algorithm.

However, both above approaches simply maximize the sum of similarity scores, and the structural information of the network is neglected during the matching phase. Actually the links between a pair of nodes and their neighbors contain valuable information that can help us de-anonymizing a network with higher accuracy. We propose a new matching algorithm, NeighborMatch, based on two observations: *First, correct mappings tend to have higher similarity scores and second, a pair of nodes is more likely to be a correct mapping if their neighbors are correct mappings.* More specifically, NeighborMatch assigns a priority for each pair of nodes, and it follows the priority to generate matchings.

To automatically assign the priority online, we use the idea of the percolation network matching method proposed by Kazemi et al. [10]. Percolation network matching (PGM) generates the results based on seeds. The seed pairs are marked as matched at the beginning. Then, node pairs whose number of matched neighbors is higher than the threshold r are matched repeatedly, until there are no more unmatched pairs with at least r neighbors being matched. NeighborMatch uses the pair with highest score as the "seed" at the beginning, and in the following iterations, it always picks the pair with highest score when there are multiple candidates.

Algorithm 3 NeighborMatch: an efficient node matching algorithm

```
Input: $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, $score$
Output: $nodeMapping$
 1: candidate set $A \leftarrow \emptyset$
 2: $nodeMapping \leftarrow \emptyset$
 3: while there exists an unmatched pair do
 4:     while $A$ is empty do
 5:         match the pair $(u, v)$ with highest score
 6:         add $(u, v)$ to $nodeMapping$
 7:         increase the score of neighboring pairs of $u$ and $v$ by $score(u, v)$
 8:         $A \leftarrow$ all unmatched pairs with score $\geq r$
 9:     end while
10:     pick a pair $(u, v)$ with highest score in $A$
11:     add $(u, v)$ to $nodeMapping$
12:     remove $(u, v)$ from $A$
13:     increase the score of neighboring pairs of $u$ and $v$ by $score(u, v)$
14: end while
15: return $nodeMapping$
```

Algorithm 3 illustrates the procedure of NeighborMatch finding the node mappings. First, from Lines 4 to 9, it matches node pairs with highest similarity score and increases the scores of their neighbors by one, until there are some candidate pairs with at least r neighbors being matched. Then it matches all the candidates in sequence of their scores from the highest to the lowest and spreads score to their neighbors (Lines 10–13). These two steps are repeated until each node in G_1 is matched with some node in G_2. Taking the similarity scores in Table 3 as an example, in the first iteration, NeighborMatch first selects a matching pair (u_1, v_1), then increases the scores of (u_2, v_2), (u_2, v_3), (u_3, v_2), (u_3, v_3), (u_4, v_2), (u_4, v_3) by 0.38, respectively, and adds them into set A. After another four iterations, all node matchings will be correctly generated.

Since NeighborMatch is a variant of percolation network matching by using different seeds, the theoretical results [10] still hold and guarantee the performance of NeighborMatch. For instance, assuming that at the beginning, m pairs of nodes with highest similarity scores are matched, and m reaches the critical value according to Theorem 1 from the work [10], then with high probability, at least $n - o(n)$ nodes can be de-anonymized successfully, where $n = |V_1 \cap V_2|$.

Moreover, NeighborMatch has several advantages over the original percolation network matching. Network percolation requires all the candidate pairs to have at least r neighbors matched previously, so the matching process gets stuck when there are no valid candidates. Our algorithm avoids getting stuck because the similarity scores provide a natural and reasonable choice of candidate pairs, i.e., picking out the one with highest score among all the unmatched pairs. Thus, our matching algorithm is capable to match more node pairs, even those whose degrees are less than the threshold r.

7 Experimental Studies

In this section, we evaluate the performance of RoleMatch through extensive experiments. First, we conduct experiments of tuning parameters for the RoleMatch. Then we

Table 4 Dataset statistics

Dataset	#V	#E	Avg. degree	Diameter	Avg. clustering coefficient	Type
LiveJournal	4,847,571	68,993,773	14.23	16	0.2742	Directed
Twitter	81,306	1,768,149	21.75	7	0.5653	Directed
Enron	36,692	367,662	10.02	11	0.4970	Undirected

compare the performance of RoleSim++ measure and NeighborMatch algorithm to the existing solutions [4, 10], respectively. Afterward we describe the performance of RoleMatch as a whole for the global de-anonymization and local de-anonymization. Finally, we also compare RoleMatch with existing seed-based de-anonymization algorithms.

7.1 Experiment Settings

All the algorithms are implemented in C++ and compiled with -O3 options. The experiments were run on a Linux server, which is equipped with an Intel Xeon E5620 CPU (16 cores, 2.4 GHz) and 64 GB memory. Furthermore, we used 16 threads to parallelize the computation of each iteration.

Datasets In the experiments, three real-world datasets are used. They are LiveJournal,[2] Twitter[3] and Enron.[4] The statistics of datasets are described in Table 4.

In addition, we follow the approach proposed in [4] to generate small networks, called synthesized datasets. The basic idea is that, given a large network G, we first randomly extract a sub-network from G as a seed network, denoted as $G_s = (V_s, E_s)$, and use the nodes in G_s to generate a crawled network $G_1 = (V_1, E_1)$ and an anonymized network $G_2 = (V_2, E_2)$ with satisfying $V_s = V_1 \cup V_2$. Recall the definition of λ; the overlap rate is $\lambda = \frac{|V_1 \cap V_2|}{|V_s|}$.

Then, we use breadth first search (BFS) algorithm to generate synthesized networks with arbitrary λ. More specifically, we use BFS to create an overlap network $G_c = (V_c, E_c)$ from G_s where $|V_c| = \lambda \times |V_s|$. The overlap network G_c is treated as a sub-network to both G_1 and G_2. And then the remaining node set $V_s \backslash V_c$ is split into two parts of same size, V_1' and V_2'. Finally, V_1 is $V_1' \cup V_c$, and V_2 is $V_2' \cup V_c$. Furthermore, we apply a selected anonymization algorithm on network G_2 to build the anonymized network. In the following experiments, we use Syn($|V_s|$,λ) to represent a synthesized dataset generated from LiveJournal dataset. For example, Syn(10,000, 50%) means a synthesized dataset created by setting $|V_s|$ to 10,000 and overlap λ to 50%.

Anonymization Algorithms Here we list all the anonymization algorithms used in the experiments. As introduced in Sect. 2, we use parameter δ to represent the degree of anonymization by an algorithm. The larger the δ is, the more edges in the anonymized network are changed. Following the previous work [4], we set $\delta = 0.1$ for each anonymization algorithm, i.e., about 10% of the edges are modified.

1. *Naive Anonymization* The naive approach simply shuffles the identifiers of nodes and leaves the structure as it is.
2. *Sparsify*(δ) The Sparsify approach removes $\delta|E|$ edges randomly, where the parameter δ controls the number of deleted edges.
3. *Perturb*(δ) The Perturb approach [3] first removes edges in exactly the same way as the Sparsify does and then adds false edges randomly until the number of edges of the anonymized network is the same as the original network. This approach can be viewed as a kind of simulation of social network evolution or "unintended" anonymization.
4. *Switch*(δ) The switch approach randomly selects two edges (i_1, j_1) and (i_2, j_2), where (i_1, j_2) and (i_2, j_1) are not in the network. The selected edges are then "switched," i.e., (i_1, j_1) and (i_2, j_2) are deleted, and (i_1, j_2) and (i_2, j_1) are added to the network. The procedure is repeated $\delta|E|/2$ times, and $\delta|E|$ edges are added and $\delta|E|$ edges are deleted.

De-anonymization Algorithms The de-anonymization algorithms we compared are described as follows.

1. *Baseline* (*BaseSim and BaseMatch*) We use the de-anonymization algorithm [4] as our baseline algorithm. The baseline algorithm consists of two parts: similarity computation phase and the node matching phase. The similarity measure in the baseline algorithm is referred as BaseSim, and the node matching algorithm is referred as BaseMatch.
2. *Seed Baseline* We use the seed-based mapping algorithm proposed by Kazemi et al. [10] as our seed-based mapping baseline.
3. *RoleMatch* The RoleMatch refers to the proposed algorithm, where two new similarity measures, RoleSim++ and α-RoleSim++, are used. Moreover, NeighborMatch

[2] http://snap.stanford.edu/data/soc-LiveJournal1.html.

[3] http://snap.stanford.edu/data/egonets-Twitter.html.

[4] https://snap.stanford.edu/data/email-Enron.html.

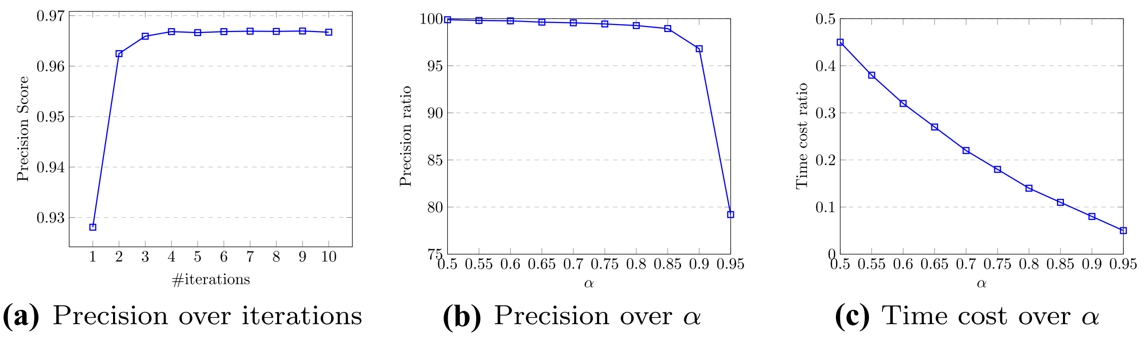


Fig. 2 Results of parameter tuning on Syn(10,000, 100%)

is used as the node matching algorithm, where the threshold r is set to 2 in the experiments.

Evaluation Criteria In order to evaluate the overall performance of de-anonymization algorithm comprehensively, we introduce three different metrics.

Precision Score This is a metric for evaluating effectiveness of de-anonymization algorithms. Assume $M(u, v)$ is the set of correct matching pairs; then, the precision score is $\frac{|M(u,v)|}{|V_1 \cap V_2|}$. The higher the precision score is, the more of correct mappings an algorithm generates.

Top-k Precision To compare the effectiveness of different similarity measures, we introduce another two precision metrics, top-1 precision and top-m% precision. The two metrics are defined according to the different criterions of the ranking positions of nodes in correct matching pairs. Given a pair $(u, v) \in M(u, v)$, and $u \in V_1$ and $v \in V_2$, we sort every $v_i \in V_2$ in decreasing order based on $\mathrm{Sim}(u, v_i)$, the rank of vertex v, denoted as $r(v)$, is the position of v in the sorted vertex list.

$$\text{Top-1 precision is } \frac{|(u,v)|(u,v) \in M(u,v) \wedge r(v) = 1|}{|V_1 \cap V_2|}.$$

$$\text{Top-m\%precision is } \frac{|(u,v)|(u,v) \in M(u,v) \wedge r(v) \le \frac{m \times |V_2|}{100}|}{|V_1 \cap V_2|}.$$

Execution Time This is a metric for evaluating efficiency. It is the time cost of running algorithms in the experiments.

7.2 Parameter Tuning

Before evaluating the RoleMatch algorithm, two parameters need to be tuned, and they are the number of iterations and the threshold αin α-RoleSim++. To showcase the parameter tuning processing, we conducted the tuning experiments on Syn(10,000, 100%), and the network is anonymized by *Naive Anonymization*. Other anonymization algorithms can be tuned in the same way.

The Number of Iterations Figure 2a shows the precision of running RoleMatch in 10 iterations. In this experiment, RoleSim++ is used as the similarity measure and NeighborMatch is used to generate the node matching. It is clear that the precision remains almost the same after five rounds of iterations, so we set the number of iterations to 5 in the following experiments.

Threshold Parameter α For computing α-RoleSim++, we use threshold parameter α to limit the number of nodes involved in the computation. The lower the parameter α is, the more the node pairs each iteration computes, resulting in higher time consumption. We set α from 0.95 to 0.50, decreasing by 0.05 in the tuning process. Figure 2b, c shows the precision ratio and time cost ratio between the two algorithms, respectively. The precision ratio is defined as the ratio between the precision of α-RoleSim++ and the precision of RoleSim++. Similarly, the time ratio is the ratio between the execution time of α-RoleSim++ and the execution time of RoleSim++ in the experiment result. From the figures, we clearly see that when α increases, the time consumption reduces almost linearly, and the precision is well retained when *alpha* ≤ 0.85. Therefore, we set α to 0.85 in the following experiments.

7.3 The Performance of RoleSim++

Effectiveness To demonstrate the effectiveness of RoleSim++ and α-RoleSim++, we compare the new similarity measures with BaseSim and RoleSim. In the experiment, we use the Syn(10,000, 50%) and Syn(10,000, 100%). The original networks are anonymized by four previously mentioned anonymization algorithms. And we use top-1 precision and top-m% precision to evaluate the effectiveness.

Figure 3a, b shows that both RoleSim++ and α-RoleSim++ have much higher top-1 precision than

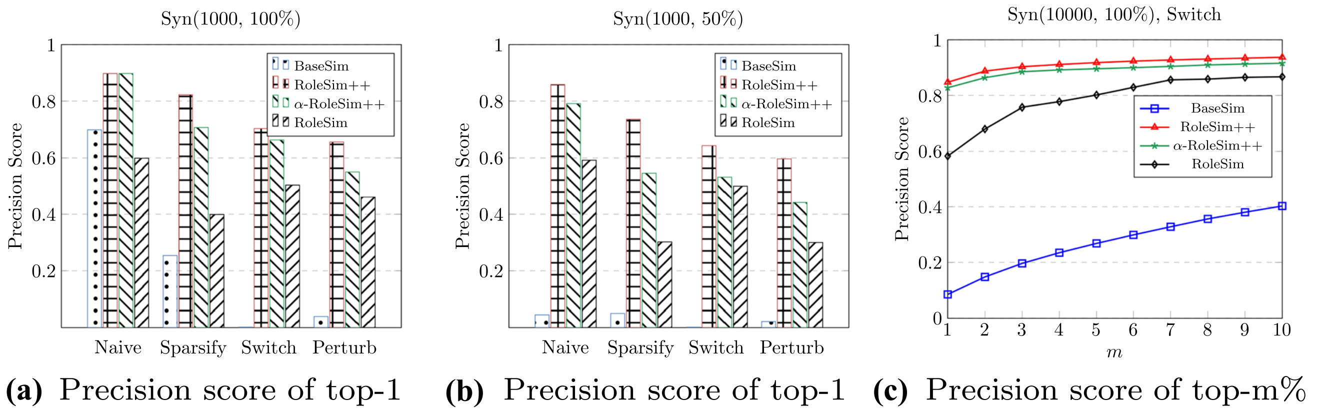


Fig. 3 Results of evaluating the performance of RoleSim++

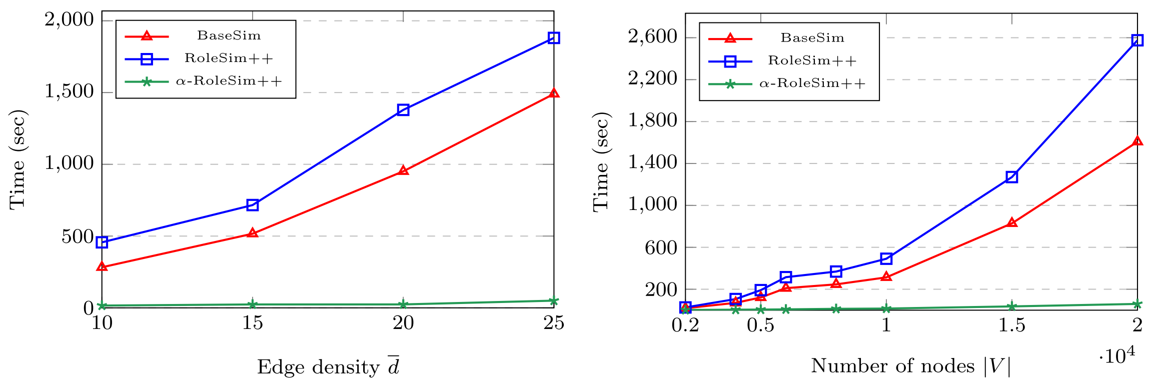


Fig. 4 Execution time over $|V|$ and $\overline{d}$

BaseSim under all four anonymization algorithms, especially for the situation of lower overlap rate ($\lambda = 50\%$). RoleSim is in between but not so good as RoleSim++ and α-RoleSim++. This is because that RoleSim++ takes two directions into consideration, while RoleSim treats all edges as undirected. Besides, the gap between Baseline and RoleSim++ becomes larger when the overlap rate is lower. Figure 3c shows that even when the comparison condition is relaxed from top-1 to top m%, both RoleSim++ and α-RoleSim++ still outperform the BaseSim. RoleSim++ and α-RoleSim++ achieve more than 80% precision in the top-1% precision measure, while the BaseSim can only achieve nearly 40% even in the top-10% precision measure.

In summary, because RoleSim++ fully exploits the structural information of a network, it improves the precision of estimating the node similarity.

Efficiency We compare the execution time of BaseSim, RoleSim++ and α-RoleSim++ to verify the efficiency of α-RoleSim++. Two aspects are taken into consideration: the average edge density $\overline{d}$ in the network and the number of nodes $|V|$ in the network. We generate synthesized datasets from LiveJournal. When varying $\overline{d}$, $|V|$ is 10,000, and edges are randomly selected from the V. When varying $|V|$, the induced subgraph of V is used.

Figure 4 shows the variation in execution time with the increase in $\overline{d}$ and $|V|$, respectively. As the edge density and the number of nodes increase, the execution time of RoleSim++ and BaseSim increases rapidly, while α-RoleSim++ is always faster and its execution time increases far slower. This is because α-RoleSim++ can prune many unqualified node pairs to speed up the computation. Moreover, the α-RoleSim++ can de-anonymize Twitter dataset in less than 30 minutes. Both the BaseSim and the RoleSim++ similarity are unable to finish 5 iterations in 24 h.

7.4 The Performance of NeighborMatch

To verify that the NeighborMatch can lead to a better de-anonymization precision in different situations, we compare our new algorithm with BaseMatch based on RoleSim++ and BaseSim. First, we compare these two matching algorithms on RoleSim++ similarity measure

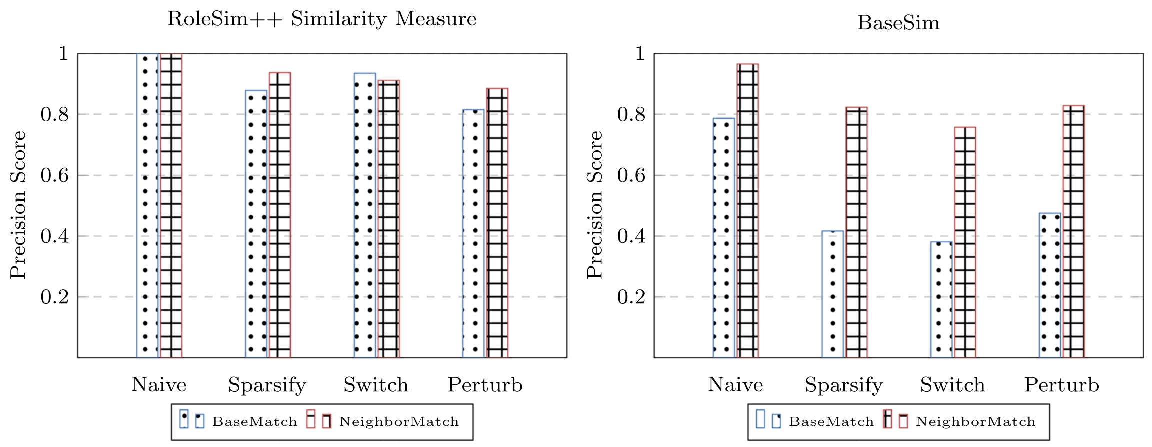


Fig. 5 Precision comparison of NeighborMatch and BaseMatch algorithm based on different similarity measures

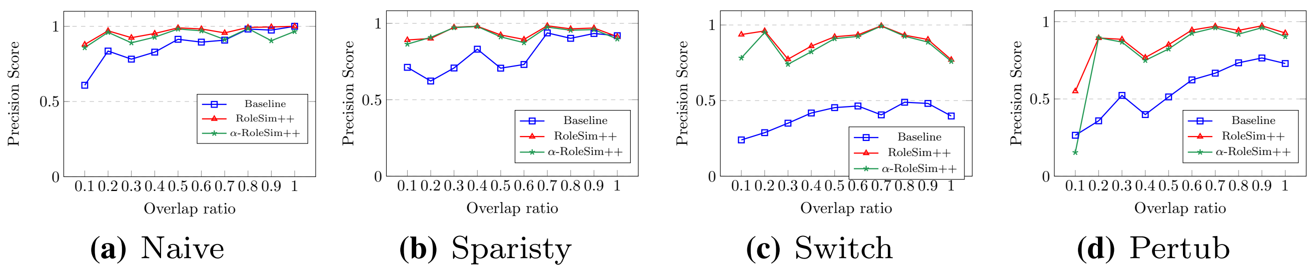


Fig. 6 Precision on synthesized datasets with different overlap rates

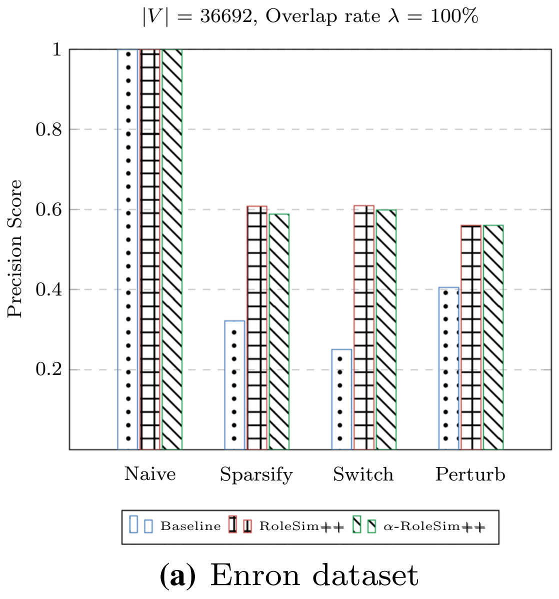
in order to show that NeighborMatch can utilize the well-measured similarity information to match more nodes correctly. Then we compare these two algorithms on BaseSim to show that even when the similarity measure has its deficiency, NeighborMatch can still generate much more correct node mapping pairs by using the structural information (Sect. 6). The experiments are conducted on Syn(10,000, 100%). The results are shown in Fig. 5.

It is clear to see that NeighborMatch can reach a better average precision score against BaseMatch on RoleSim++ similarity when anonymization methods (Sparsify and Perturb) are used. The BaseMatch outperforms the NeighborMatch in Switch anonymization, but in that case both algorithms can de-anonymize more than 90% of nodes and the difference is subtle. When running on BaseSim, NeighborMatch has a much better performance than BaseMatch. NeighborMatch can achieve at least 75% de-anonymization precision when different anonymization algorithms are applied, while the BaseMatch performs poorly when non-trivial anonymization algorithms are applied, resulting in less than 50% precision in Sparsify, Switch and Perturb, respectively. The advantage of NeighborMatch is gained from making use of neighborhood structural information.

7.5 Precision Comparison of Global De-anonymization

To demonstrate the effectiveness of RoleMatch for processing global de-anonymization, we compare all three de-anonymization algorithms (the baseline algorithm, RoleMatch with RoleSim++ and RoleMatch with α-RoleSim++) on both real datasets, Enron, Twitter and synthesized datasets, and use Naive, Sparsify, Switch and Perturb anonymization algorithms to generate anonymized networks. Each experiment is ran five times and the average precision score is reported.

Figure 6 presents the results on synthesized datasets. For each anonymization algorithm, we create ten Syn(10,000, λ), where λ is in the range of [0.1, 1]. First, it is easy to figure out that RoleMatch outperforms the baseline across different overlap rates, and RoleMatch with RoleSim++ and RoleMatch with α-RoleSim++ have the similar precision scores. But the improvements between



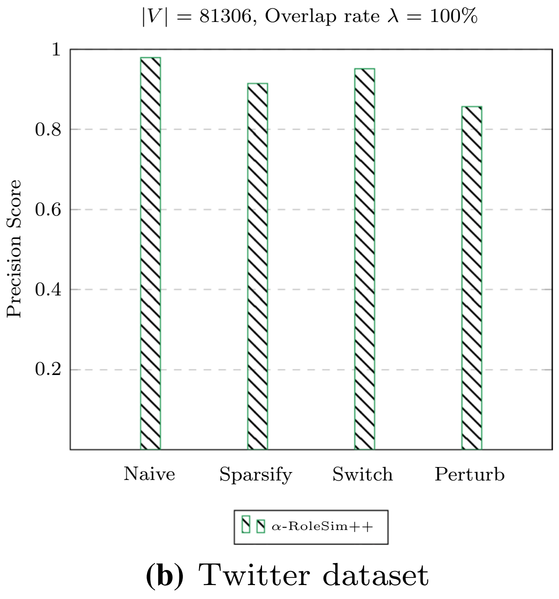


Fig. 7 Precision over different anonymization algorithms on *Enron* and *Twitter*

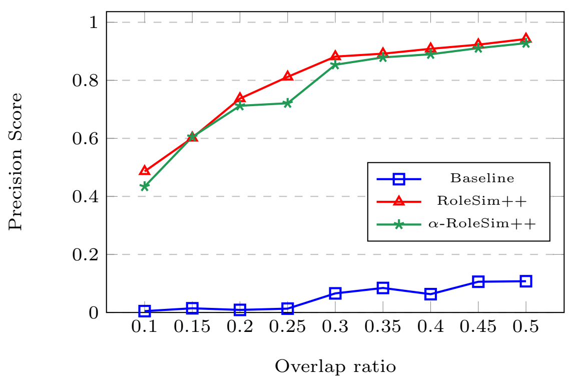
RoleMatch and the baseline vary regard to the anonymization algorithms. When the network is anonymized by Naive method, all three de-anonymization algorithms perform almost equally. This is because that with the naive anonymization, the overlap network remains the same between the crawled network and the anonymized network. Except when the overlap is extremely low, the baseline suffers from the noise. For example the overlap rate is 10% when using Naive anonymization, the precision score of baseline algorithm is about 60%, while the other two can still achieve around 86% and 88 %. When different anonymization algorithms are applied, RoleMatch algorithms have much better performance on precision than the baseline algorithm. For anonymization method Switch, RoleMatch algorithms have almost twice the precision of the baseline, and for Sparsify and Perturb anonymization, the advantage is about 10%. Second, with the increasing of overlap rate, the precision score goes up generally among different anonymization algorithms. This is because the lower overlap brings higher data noise.

Figure 7 describes the results on real datasets: Enron and Twitter. In these two datasets, we set overlap rate to 100%. The performance results are similar to the one on synthesized datasets, and that is, RoleMatch performs the best. For the Twitter dataset, only the result of RoleMatch with α-RoleSim++ is presented. The baseline algorithm and RoleMatch with RoleSim++ need to store an $n \times n$ matrix for node similarity, they cannot process the dataset successfully because of the limited memory. However, RoleMatch with α-RoleSim++ is efficient in memory usage and it can produce a satisfying result on this dataset, de-anonymizing over 85% nodes.

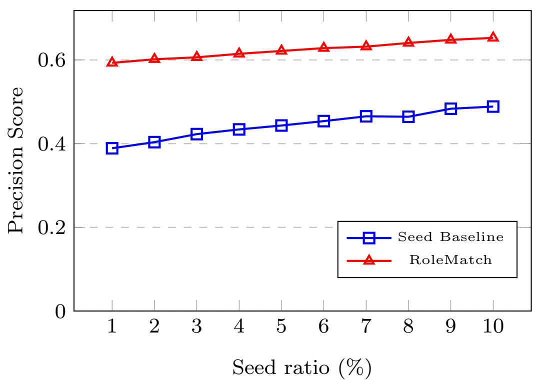
7.6 Precision Comparison of Local De-anonymization

In this subsection, we consider the local de-anonymization case. To generated G_1 and G_2 satisfying $|G_1| \ll |G_2|$, we first extract a sub-network G_1 with 10,000 nodes from LiveJournal and then randomly crawl a sub-network G_0 from G_1 with a given size. We anonymize the sub-network G_0 to generate an anonymized network G_2. Here Sparsify is used as the anonymization algorithm with the probability of deleting an edge set to 0.1. Furthermore, we set the overlap rate of G_1 and G_0 from 10 to 50%, and the overlap part is exactly G_0. The results are presented in Fig. 8a.

From the figure, we can see that generally the precisions of these three algorithms increase as the overlap rate increases. In spite of the general tendency, there is sharp difference between the precisions of the baseline algorithm and the RoleMatch algorithms. The precision of the baseline algorithm basically is always below 0.1 in the experiment; however, when the overlap rate is above 20%, both the RoleMatch with RoleSim++ and RoleMatch with α-RoleSim++ can de-anonymize 80% of the overlapped nodes. The difference in precision reveals that the RoleMatch algorithms are far more effective in local de-anonymization situations, especially when the overlap rate is not too low. And they can



(a) Local De-anonymization on LiveJournal



(b) Seed-Based De-anonymization on Enron

Fig. 8 Precision comparison with different settings

easily de-anonymize most of the overlapped nodes with no need to crawl the whole network.

Furthermore, compared with results of global de-anonymization, we can see that the improvement in RoleMatch is much larger in local de-anonymization case. This is because there is much more noise in local de-anonymization case, and RoleMatch is robustness to the noise as analyzed in Sect. 4.2.

7.7 Seed-Based De-anonymization

Finally, we conduct experiments to demonstrate that the RoleMatch can be adapted to situations where seed mappings are provided for de-anonymization.

The seed version RoleMatch was ran on the Enron dataset, with the correct seed ratio from 1 to 10%. The anonymization algorithm applied here is Sparsify, and we compared this seed version RoleMatch with α-RoleSim++ and the seed mapping de-anonymization from the work [10].

The experiment result is shown in Fig. 8b. Basically the precision goes up with the seed ratio increasing, but there is a distinct gap between the two algorithms. The precision of RoleMatch is higher than that of the seed baseline algorithm by about 0.2, which reveals the effectiveness of seed version RoleMatch.

8 Related Works

The work of de-anonymizing networks is highly related to three topics. They are (1) anonymization algorithms, (2) de-anonymization algorithms and (3) node similarity measures. In the following subsections, we describe the related work of each topic separately.

8.1 Anonymization Algorithms

According to a previous survey [18] on social network anonymization, anonymization algorithms are classified into three categories: K-anonymity, edge randomization and clustering-based generalization.

K-anonymity [15, 21] modifies network structure by edge deletions and additions, so that each node in the modified network is indistinguishable with at least $K-1$ other nodes, in terms of some structural patterns like degree. This approach have good performance in anonymity but (1) is relatively complex to implement and (2) may have modifications to network structure to a too large extent. Edge randomization modifies the network via random deletions, additions or switches of edges. It protects user privacy in a probabilistic manner with simple yet effective approaches. Clustering-based generalization [20] firstly clusters nodes into groups and next anonymizes a sub-network into a super-node without individual node's specific information. This approach can be effective against de-anonymization. However, it has the loss of individual information as well as scale information, which may dramatically change the results in social network analysis.

8.2 De-anonymization Algorithms

De-anonymization is the reverse process of anonymization, and researchers have been studying it in different methods [1, 5, 10, 11, 13, 16, 17, 19]. It often appears in reality as part of an attack to leak user privacy [8].

Backstrom et al. [1] proposed a family of de-anonymizations so that it is possible for an adversary to learn whether edges exist or not between specific targeted pairs of nodes. The weakness is that the algorithm is vulnerable when the networks are modified before publishing, although it works fine for naive anonymization where node numbers are switched.

Narayanan and Shmatikov [17] presented a framework for analyzing privacy and proposed a de-anonymizing algorithm. The algorithm is based on the network topology only and is relatively robust to noise and most defenses. It requires a few seed mappings and propagates to the whole networks. However, the quality of seed mappings has significant influence on whether the attack will succeed. Later Narayanan et al. [16] introduced a simulated annealing-based weight network matching algorithm for finding good initial seed mappings for de-anonymization. Yartseva and Grossglauser [19] used network percolation to propose a seed-based network matching algorithm. Kazemi et al. [10] proposed a scalable network matching algorithm using smaller seed mappings to match a pair of networks, with a small increase in matching errors. Korula and Lattanzi [11] applied network percolation to power-distributed networks and had an improved performance for real-world social network matching.

Fu et al. [4] proposed a seedless algorithm for social network de-anonymization. The algorithm first computes each node pair's similarity iteratively based on maximum matching and then matches nodes according to the node pair similarity scores from high to low. Our RoleMatch follows the same de-anonymization framework, but RoleMatch applies a new similarity measure and a new node matching algorithm.

8.3 Node Similarity Measure

Node similarity is a basic metric for network analysis. So far, many different similarity measures have been proposed.

Henderson et al. [6] proposed an algorithm that recursively combines local features with neighborhood features to produce regional features, and then used these regional features to compute node similarity for de-anonymization. These regional features can effectively narrow the range of possible corresponding nodes, but with no evidence of the ability to find the real identity with the most similar pairs.

The famous "SimRank" measure by Jeh and Widom [7] provides node similarity measure within one network, which is inapplicable in de-anonymization problems. Blondel et al. [2] proposed a cross-network node similarity measure by summing up similarity scores of neighbors of two nodes, which is much like the two-network version of "SimRank." Fu et al. [4] proposed a two-network node similarity measure by iteratively matching neighbors with top similarity scores for two nodes, which is an improved measure compared to simple transplanting "SimRank" to cross-network computation. This approach works for nodes with large degrees but since it lacks normalization, for small-degree nodes the similarity scores are usually too small to be meaningful. Jing et al. [9] proposed a node similarity measure "RoleSim" with normalization for node within a single network. It can be a good depiction of nodes' structural information, but the definition and computation method limits it to a single-network measure. Our new cross-network node similarity measure is designed on the basis of all these measures.

9 Conclusions

Social network de-anonymization is a popular approach to test the strength of anonymization algorithms. With the help of a good de-anonymization solution, we can guide companies to design a much better anonymization approach to protect the user's privacy. In this paper, we developed a fast seedless de-anonymization algorithm, named RoleMatch. RoleMatch de-anonymizes networks only based on the structural information. Thanks to the new similarity measure, RoleSim++, it can compute the node similarity in high precision. Moreover, during the node matching phase, besides the node similarity, RoleMatch also uses the neighborhood information to improve the mapping results. The comprehensive experimental results have demonstrated the advantages of RoleMatch compared with previous works. Besides the algorithm itself, the performance of de-anonymization is also related to the properties of network. We will study such relationship in the future.

Acknowledgements This research is funded by the National Natural Science Foundation of China (No. 61702015).

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

Appendix

The Proof of Lemma 1

Proof We prove the lemma by induction on the number of iterations.

1. *Base Case* Note that $\mathrm{Sim}^0(u,v)=1$, then $\mathrm{Sim}^1(u,v)=$

$$(1-\beta)\frac{\min\{|N_1^{\text{out}}(u)|,|N_2^{\text{out}}(v)|\}+\min\{|N_1^{\text{in}}(u)|,|N_2^{\text{in}}(v)|\}}{\Delta^{\text{out}}(u,v)+\Delta^{\text{in}}(u,v)} + \beta.$$

   Thus, $\mathrm{Sim}^1(u,v)\le 1=\mathrm{Sim}^0(u,v)$.
2. *Induction Step* Assume that in the kth iteration, the matching $\mathcal{M}^{\text{out}}(u,v)$ of $N_1^{\text{out}}(u), N_2^{\text{out}}(v)$ is

$$\Gamma_k^{\text{out}}(u,v) = \sum_{(x,y)\in\mathcal{M}^{\text{out}}(u,v)} \text{Sim}^{k-1}(x,y).$$

Similarly,

$$\Gamma_k^{in}(u,v) = \sum_{(x,y)\in\mathcal{M}^{in}(u,v)} \text{Sim}^{k-1}(x,y).$$

Then,

$$\text{Sim}^k(u,v) = (1-\beta)\frac{\Gamma_k^{\text{out}}(u,v)+\Gamma_k^{\text{in}}(u,v)}{\Delta^{\text{out}}(u,v)+\Delta^{\text{in}}(u,v)} + \beta \tag{2}$$

$$= (1-\beta)\frac{\sum_{(x,y)\in\mathcal{M}^{\text{out}}(u,v)} \text{Sim}^{k-1}(x,y) + \sum_{(x,y)\in\mathcal{M}^{in}(u,v)} \text{Sim}^{k-1}(x,y)}{\Delta^{\text{out}}(u,v)+\Delta^{\text{in}}(u,v)} + \beta \tag{3}$$

$$\leq (1-\beta)\frac{\sum_{(x,y)\in\mathcal{M}^{\text{out}}(u,v)} \text{Sim}^{k-2}(x,y) + \sum_{(x,y)\in\mathcal{M}^{in}(u,v)} \text{Sim}^{k-2}(x,y)}{\Delta^{\text{out}}(u,v)+\Delta^{\text{in}}(u,v)} + \beta \tag{4}$$

$$\leq (1-\beta)\frac{\Gamma_{k-1}^{\text{out}}(u,v)+\Gamma_{k-1}^{\text{in}}(u,v)}{\Delta^{\text{out}}(u,v)+\Delta^{\text{in}}(u,v)} + \beta \tag{5}$$

$$= \text{Sim}^{k-1}(u,v). \tag{6}$$

The transition (4) is due to the induction, and the transition (5) is due to the definition of function Γ.

Thus, $\text{Sim}^{k-1}(u,v) \leq \text{Sim}^k(u,v)$ holds for all (u,v).

Conclusion $\text{Sim}^k(u,v) \geq \text{Sim}^{k+1}(u,v)$ is true for all $k \in \mathbb{N}_0$. □

The Proof of Proposition 2

Proof First, when the similarity $\text{Sim}(u,v)$ is converged, we use $M^{\text{out}}(u,v)$ to denote the best matching with maximum similarity score between $N_1^{\text{out}}(u)$ and $N_2^{\text{out}}(v)$. So does the $M^{\text{in}}(u,v)$. Then let $\delta^{\text{out}}(u,v) = min\{|N_1^{\text{out}}(u)|, |N_2^{\text{out}}(v)|\}$, and $\delta^{\text{in}}(u,v) = min\{|N_1^{\text{in}}(u)|, |N_2^{\text{in}}(v)|\}$, it is easy to figure out $\frac{\delta^{\text{out}}(u,v)+\delta^{\text{in}}(u,v)}{\Delta^{\text{out}}(u,v)+\Delta^{\text{in}}(u,v)} \leq 1$.

In the kth iteration, let $M_k^{\text{out}}(u,v)$ be the best matchings with maximum similarity score between $N_1^{\text{out}}(u)$ and $N_2^{\text{out}}(v)$, then we have

$$\begin{aligned}
&\Gamma_k^{\text{out}}(u,v) - \Gamma^{\text{out}}(u,v)\\
&\quad= \sum_{(x,y)\in M_k^{\text{out}}(u,v)} \text{Sim}_{k-1}(x,y) - \sum_{(x',y')\in M^{\text{out}}(u,v)} \text{Sim}(x',y')\\
&\quad\leq \sum_{(x,y)\in M_k^{\text{out}}(u,v)} (\text{Sim}_{k-1}(x,y) - \text{Sim}(x,y))\\
&\qquad \because \text{When converged, } M^{\text{out}} \text{ is better than } M_k^{\text{out}}.\\
&\quad\leq \delta^{\text{out}}(u,v) max_{(x,y)\in M_k^{\text{out}}(u,v)}\{(\text{Sim}_{k-1}(x,y) - \text{Sim}(x,y))\}.
\end{aligned} \tag{7}$$

Similarly, we have

$$\begin{aligned}
&\Gamma_k^{\text{in}}(u,v) - \Gamma^{\text{in}}(u,v)\\
&\quad= \sum_{(x,y)\in M_k^{\text{in}}(u,v)} \text{Sim}_{k-1}(x,y)\\
&\qquad - \sum_{(x',y')\in M^{\text{in}}(u,v)} \text{Sim}(x',y')\\
&\quad\leq \delta^{\text{in}}(u,v)\max_{(x,y)\in M_k^{\text{in}}(u,v)}\\
&\qquad \{(\text{Sim}_{k-1}(x,y) - \text{Sim}(x,y))\}.
\end{aligned} \tag{8}$$

Next we prove the proposition as follows,

$$\begin{aligned}
\epsilon^k(u,v) &= \text{Sim}^k(u,v) - \text{Sim}(u,v)\\
&= (1-\beta)\frac{(\Gamma_k^{\text{out}}(u,v)+\Gamma_k^{\text{in}}(u,v)-\Gamma^{\text{out}}(u,v)-\Gamma^{\text{in}}(u,v))}{\Delta^{\text{out}}(u,v)+\Delta^{\text{in}}(u,v)}\\
&\leq (1-\beta)\frac{1}{\Delta^{\text{out}}(u,v)+\Delta^{\text{in}}(u,v)}\\
&\quad (\delta^{\text{out}}(u,v)\max_{(x,y)\in M_k^{\text{out}}(u,v)}\{(\text{Sim}_{k-1}(x,y)-\text{Sim}(x,y))\}\\
&\quad + \delta^{\text{in}}(u,v)\max_{(x,y)\in M_k^{\text{in}}(u,v)}\{(\text{Sim}_{k-1}(x,y)-\text{Sim}(x,y))\})\\
&\leq (1-\beta)\frac{\delta^{\text{out}}(u,v)+\delta^{\text{in}}(u,v)}{\Delta^{\text{out}}(u,v)+\Delta^{\text{in}}(u,v)}\\
&\quad \max\{\max_{(x,y)\in M_k^{\text{out}}(u,v)}(\text{Sim}_{k-1}(x,y)-\text{Sim}(x,y)),\\
&\quad \max_{(x,y)\in M_k^{\text{in}}(u,v)}(\text{Sim}_{k-1}(x,y)-\text{Sim}(x,y))\}.
\end{aligned}$$

Assuming that $\text{Sim}_{k-1}(x_{k-1}^*, y_{k-1}^*) - \text{Sim}(x_{k-1}^*, y_{k-1}^*) = \max\{\max_{(x,y)\in M_k^{\text{out}}(u,v)}\{(\text{Sim}_{k-1}(x,y)-\text{Sim}(x,y))\}, \max_{(x,y)\in M_k^{\text{in}}(u,v)}\{(\text{Sim}_{k-1}(x,y)-\text{Sim}(x,y))\}\}$, then

$$\begin{aligned}
\epsilon^k(u,v) &= \text{Sim}^k(u,v) - \text{Sim}(u,v)\\
&\leq (1-\beta)\frac{\delta^{\text{out}}(u,v)+\delta^{\text{in}}(u,v)}{\Delta^{\text{out}}(u,v)+\Delta^{\text{in}}(u,v)}\\
&\quad \{(\text{Sim}_{k-1}(x_{k-1}^*, y_{k-1}^*) - \text{Sim}(x_{k-1}^*, y_{k-1}^*))\}\\
&\leq (1-\beta)(\text{Sim}_{k-1}(x_{k-1}^*, y_{k-1}^*) - \text{Sim}(x_{k-1}^*, y_{k-1}^*))\\
&\dots\\
&\leq (1-\beta)^k(\text{Sim}_0(x_0^*, y_0^*) - \text{Sim}(x_0^*, y_0^*))\\
&\leq (1-\beta)^{k+1} \quad \because \text{Sim}(x_0^*, y_0^*) \geq \beta.
\end{aligned}$$

Conclusion $\epsilon^k(u,v) \leq (1-\beta)^{k+1}$ holds for all $k \in \mathbb{N}_0$. □

The Proof of Proposition 3

Proof We prove this proposition by induction on number of iterations.

1. *Base Case* Initially we have $\text{Sim}^0(u,v) = 1 = c_0$.

2. *Induction Step* In the kth iteration, it is obvious that u and v have a proportion $(1-\delta)$ of common neighbors, whose similarity score is no less than c_{k-1}. So there exist matchings $M^{\text{out}}(u,v)$ and $M^{\text{in}}(u,v)$ between $N_1(u)$ and $N_2(v)$, so that $\Gamma^{\text{out}}(u,v)+\Gamma^{\text{in}}(u,v) \geq c_{k-1}(1-\delta)(\Delta^{\text{out}}(u,v)+\Delta^{\text{in}}(u,v))$.

Thus,

$$\begin{aligned}
\text{Sim}^k(u,v) &= (1-\beta)\frac{\Gamma_k^{\text{out}}(u,v)+\Gamma_k^{\text{in}}(u,v)}{\Delta^{\text{out}}(u,v)+\Delta^{\text{in}}(u,v)}+\beta \\
&\geq (1-\beta)(1-\delta)c_{k-1}+\beta \\
&= a\left(a^{k-1}+\beta\frac{1-a^{k-1}}{1-a}\right)+\beta \\
&= a^k+\beta\frac{1-a^k}{1-a} \\
&= c_k.
\end{aligned}$$

Conclusion $\text{Sim}^k(u,v) \geq c_k$ is true for all $k \in \mathbb{N}_0$. □

References

1. Backstrom L, Dwork C, Kleinberg J (2007) Wherefore art thou r3579x? Anonymized social networks, hidden patterns, and structural steganography. In: WWW, pp 181–190
2. Blondel VD, Gajardo A, Heymans M, Senellart P, Van Dooren P (2004) A measure of similarity between graph vertices: applications to synonym extraction and web searching. SIAM Rev 46(4):647–666
3. Bonchi F, Gionis A, Tassa T (2014) Identity obfuscation in graphs through the information theoretic lens. Inf Sci 275:232–256
4. Fu H, Zhang A, Xie X (2015) Effective social graph deanonymization based on graph structure and descriptive information. ACM Trans Intell Syst Technol 6(4):49
5. Gulyás GG, Simon B, Imre S (2016) An efficient and robust social network de-anonymization attack. In: Proceedings of workshop on privacy in the electronic society, WPES '16, pp 1–11
6. Henderson K, Gallagher B, Li L, Akoglu L, Eliassi-Rad T, Tong H, Faloutsos C (2011) It's who you know: graph mining using recursive structural features. In: KDD, pp 663–671
7. Jeh G, Widom J (2002) SimRank: a measure of structural-context similarity. In: KDD, pp 538–543
8. Ji S, Li W, Srivatsa M, He JS, Beyah R (2016) General graph data de-anonymization: from mobility traces to social networks. ACM Trans Inf Syst Secur 18(4):12:1–12:29
9. Jin R, Lee VE, Hong H (2011) Axiomatic ranking of network role similarity. In: KDD, pp 922–930
10. Kazemi E, Hassani SH, Grossglauser M (2015) Growing a graph matching from a handful of seeds. Proc VLDB Endow 8(10):1010–1021
11. Korula N, Lattanzi S (2014) An efficient reconciliation algorithm for social networks. Proc VLDB Endow 7(5):377–388
12. Kuhn HW (1955) The hungarian method for the assignment problem. Naval Res Logist Q 2(1–2):83–97
13. Li H, Chen Q, Zhu H, Ma D (2017) Hybrid de-anonymization across real-world heterogeneous social networks. In: Proceedings of the ACM turing 50th celebration conference—China, ACM TUR-C '17, pp 33:1–33:7
14. Li H, Zhu S, Du X, Liang X Shen (2018) Privacy leakage of location sharing in mobile social networks: attacks and defense. IEEE Trans Dependable Secur Comput 15(4):646–660
15. Liu K, Terzi E (2008) Towards identity anonymization on graphs. In: SIGMOD, pp 93–106
16. Narayanan A, Shi E, Rubinstein BI (2011) Link prediction by de-anonymization: how we won the kaggle social network challenge. In: IJCNN, pp 1825–1834
17. Narayanan A, Shmatikov V (2009) De-anonymizing social networks. In: ISSP, pp 173–187
18. Wu X, Ying X, Liu K, Chen L (2010) A survey of privacy-preservation of graphs and social networks. In: Managing and mining graph data, pp 421–453
19. Yartseva L, Grossglauser M (2013) On the performance of percolation graph matching. In: Proceedings of the first ACM conference on online social networks, pp 119–130
20. Zheleva E, Getoor L (2008) Preserving the privacy of sensitive relationships in graph data. In: KDD, pp 153–171
21. Zhou B, Pei J (2008) Preserving privacy in social networks against neighborhood attacks. In: ICDE, pp 506–515