De-anonymization of Heterogeneous Random Graphs in Quasilinear Time

(extended abstract)

Karl Bringmann[1], Tobias Friedrich[2], and Anton Krohmer[2]

[1] Max Planck Institute for Informatics, Saarbrücken, Germany
[2] Friedrich-Schiller-Universität Jena, Germany

Abstract There are hundreds of online social networks with billions of users in total. Many such networks publicly release structural information, with all personal information removed. Empirical studies have shown, however, that this provides a false sense of privacy — it is possible to identify almost all users that appear in two such anonymized network as long as a few initial mappings are known.
We analyze this problem theoretically by reconciling two versions of an artificial power-law network arising from independent subsampling of vertices and edges. We present a new algorithm that identifies most vertices and makes no wrong identifications with high probability. The number of vertices matched is shown to be asymptotically optimal. For an n-vertex graph, our algorithm uses n^{ε} seed nodes (for an arbitrarily small ε) and runs in quasilinear time. This improves previous theoretical results which need $\Theta(n)$ seed nodes and have runtimes of order $n^{1+\Omega(1)}$. Additionally, the applicability of our algorithm is studied experimentally on different networks.

1 Introduction

Imagine owning a large social network G_1 (like Facebook or Google+), and a competitor publishes an anonymized version of its own social network G_2, i.e. the graph structure without any additional labeling. This can happen on purpose or indirectly by APIs which are permitted to access the competitor's network or special access granted to advertising partners. If we identify vertices that are the same in both networks, we effectively deanonymize G_2 and gain new information about G_1, as there are connections in G_2 that do not exist in our social network G_1. This is valuable information for e.g. suggesting friends who are not yet connected in one of the networks. In this paper, we approach this social network reconciliation problem from an algorithm theory point of view.

Model. We model the above situation by assuming the existence of an underlying "real" social network $G = (V, E)$, which encodes whether two people know each other in the real world. We assume that G is a power-law network described by a Chung-Lu random graph [1, 4, 5] with n vertices. Then we assume the online

social networks G_1 and G_2 to be subsets of G: Every node of G exists in G_i independently with probability q_i, and every edge of G exists in G_i independently with probability p_i. Additionally, we randomly permute the graphs G_1 and G_2. We assume that there is a set of seed nodes $V_I \subseteq V$ which are known to match between G_1 and G_2 because they e.g. are persons of public interest. The algorithmic problem now is to identify as many vertices as possible from the given graphs G_1 and G_2 without making any wrong identifications (with high probability). We call a vertex *identifiable* if it survives in both graphs G_1 and G_2.

Theoretical results. We present an algorithm with the following guarantees. Here we let δ be the (expected) average degree of the graph G. See Section 2 for the technical assumptions about the parameters of the Chung-Lu random graph G and the parameters of the subsampling process.

Theorem 1. *Assume we are given the n^ε largest identifiable vertices as seed nodes for an arbitrary constant $\varepsilon > 0$. There is an algorithm that with high probability*[1] *makes no wrong identifications and successfully matches a fraction of $1 - \exp(-\Omega(p_1 p_2 q_1 q_2 \delta))$ of the identifiable vertices.*[2] *The algorithm runs in expected quasilinear runtime $\mathcal{O}(\delta n \log(n) / \min\{p_1, p_2\}^{\mathcal{O}(1/\varepsilon)})$.*

In the full version, we also show that this fraction of identified vertices is asymptotically optimal, since intuitively an $\exp(-\mathcal{O}(p_1 p_2 q_1 q_2 \delta))$ fraction of the vertices does not have any common neighbors in the two social networks. For constant p_1, p_2, q_1, q_2, the runtime is $\mathcal{O}(\delta n \log(n))$, which is within a factor $\log(n)$ of the expected number of edges $\Theta(\delta n)$ of G. Thus, our algorithm is the first with *quasilinear runtime.* This is crucial for handling large graphs. The best previous algorithms have a runtime of order $\mathcal{O}(\delta n \, \Delta^2)$ [9] or $\mathcal{O}(\delta n \, \Delta \log(\Delta))$ [7], where Δ is the maximum degree, which is typically of size $n^{\Omega(1)}$. Our approach also only needs n^ε seed nodes — previous algorithms with proven runtime and quality use at least $\Theta(n)$ seeds [7]. We remark that our algorithm is also successful with only $\mathcal{O}(\log(n)/p_1 p_2)$ seed nodes, but runs in quadratic time in this case.

Empirical results. We implemented a variant of our algorithm and applied it to different sets of networks. We match $\geqslant 89\%$ of the vertices in Chung-Lu graphs, preferential attachment graphs (PA), affiliation networks, and also subsampled real-world networks (Facebook, Orkut). All runs took less than 60 minutes on a single core, where previous results used compute clusters for an unreported amount of time [7]. In all cases, we need $\leqslant 0.03\%$ seed nodes to bootstrap our algorithm. This indicates that our approach translates to a wide variety of scale-free networks, even though we formally prove it on the Chung-Lu model.

Algorithm description. Starting with the seed nodes, we identify the remaining vertices by their *signatures*, an idea used in many algorithms for graph isomor-

[1] Throughout the paper, we say that a bound holds *with high probability* (w.h.p.) if it holds with probability at least $1 - n^{-c}$ for some $c > 0$.

[2] In the whole paper $\mathcal{O}(\cdot)$ and $\Omega(\cdot)$ hide any constant depending only on the power law exponent β of G. We always assume $2 < \beta < 3$.

phism. However, we have to cope with the additional complexity of the neighborhoods of identical vertices not being equal. We identify vertices when their signatures are strongly overlapping, and show an easy criterion for deciding whether two signatures stem from identical vertices. Using this criterion we make no errors with high probability and identify a large constant fraction of the vertices. We achieve a quasilinear runtime by locality sensitive hashing [6], which reduces the number of comparisons.

Applications. Anonymous copies of some social networks are available online. Several experimental papers describe how to find mappings between two online social networks [2, 9, 13, 14]. Most of them use metadata like browser history [13], group memberships [15], writing style [11], semantic features of user aliases [10], or artificially added subgraphs [2]. The only theoretical result on this subject is by Korula and Lattanzi [7]. They identify 97% of the nodes on subsampled ($p_1, p_2 \geqslant \sqrt{22/\delta}$, $q_1 = q_2 = 1$) preferential attachment graphs [3], but need a linear amount of seed nodes and substantially more computing resources.

2 Preliminaries

Graph model. The model has two adjustable parameters: the exponent of the scale-free network β and the average degree δ. Depending on these two parameters, each node i has a weight w_i. For $n \in \mathbb{N}$ and weight distribution $\mathbf{w} = (w_1, \dots, w_n) \in \mathbb{R}^n_{\geqslant 0}$ the Chung-Lu graph Chung-Lu$(n, \mathbf{w})$ is a graph on vertex set $V = [n]$ that contains each edge $\{u, v\}$, $u \neq v \in V$, with probability $p_{u,v} := \min\{w_u w_v / W, 1\}$, where $W := \sum_{v \in V} w_v$.

In order to simplify the presentation, we use a simple explicit weight distribution $w_i = \delta(n/i)^{1/(\beta-1)}$. Then $W = (1 + o(1))\frac{\beta-1}{\beta-2}\delta n = \Theta(\delta n)$, the expected average degree is $(1+o(1))\frac{2(\beta-1)}{\beta-2}\delta = \Theta(\delta)$, and we get a power law with exponent β [12]. We note that most of our results generalize to other weight distributions and even to weights drawn at random from "nice" distributions. We assume constant $2 < \beta < 3$, as real-world social networks have been observed to fulfill this. Moreover, we require $\delta \leqslant n^{o(1)}$, $p_1, p_2 \geqslant n^{-o(1)}$, and $q_1, q_2 = \Theta(1)$. We also assume that we know a lower bound for q_1, q_2, so that we know a constant factor approximation of n. For the sake of readability we even assume that we know n exactly. Finally, we require that $p_1 p_2 q_1 q_2 \delta$ is at least a sufficiently large constant (depending only on β).

De-anonymization. In our problem we have an underlying graph $G =$ Chung-Lu$(n, \mathbf{w})$ as defined above. This graph gets subsampled twice to generate two subgraphs: We put each node $v \in V := V(G)$ into V_1 independently with probability q_1. Then we put each edge $e \in E \cap \binom{V_1}{2}$ into E_1 independently with probability p_1 to form a graph $G_1 = (V_1, E_1)$. Now we randomly permute G_1 to obtain a graph $\widetilde{G}_1$. We repeat this process with independent choices (and probabilities q_2, p_2) to form G_2 and $\widetilde{G}_2$.

We call two nodes $\widetilde{v}_i$ in $\widetilde{G}_i$, $i \in \{1, 2\}$ *identical*, if they stem from the same node $v \in V$. The *identifiable* nodes are $V_\cap := V_1 \cap V_2$.

The input for the de-anonymization problem is $(\widetilde{G}_1, \widetilde{G}_2)$ and the task is to report pairs of vertices ("identified vertices") such that with high probability every identified pair is identical. We want to maximize the number of identified pairs. Note that the algorithm gets the randomly permuted graphs $\widetilde{G}_i$, but in the analysis we usually talk about the graphs G_i for the sake of readability. We write $\deg_i(v)$ for the degree of vertex $v \in V_i$ in G_i and $N_i(v)$ for its neighborhood in graph G_i, $i \in \{1, 2\}$.

3 Estimating Weights and Edge Probabilities

In this section we show how to compute upper and lower bounds for the weight w_v of any vertex v based on the degree $\deg_i(v)$, $i \in \{1, 2\}$. This also yields bounds for the edge probabilities $p_{u,v}$ for any vertices u and v. These bounds hold with high probability. Then we argue that our subsequent de-anonymization algorithms can use these computed bounds and still assume that all edges of G_i and G were sampled independently as if these graphs were not looked at before (where we used G_i as a short term for both graphs G_i, $i \in \{1, 2\}$).

For the sake of readability we assume that the parameters n, β, p_1, and p_2 are known to the algorithm. However, it would be easy to also estimate these parameters with small error, and run our subsequent algorithms with these approximations. For a sketch of this, we note that we can estimate β from the degree distributions in G_i, similar to what we do for individual weights in this section. Moreover, we can estimate p_2 (and p_1, respectively) by dividing the number of edges that appear in $G_1[V_I] \cap G_2[V_I]$ by the number of edges in $G_2[V_I]$ ($G_1[V_I]$).

Afterwards we can run the method presented in this section to estimate the individual weights w_v and W. Additionally, one could estimate δ (e.g. from W and β) and q_1/q_2 (e.g. from $|V_1|/|V_2|$), but our algorithms do not need them. Note that from our model it is hard to estimate q_i itself.

The degree of each vertex v in G_i is composed of a random decision for each other node u, namely whether it is connected to v in the original graph G and a random decision whether this edge is present in the subsampled graph. In total,

$$\deg_i(v) \sim \sum_{u \in V \setminus \{v\}} \mathrm{Ber}\left(p_i q_i \cdot \min\left\{\frac{w_v w_u}{W}, 1\right\}\right),$$

if node v survives in G_i. By a Chernoff bound, we see that this degree is concentrated. This allows to compute intervals for the weights w_v for all v in G_i.

Lemma 1. *Let $i \in \{1, 2\}$. Given $\deg_i(v)$ (and p_i and an approximation of n) we can compute $0 \leqslant \underline{w}_v \leqslant \overline{w}_v$ such that w.h.p. for all $v \in V$ we have*

1. $\underline{w}_v \leqslant q_i w_v \leqslant \overline{w}_v$,
2. $\overline{w}_v \leqslant \mathcal{O}\big(q_i w_v + \frac{1}{p_i} \log n\big)$, *and*
3. $\underline{w}_v \geqslant \Omega(q_i w_v) - \mathcal{O}\big(\frac{1}{p_i} \log n\big)$.

In a similar fashion, we can compute a bound on $q_i^2 \cdot W$.

Lemma 2. *Let $i \in \{1,2\}$. Given G_i, we can compute $\underline{W}$ such that $\underline{W} \leqslant q_i^2 W \leqslant (1+o(1))\underline{W}$ holds with high probability.*

The proof of Lemma 1 and most other proofs in the remainder can be found in the full version. Plugging the estimated weights into the edge probability formula allows us to compute bounds on the edge probabilities. It is worth noting that although the estimations on $\overline{w}_v$ and $\underline{W}$ give a result depending on q_i, the computed upper bound $\overline{p}_{uv}$ on the edge probabilities is oblivious to q_i.

Corollary 1. *For any $u, v \in V_i$ we can compute bounds $\overline{p}_{u,v} := \min\{\overline{w}_u \overline{w}_v / \underline{W}, 1\}$ such that w.h.p. we have*

$$p_{u,v} \leqslant \overline{p}_{u,v} \leqslant \mathcal{O}\Big(p_{u,v}\Big(1+\frac{\log n}{p_i q_i w_u}\Big)\Big(1+\frac{\log n}{p_i q_i w_v}\Big)\Big).$$

In particular, for $w_u, w_v = \Omega(\frac{1}{p_i q_i}\log n)$ we have $\overline{p}_{u,v} = \mathcal{O}(p_{u,v})$.

Corollary 1 allows to compute estimations for all edge probabilities with certain guarantees that hold with high probability. We want to use these estimations in the subsequent algorithms without losing the independence of the edges, i.e., in the subsequent algorithms we want to assume that the (edges of the) graphs G_i were not revealed yet, although we already computed bounds on the weights based on the degrees in G_i. In order to see that this might be a problem, assume that throughout a proof we reveal edges of a node v. However, once we have seen $\deg_i(v)$ edges, we know that there can be no other edge anymore, which violates our intuition of having independent edges.

To solve this technical problem, we model our weight estimation method as an *adaptive adversary*, which knows the parameters $p_1, p_2, q_1, q_2, w_1, \ldots, w_n$, G_1 and G_2, and reports estimations $\overline{p}_{u,v}$ for all $u, v \in V$ that fulfill the guarantees in Corollary 1 w.h.p. (over the randomness of the instance generation). The subsequent de-anonymization algorithms are then designed such that they assume to get edge probability estimations by our above method (or an adversary) that fulfill the said guarantees but are otherwise arbitrary. Then they may still assume that the random graphs G_i are not revealed. The details of this can be found in the full version of this paper.

4 Matching Phase

In the matching phase we assume that we know the identity of some vertices $V_I \subseteq V$ containing the $h = \Omega(\log(n)/p_1 p_2)$ highest weight nodes (that survive in both G_1, G_2), and show how to identify most of the remaining vertices based on these initial nodes. Observe that the adaptive adversary model allows us to assume that all edges are independently present with their respective probability $p_{u,v}$.

4.1 The Y-Test

Denote by V_I the thus far identified vertices. Then for every unidentified vertex v in G_i we consider its *signature* $S_i^v := N_i(v) \cap V_I$. Unlike in the Graph Isomorphism

problem, in our case signatures of identical vertices are not equal. However, for identical vertices the signatures S_1^v, S_2^v should be similar sets, while for non-identical vertices $u \neq v$ the signatures S_1^u, S_2^v should have small intersection. One contribution of our work is the test presented in this section, which allows to check whether two nodes are identical based on their signatures. This test never identifies two non-identical vertices (w.h.p.) and it identifies most vertices once sufficiently many of their neighbors are identified.

Let $v_1, v_2 \in V \setminus V_I$ and $u \in V_I$. Consider all possibilities of the edges $\{v_1, u\} \in E_1$ and $\{v_2, u\} \in E_2$ being present or not. We denote by A_u the event that both of these edges are present, by B_u^i the events that exactly one edge is present in G_i, and by C_u the remaining case. Based on these cases we now define

$$Y_u = Y_u^{v_1,v_2} := \begin{cases} \frac{1}{2\overline{p}_{v_1,u}}, & \text{if } u \in S_1^{v_1} \cap S_2^{v_2} \quad (A_u) \\ 1 - \frac{p_2}{2}, & \text{if } u \in S_1^{v_1} \text{ and } u \notin S_2^{v_2} \quad (B_u^1) \\ 1 - \frac{p_1}{2}, & \text{if } u \notin S_1^{v_1} \text{ and } u \in S_2^{v_2} \quad (B_u^2) \\ 1, & \text{otherwise } (C_u). \end{cases}$$

and $Y := \prod_{u \in V_I} Y_u$. Intuitively, we tailored Y_u to be the "estimated probability" that $v_1 = v_2$ based on the evidence of identified node u. The smaller the degree of an identified node u, the lower is the probability that two different nodes $v_1 \neq v_2$ both connect to u. Conversely, the larger p_i is, the likelier it is that two equal nodes $v_1 = v_2$ attain non-overlapping signatures. Note that when $v_1 = v_2$ we have $\overline{p}_{v_1,u} \approx \overline{p}_{v_2,u}$, so in the case (A_u) having one of the estimates turns out to be sufficient. The technical factor $1/2$ is needed later for some tail bounds.

We claim that Y is typically small for non-identical $v_1 \neq v_2$ and can be large (if V_I contains sufficiently many neighbors of v_1) if $v_1 = v_2$. In particular, we can test whether $v_1 = v_2$ by testing $Y > n^c$ for some appropriate constant $c > 0$. We call this the Y-test. This intuition is proven by the following lemmas. First we show that Y is not too large if $v_1 \neq v_2$ (w.h.p.). To this end, we verify $\mathbb{E}[Y_u] \leqslant 1$, then the statement follows from independence of the edges and Markov's inequality.

Lemma 3. *For any $v_1 \neq v_2 \in V \setminus V_I$ and $t > 0$ we have $\Pr[Y > t] \leqslant 1/t$.*

The next lemma can be used to show that our test allows to identify the two copies of v if we have already identified enough low-degree neighbors of v. We call the high-degree neighbors u "bad nodes" as they result in an estimated connection probability of $\overline{p}_{v,u} = 1$.

Lemma 4. *Let V_I be any set of identified vertices, and consider an unidentified $v \in V_\cap \setminus V_I$. Let $B \subseteq V_I$ ("bad nodes") be the vertices u with $p_{u,v} \geqslant b > 0$, b being a sufficiently small constant. Assume that for $c > 0$ we have*

$$\sum_{u \in V_I} p_{u,v} \geqslant \Omega\big(\tfrac{1}{p_1 p_2} c \log n + |B|\big)$$

with a sufficiently large hidden constant. Then we have $\Pr[Y^{v,v} > n^c] \geqslant 1 - n^{-c}$.

For a vertex v with small weight $w_v = n^{o(1)}$ the above lemma does not apply and we have to take a closer look at $Y^{v,v}$.

Lemma 5. *Let $c > 0$ and consider an unidentified vertex $v \in V_\cap \setminus V_I$ with $w_v \leqslant n^{o(1)}$. Let $T \subseteq V_I$ be a set of identified vertices with $p_{u,v} = \Theta(\varepsilon)$ for all $u \in T$ and some $\varepsilon > 0$. Assume that $\mu := p_1 p_2 \varepsilon |T|$ is at least a sufficiently large constant (depending only on c and β). Then we have*

$$\Pr[Y^{v,v} > n^c] \geqslant 1 - n^{-c} - \exp(-\Omega(\mu)).$$

4.2 The Algorithm

We use the test developed in the last section as follows. As we build an algorithm that w.h.p. never identifies non-identical vertices, we can again write this algorithms in terms of the graphs G_1, G_2, but it can easily be translated to the randomly permuted graphs $\widetilde{G}_1, \widetilde{G}_2$.

Our algorithm gets as input the graphs G_1, G_2 and an initial set V_I of identified vertices containing the h highest weight vertices. Then in every round the algorithm compares all pairs v_1, v_2 of unidentified vertices. One comparison consists of a Y-test, i.e., we compute Y^{v_1,v_2} and test whether it is at least n^c, where $c > 0$ a constant. If this is the case, then we identify v_1 and v_2. The algorithm terminates after the first round in which no new vertex is identified.

Note that this algorithm is oblivious to the q_i's, as it only considers edges to nodes that are already identified, and thus survive in both subsampled graphs.

We will see that it suffices to run this algorithm for $\mathcal{O}(\log n)$ rounds to identify most of the vertices. As Y^{v_1,v_2} can be computed in time $\mathcal{O}(\deg_1(v_1) + \deg_2(v_2))$, the immediate runtime of this algorithm is $\mathcal{O}(nm \log n)$, where m is the total number of edges in G_1 and G_2, which is $\mathcal{O}(\delta n)$ with high probability. We will see in Section 5 how to decrease this to quasilinear runtime.

Algorithm 1 De-anonymization using Y-tests

Input: graphs G_1, G_2, identified vertices $V_I \supseteq \{1, \dots, h\}$
  for $r = 1, 2, \dots, \mathcal{O}(\log n)$ **do**
    for all $v_1, v_2 \in V \setminus V_I$ **do**
      if $Y^{v_1,v_2} > n^c$ **then**
        $V_I := V_I \cup \{v_1, v_2\}$. ▷ We identified $v_1 = v_2$

Using Lemma 3 it is easy to see that Algorithm 1 never identifies any non-identical vertices. Note that choosing $c > 2$ yields error probability $o(1)$.

Lemma 6. *Algorithm 1 does not identify any two non-identical vertices with probability at least $1 - \mathcal{O}(n^{2-c} \log n)$.*

4.3 Quality Analysis

It remains to show that the algorithm identifies most vertices. We do this by examining a particular order on the vertices in which they can be identified. For

this, let $L_j := \{2^{j-1}, \dots, 2^j - 1\}$, $j = 1, \dots, \log n$, be the j-th layer of vertices, and let $\tilde{L}_j \subseteq L_j$ be the set of vertices from layer j that survive in both graphs. If $|L_j| = \Omega(\log n)$, then w.h.p. by a Chernoff bound we have $|\tilde{L}_j| \geqslant \Omega(q_1 q_2 |L_j|)$.

For the analysis, choose k such that the estimated edge probability $\overline{p}_{v,h}$ of every vertex $v \in L_k$ with vertex h (the h-th highest weight vertex) is at most a sufficiently small constant, and k is minimal with this property. We can compute that $|L_k| = \Omega(n^{3-\beta}/\log n)$, meaning that $|\tilde{L}_k| = \Theta(q_1 q_2 |L_k|)$. In the first step of the analysis of our algorithm we show that after round 1 layer $\tilde{L}_k$ is identified with high probability.

In the second step, we show that from there on we identify one more layer each round, i.e., after round r we have identified layer $\tilde{L}_{k+r-1}$. This, however, cannot hold w.h.p. once the weights drop below $\mathcal{O}(\text{polylog}\, n)$. Instead, each vertex $v \in \tilde{L}_j$, $j > k$ is identified after round $j-k+1$ with probability at least $1-\alpha_j \geqslant 1-\exp(-\Omega(p_1 p_2 q_1 q_2 \delta))$. This holds independently of the other vertices in L_j and of the edges from vertices above layer L_j to vertices below layer L_j or layer L_j itself. This way we identify most of the vertices in the layers above k in at most $\log(n) - k + 1$ rounds. We remark that these vertices could be identified already earlier, but we claim that they are identified at the latest after round $j - k + 1$ (with the mentioned probability).

In the third step, we show that after round $\log(n) - k + 2$ all vertices in layers below $\tilde{L}_k$ are identified with high probability. As the number of such vertices is small, this third step is not necessary for the conclusion that the algorithm identifies a large fraction of all identifiable vertices — it proves, however, the intuition that this algorithm identifies all vertices with sufficiently high weight ($\log^{\Omega(1)}(n)$) with high probability. We omit this third step in this extended abstract.

First step. Initially we know the identity of a set V_I of vertices containing the h highest weight nodes that survive in both graphs. We let $h = \gamma^2 \frac{1}{p_1 p_2} \log n$ where γ is a sufficiently large constant. Let $\ell := \gamma \frac{1}{p_1 p_2} \log n$. Choose a layer L_k such that for any $v \in L_k$ we have $p_{v,\ell} \approx b$, where $b = \Theta(1)$ is the constant from Lemma 4, so that we have $|B| = \mathcal{O}(\ell) = \mathcal{O}(\gamma \frac{1}{p_1 p_2} \log n)$ *bad nodes*. For our weight distributions one can show that $p_{v,h} = p_{v,\ell} \cdot (\ell/h)^{1/(\beta-1)} = \Theta(\gamma^{-1/(\beta-1)})$. Hence, any node $1 \leqslant u \leqslant h$ has $p_{v,u} \geqslant p_{v,h} = \Theta(\gamma^{-1/(\beta-1)})$. Summing up over $1 \leqslant u \leqslant h$, we have $\sum_{u \in V_I} p_{v,u} \geqslant \Omega(\gamma^{2-1/(\beta-1)} \frac{1}{p_1 p_2} \log n)$. Since $\gamma^{2-1/(\beta-1)} = \gamma^{1+\Omega(1)}$ and γ is sufficiently large, for any arbitrarily large hidden constant we have

$$\sum_{u \in V_I} p_{v,u} \geqslant \Omega((\gamma + c) \tfrac{1}{p_1 p_2} \log n) = \Omega(\tfrac{1}{p_1 p_2} c \log n + |B|),$$

which proves that the assumption of Lemma 4 is fulfilled and we identify v in the first round with high probability.

Second step. Consider any following level $k < j \leqslant \log n - \Omega(\log \log n)$ (with sufficiently large hidden constant) and let $v \in \tilde{L}_j$. We prove by induction that v is identified in round $j-k+1$ with high probability. By induction hypothesis, every vertex $u \in \tilde{L}_{j-1}$ is identified after round $j-k$ with high probability. The probability of v to connect to a vertex $u \in L_{j-1}$ is $p_{u,v} = \min\{w_v w_u / W, 1\}$. Plugging in

$w_v, w_u = \Theta(\delta(n/2^j)^{1/(\beta-1)})$ yields $p_{u,v} = \Theta(\varepsilon)$ for $\varepsilon := \delta n^{(3-\beta)/(\beta-1)}\, 2^{-2j/(\beta-1)}$ and $\overline{p}_{u,v} = \mathcal{O}\big(p_{u,v} + \frac{1}{\delta n}\log^2 n\big)$. Note that since $j \leqslant \log n - \Omega(\log\log n)$ we have $w_v \geqslant \log^{\Omega(1)} n$ so that $\overline{p}_{u,v} = \mathcal{O}(p_{u,v})$. We can apply Lemma 4 with $|B| \leqslant \mathcal{O}(\frac{1}{p_1 p_2}\log n)$ (since the number of bad vertices is at most the number of bad vertices for layer L_k). Considering only the edges to $V_I \cap \tilde{L}_{j-1}$ and using $|\tilde{L}_{j-1}| = \Omega(q_1 q_2 2^j)$ we obtain

$$\sum_{u \in V_I} p_{u,v} \geqslant \Omega\big(q_1 q_2 2^j \delta n^{(3-\beta)/(\beta-1)}\, 2^{-2j/(\beta-1)}\big) = \Omega\big(q_1 q_2 \delta \log^{\Omega(1)} n\big),$$

which is larger than $\Omega(\frac{1}{p_1 p_2}\log n)$ since $p_1 p_2 q_1 q_2 \delta$ is at least a sufficiently large constant. Hence, Lemma 4 implies that we identify all vertices in $\tilde{L}_j$ with high probability.

For $\log n - o(\log n) \leqslant j \leqslant \log n$, so that $w_v = n^{o(1)}$, we instead use Lemma 5 to show that any vertex $v \in \tilde{L}_j$ is identified after round $j-k+1$ with probability at least $1 - \alpha_j \geqslant 1 - \exp(-\Omega(p_1 p_2 q_1 q_2 \delta))$. We again consider the edges of v into $T := V_I \cap \tilde{L}_{j-1}$ and obtain

$$\mu := p_1 p_2 \varepsilon |T| = \Omega\big(p_1 p_2 q_1 q_2 2^j \delta n^{(3-\beta)/(\beta-1)}\, 2^{-2j/(\beta-1)}\big) = \Omega(p_1 p_2 q_1 q_2 \delta).$$

Hence, the assumption of Lemma 5 amounts to $p_1 p_2 q_1 q_2 \delta$ being at least a sufficiently large constant, and we identify each vertex in $\tilde{L}_j$ with probability at least $1 - \alpha_j := 1 - \exp(-\Omega(\mu)) - n^{-c} \geqslant 1 - \exp(-\Omega(p_1 p_2 q_1 q_2 \delta))$.

5 Quasilinear Runtime

Algorithm 1 in its pure form takes quadratic time, as we have seen in the last section. In this section we show how to decrease its runtime to quasilinear using locality sensitive hashing [6]. We assume to have identified the $h = n^{2\varepsilon}$ highest weight vertices for any constant $\varepsilon > 0$.

Algorithm 2 Fast de-anonymization using Y-tests

Input: graphs G_1, G_2, identified vertices $V_I \supseteq \{1, \dots, n^{2\varepsilon}\}$
  for $r = 1, 2, \dots, \Theta(p)^{-\Theta(1/\varepsilon)} \log n$ **do**
    choose a random permutation π of V_I
    for all $v \in V \setminus V_I$ and $i \in \{1, 2\}$ **do**
      if $|N_i(v) \cap T_v| \geqslant C/\varepsilon$ **then**
        compute the set M_v^i of the first C/ε vertices in $N_i(v) \cap T_v$ w.r.t. π
        hash (v, i) at M_v^i
    for all hash collisions of the form $(v_1, 1)$ and $(v_2, 2)$ **do**
      if $Y^{v_1, v_2} > n^c$ **then**
        $V_I := V_I \cup \{v_1, v_2\}$. ▷ We identified $v_1 = v_2$

The basic idea for speeding up the algorithm is to reduce the number of tested pairs v_1, v_2. To this end, in every round we choose a random permutation

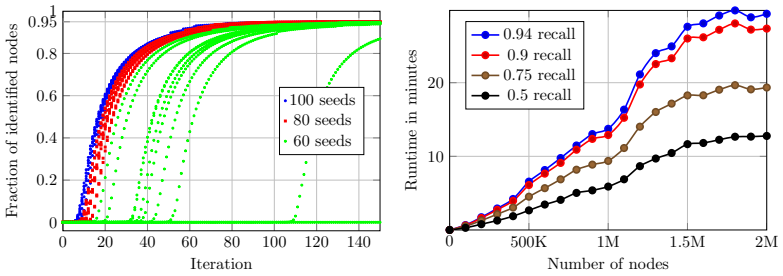

Figure 1: Algorithm performance on Chung-Lu graphs with $\beta = 2.5$, $\delta = 30$ and subsampling probabilities $p_i = 0.5$, $q_i = 0.85$. The left plot shows recall vs. iterations for 30 different runs with $n = 10^6$ nodes. The precision is $1 - 10^{-5}$ for all runs. The right plot shows the runtime depending on the number of nodes n.

π of V_I. For a vertex $v \in V \setminus V_I$ in G_i consider the vertices in V_I that have a small estimated probability to connect to v, $T_v := \{u \in V_I \mid \overline{p}_{v,u} \leqslant n^{-\varepsilon}\}$. We compute the first C/ε vertices $(u_1, \dots, u_{C/\varepsilon}) =: M_v^i$ in $N_i(v) \cap T_v$ with respect to the order π (for some constant $C \geqslant 2$ to be fixed later). Note that M_v^i can be computed in constant time, if we permute the graphs $G_1[V_I]$ and $G_2[V_I]$ with respect to π (and store a version containing only the edges in $\bigcup_v T_v$), so that the first neighbor of v is simply the first entry of its adjacency list. In the analysis we show that for a so-called *good* vertex $v \in V_\cap$ the sets M_v^1, M_v^2 are equal with probability $\Theta(p_1 + p_2)^{\Theta(1/\varepsilon)}$, while for non-identical vertices $v_1 \neq v_2$ these sets are equal with probability at most $1/n$. Thus, we may hash v at M_v^i (with a perfect hash function that produces collisions only if the corresponding hash values are equal) and test vertices v_1, v_2 only if they form a hash collision. Then in expectation we test $\mathcal{O}(n)$ pairs of vertices per round. More precisely, one can show that these tests take expected time $\mathcal{O}(m)$ per round, where m is the total number of edges in G_1 and G_2. Everything else we do in one round also runs in time $\mathcal{O}(m)$. Note that in expectation we have $m = \mathcal{O}((p_1 + p_2)\delta n) \leqslant \mathcal{O}(\delta n)$.

As we will see in the full version, roughly the same quality analysis as in the last section goes through, with the necessary number of rounds growing to $\Theta(\min\{p_1, p_2\})^{-\Theta(1/\varepsilon)} \log n$. As $\varepsilon > 0$ is a constant, this is $\mathcal{O}(\min\{p_1, p_2\}^{-\mathcal{O}(1)} \log n)$. Furthermore, by the same arguments as in the last section, Algorithm 2 makes no wrong identifications w.h.p. (intuitively, it only does a subset of the Y-tests of Algorithm 1, but since we let it run for a few more rounds the error probability grows to $\mathcal{O}(n^{2-c} \min\{p_1, p_2\}^{-\mathcal{O}(1)} \log n)$). In total we get an expected runtime of $\mathcal{O}(\min\{p_1, p_2\}^{-\mathcal{O}(1)} \delta n \log n)$.

6 Experiments

The focus of this paper lies on the theoretical algorithm analysis. To check our theory for robustness, however, we conducted a preliminary empirical study. We implemented a variant of the fast algorithm single threaded in C++. The source code is available upon request. The experiments were run on a single computer with Dual Xeon CPU E5-2670 and 128 GB RAM.

Graph Model								
Name	n	m	$p_1 \cdot p_2$	$q_1 \cdot q_2$	Seeds	Recall	Prec.	Runtime
Chung-Lu	2M	40M	0.25	0.72	80	0.95	1	29 min.
Pref. Attachment	1M	20M	0.25	1	200	0.95	1	9 min.
Affiliation Network	60K	8M	–	1	50	0.83	0.99	56 sec.
Facebook	63K	1.5M	0.58	1	50	0.50	0.95	6 sec.
Orkut	3M	117M	0.56	0.81	1000	0.89	0.88	54 min.

Table 1: Performance of our de-anonymization algorithm on various graphs.

We evaluated the algorithm on Chung-Lu graphs as shown in Figure 1. The plots indicate that our quality bounds hold up well in practice, as we identify 95% of the nodes with as little as 0.008% seeds (80 nodes). Similarly, the runtime plot follows our asymptotic bounds which allows for deanonymizing large graphs (2 million nodes) on a single core, whereas previous approaches would typically require a computing cluster due to their polynomial runtime.

Finally, we investigated the robustness of our algorithm with respect to changes to the underlying graph model. We ran it on different random graph models (Preferential Attachment [3], Affiliation Networks[3] [8]) and even real graphs (Facebook, Orkut)[4]. The results are presented in Table 1. We point out that [7] also performed experiments on Facebook and Affiliation Networks, achieving slightly better recall (0.6 and 0.9, respectively). However, they typically use 10% of the networks as seeds; and they do not report on their runtimes and machines.

In all cases, our algorithm was able to extend the small set of identified seed nodes to a linear fraction of the entire graph; while making comparatively few errors. This indicates that even though our proofs rely on the topology of the Chung-Lu model (e.g. independent edge probabilities), the algorithm performs reasonably well in practice.

7 Conclusion

We presented a new method for de-anonymizing scale-free networks with two crucial improvements compared to previous work: (i) faster runtime and (ii) less required a-priori knowledge.

While all previous algorithms have a runtime of $\Omega(n\Delta)$, our new algorithm runs in quasilinear time. This improvement is not only asymptotical: Recent experiments of Korula and Lattanzi [7] required large compute clusters, whereas our algorithm can handle graphs with millions of vertices in less than an hour on off-the-shelf hardware. The quasilinear runtime is achieved by a variant of locality sensitive hashing. We believe that this technique can be used in future work to speed up other matching and graph isomorphism algorithms.

Our second contribution is a rigorous proof that much fewer seed nodes suffice for de-anonymizing subsamples of a common model of scale-free networks. Our

[3] In this model, a bipartite graph of users and interests is constructed; and two users are connected if they share an interest. To create two subsampled graphs, each interest is deleted independently with probability 0.25 in both graphs.

[4] `http://snap.stanford.edu/data/`

approach needs only n^{ε} seed nodes, while all previous algorithms with proven runtime and quality use $\Theta(n)$ seed nodes. The analysis is based on a new weight estimation scheme relative to an adaptive adversary, which appears to be useful also for analyzing other algorithms on Chung-Lu graphs. Our result shows that de-anonymization is possible with few seed nodes, which is important for practical attacks on anonymized networks.

Bibliography

[1] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In *32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 171–180, 2000.

[2] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: Anonymized social networks, hidden patterns, and structural steganography. In *16th International Conference on World Wide Web (WWW)*, pages 181–190, 2007.

[3] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[4] F. Chung and L. Lu. Connected components in random graphs with given expected degree sequences. *Annals of Combinatorics*, 6(2):125–145, 2002.

[5] F. Chung and L. Lu. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences (PNAS)*, 99(25):15879–15882, 2002.

[6] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 604–613, 1998.

[7] N. Korula and S. Lattanzi. An efficient reconciliation algorithm for social networks. In *40th International Conference on Very Large Data Bases (VLDB)*, pages 377–388, 2014.

[8] S. Lattanzi and D. Sivakumar. Affiliation networks. In *41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 427–434, 2009.

[9] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *30th IEEE Symposium on Security and Privacy (SP)*, pages 173–187, 2009.

[10] J. Novak, P. Raghavan, and A. Tomkins. Anti-aliasing on the web. In *13th International Conference on World Wide Web (WWW)*, pages 30–39, 2004.

[11] J. R. Rao and P. Rohatgi. Can pseudonymity really guarantee privacy? In *9th USENIX Security Symposium (USENIX)*, pages 85–96, 2000.

[12] R. van der Hofstad. Random graphs and complex networks. Available at `www.win.tue.nl/~rhofstad/NotesRGCN.pdf`, 2009.

[13] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel. A practical attack to de-anonymize social network users. In *IEEE Symposium on Security and Privacy (SP)*, pages 223–238, 2010.

[14] R. Zafarani and H. Liu. Connecting corresponding identities across communities. In *3rd International Conference on Weblogs and Social Media (ICWSM)*, pages 354–357, 2009.

[15] E. Zheleva and L. Getoor. To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles. In *18th International Conference on World Wide Web (WWW)*, pages 531–540, 2009.