Algorithmica (2018) 80:3397–3427
https://doi.org/10.1007/s00453-017-0395-0

De-anonymization of Heterogeneous Random Graphs in Quasilinear Time

Karl Bringmann[1] · **Tobias Friedrich**[2] · **Anton Krohmer**[2]

Received: 7 September 2016 / Accepted: 8 November 2017 / Published online: 15 November 2017
© Springer Science+Business Media, LLC, part of Springer Nature 2017

Abstract There are hundreds of online social networks with altogether billions of users. 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.

Keywords Social networks · Locality-sensitive hashing · Network privacy

A preliminary conference version [8] without most proofs appeared in the 22nd European Symposium on Algorithms (ESA 2014).

✉ Anton Krohmer
Anton.Krohmer@hpi.de

Karl Bringmann
kbringma@mpi-inf.mpg.de

Tobias Friedrich
Tobias.Friedrich@hpi.de

1 Max Planck Institute for Informatics, Saarbrücken, Germany

2 Hasso Plattner Institute, Potsdam, Germany

1 Introduction

Imagine owning a large social network G_1 (like Facebook or Google+), and that 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 e.g. through 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, 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.

1.1 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. Empirical studies showed that many social networks have a power law or log normal degree sequence [21,24]. Since both of these distributions are closely related [21], we focus on the power law case by modeling G as an n-vertex Chung–Lu random graph [1,2,9–11]. 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 σ_i, and every edge of $G[V(G_i)]$ exists in G_i independently with probability ρ_i. Additionally, we randomly permute the vertices of the graphs G_1 and G_2, i.e., we forget the vertex labels and permute the order of the vertices in the input description.[1] We assume that there is a set of high-degree seed nodes $V_I \subseteq V$ which are known to match between G_1 and G_2 because, for example, the person concerned generates high 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.

This setup is similar to [18], only that instead of the Preferential Attachment graphs [7] we use Chung–Lu random graphs. These are typically more accessible to probabilistic analysis, yet it has been shown that both models exhibit similar behavior [28].

1.2 Theoretical Results

We present an algorithm with the following guarantees. Here we let δ be the (expected) average degree of the graph G. See Sect. 2 for the technical assumptions about the

[1] As our graphs are given by adjacency lists, this means that for each graph G_i we pick a random permutation π_i. Then the jth adjacency list becomes the $\pi_i(j)$th adjacency list, each entry ℓ of an adjacency list is replaced by $\pi_i(\ell)$, and finally we sort each adjacency list to obtain a proper description of the permuted graph.

parameters of the Chung–Lu random graph G and the parameters of the subsampling process.

Theorem 1 *Assume we are given the n^ε identifiable vertices with largest degrees as seed nodes for an arbitrary constant $\varepsilon > 0$. There is an algorithm that with high probability*[2] *makes no wrong identifications and successfully matches a fraction of $1-\exp(-\Omega(\rho_1\rho_2\sigma_1\sigma_2\delta))$ of the identifiable vertices.*[3] *The algorithm runs in expected quasilinear runtime $\mathcal{O}(\delta n \log(n)/\min\{\rho_1, \rho_2\}^{\mathcal{O}(1/\varepsilon)})$.*

We also show that this fraction of identified vertices is asymptotically optimal, since, roughly speaking, we show that an $\exp(-\mathcal{O}(\rho_1\rho_2\sigma_1\sigma_2\delta))$ fraction of the vertices does not have any common neighbors in the two social networks and thus cannot be identified. For constant $\rho_1, \rho_2, \sigma_1, \sigma_2$, 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 $\mathcal{O}(\delta n\, \Delta^2)$ [22] or $\mathcal{O}(\delta n\, \Delta \log(\Delta))$ [18], where Δ is the maximum degree, which is typically of order $n^{\Omega(1)}$.

1.3 Seeds

Our approach uses only the n^ε largest degree nodes as seeds—previous algorithms with proven runtime and quality use at least $\Theta(n)$ uniformly sampled seeds [18]. In practice, it is often easier to find an initial matching of nodes with large degrees. For example, high-degree nodes in social networks typically correspond to public figures[4] which can be easily identified in both instances G_1, G_2. Let us also remark that our analysis goes through even when only a linear fraction of the largest n^ε nodes is given as seeds. This essentially corresponds to increasing the rate of subsampled nodes σ_1, σ_2 by a constant factor in the bootstrapping phase. Thus, our assumptions are strictly weaker than knowing a linear fraction of *all* nodes as seeds which was required in [18]. Finally, our algorithm is also successful with only $\mathcal{O}(\log(n)/\rho_1\rho_2)$ seed nodes, but runs in quadratic time in this case.

1.4 Empirical Results

Even though the Chung–Lu random graph is a popular model for scale-free networks, it does not reflect all properties observed in real world graphs. Most prominently, these graphs have a large clustering coefficient, whereas the Chung–Lu model assumes

[2] 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 $c > 0$.

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

[4] A realistic application of deanonymization algorithms such as ours could be to manually identify a few high degree nodes, corresponding to public figures, and to run the algorithm to identify the remaining nodes. For the manual step, one may exploit additional metadata—such high-profile vertices are typically public and share lots of information. The algorithm itself does not rely on any such information and only uses graph structure.

independent edges and therefore has small clustering. To check our approach for robustness, we thus implemented a variant of our algorithm and applied it to different sets of networks.

We identify $\geq$ 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 [18]. In all cases, we need $\leq$ 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.

1.5 Algorithm Description

Starting with the seed nodes, we identify the remaining vertices by their *signatures*, i.e., their neighbors among the identified vertices, an idea used in many algorithms for graph isomorphism. 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 [16], which reduces the number of comparisons.

1.6 Applications

Anonymous copies of some social networks are available online. Several experimental papers describe how to find mappings between two online social networks [6,22,30, 31]. While some use the network structure alone [22], most of them exploit metadata like browser history [30], group memberships [32], writing style [26], semantic features of user aliases [25], or artificially added subgraphs [6]. The only theoretical result on this subject is by [18]. They identify 97% of the nodes on subsampled ($\rho_1, \rho_2 \geq \sqrt{22/\delta}$, $\sigma_1 = \sigma_2 = 1$) preferential attachment graphs [7], but need substantially more computing resources and a linear amount of uniformly sampled seed nodes.

1.7 Related Work

The studied problem is a variant of the *Graph Isomorphism* (GI) problem, a famous problem in graph theory. GI is one of the rare computational problems which is neither known to be polynomial-time tractable nor NP-Complete. Closest to our interest is the optimization version Max-GI, aiming at maximizing the number of mapped edges. Arvind et al. [5] showed that it is NP-hard to approximate this beyond a factor of 1.06. They further proved that also minimizing the number of mismatches is NP-hard to approximate for any constant factor. Max-GI restricted to trees is still NP-hard to

approximate within a factor of $1 + \Omega(1/\log n)$ [29]. On the other hand, Max-GI on dense graphs can be $(1+\varepsilon)$-approximated in time $n^{\mathcal{O}(\log n/\varepsilon^2)}$ [5]. Newman and Sohler [23] proved for any graph family defined by a set of forbidden minors that the property of two graphs being ε-far from isomorphic is efficiently testable. The main difference between GI and our question stems from the subsampling. This implies that a vertex does not have the same neighbors in both graphs and that we only want to identify a large fraction of the vertices, not all.

Orthogonally to our work, a well-studied technique for releasing provably privacy-preserving statistics is *differential privacy* [13]. It protects privacy by adding calibrated noise to the data. Differentially-private graph models were studied in [27]. Differential privacy is even harder to achieve in distributed settings with more than one party [20]. We note that our results give one possible theoretical explanation of why differential privacy techniques are necessary and simple anonymization (such as removing vertex labels) does not suffice in the social network context.

The family of random graphs we consider was introduced by Chung and Lu [10,11] as a model for generating graphs with a power law degree sequence. Their graph properties have been studied extensively. For example, with high probability, there is a giant connected component that contains a linear fraction of the nodes [9] and the average distance is $\mathcal{O}(\log\log n)$ [11,12]. Algorithmically, they have been examined in the context of information dissemination [14], bootstrap percolation [4], and finding cliques [15]. For a thorough survey on power-law graphs, we refer to [28].

1.8 Organization

Section 2 introduces the graph model and our notations. In Sect. 3, we show how to compute upper and lower bounds for the weights of the graph model. In Sect. 4 we present our algorithm. Finally, Sect. 5 demonstrates how to achieve a quasilinear runtime by locality sensitive hashing and Sect. 6 proves that the number of identified vertices is asymptotically optimal. In Sect. 7 we experimentally evaluate the performance of our algorithm.

2 Preliminaries

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 $c > 0$.

2.1 Graph Model

The model has two adjustable parameters: the exponent of the scale-free network β and the average degree δ, both chosen independently of n. Depending on these two parameters, each node i has a weight w_i. For $n \in \mathbb{N}$ and weight distribution $\mathbf{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n_{\geq 0}$ 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 [28]. We note that most of our results generalize to other weight distributions and even to weights drawn at random from "nice" distributions. We assume a constant $2 < \beta < 3$, as this holds for most real-world social networks [24]. Moreover, we require $\delta \leq n^{o(1)}$, $\rho_1, \rho_2 \geq n^{-o(1)}$, and $\sigma_1, \sigma_2 = \Theta(1)$. We also assume that we know a lower bound for σ_1, σ_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 $\rho_1\rho_2\sigma_1\sigma_2\delta$ is at least a sufficiently large constant (depending only on β).

2.2 De-anonymization

Let an underlying graph $G = \text{Chung–Lu}(n, \mathbf{w})$ be given. This graph is subsampled twice to generate two subgraphs. This means we put each node $v \in V := V(G)$ into V_1 independently with probability σ_1. Then we put each edge $e \in E \cap \binom{V_1}{2}$ into E_1 independently with probability ρ_1 to form a graph $G_1 = (V_1, E_1)$. Now we randomly permute the nodes of G_1 to obtain a graph $\widetilde{G}_1$. We repeat this process with independent choices (and probabilities σ_2, ρ_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, $i \in \{1, 2\}$.

2.3 Chernoff Bound

We use the following standard concentration inequality.

Lemma 1 (Chernoff bound, see [3]) *Let $X_1, \ldots, X_n$ be independent random variates taking values in $\{0, 1\}$. Let $X := X_1 + \cdots + X_n$ and $\mu := \mathbb{E}[X]$. Then for any $t \geq 0$ we have*

$$\Pr[|X - \mu| > t] \leq \exp\left(-t^2/(2\mu + t/3)\right).$$

Slightly stronger, for any $\lambda > \mu$ we have

$$\Pr[X > \lambda] \leq \left(\frac{\mathrm{e}\,\mu}{\lambda}\right)^{\lambda}.$$

Our default variant of Chernoff is the first bound. We will often simplify the error probability to $\exp(-\Omega(t^2/\mu))$ if $t = \mathcal{O}(\mu)$, and to $\exp(-\Omega(t))$ if $t = \Omega(\mu)$. Applied to the concentration of node degrees, it directly follows

$$\Pr\left[|\deg_i(v) - \mu| > t \mid v \in V_i\right] \leq \exp\left(-t^2/(2\mu + t/3)\right).$$

2.4 Expected Degrees in Chung–Lu Graphs

We will make use of the following standard technical facts about Chung–Lu random graphs.

Lemma 2 1. *For any $z > 0$ we have $\sum_{u\in V,\, w_u > z} w_u/z = \mathcal{O}(n(\delta/z)^{\beta-1})$.*
2. *For any vertex v, we have $w_v \geq \mathbb{E}[\deg(v)] \geq w_v - \mathcal{O}(w_v^2/W + n^{2-\beta}w_v^{\beta-1})$.*
3. *We have $W \geq \sum_{v\in V}\mathbb{E}[\deg(v)] \geq W - \mathcal{O}(\delta n^{(3-\beta)/(\beta-1)} + \delta^{\beta-1}n^{3-\beta}\log n) = W(1-o(1))$.*

Proof For (1), we define $f(x) = \delta(n/x)^{1/(\beta-1)}$ and thus $f^{-1}(z) = n(\delta/z)^{\beta-1}$. Approximating the sum by an integral, we obtain

$$\sum_{u\in V,\, w_u > z} w_u \leq \int_0^{f^{-1}(z)} f(x)dx.$$

Inspecting $f(x)$ we see that its integral is $\frac{\beta-1}{\beta-2}\cdot x\cdot f(x)$. This yields

$$\sum_{u\in V,\, w_u > z} w_u \leq \left[\tfrac{\beta-1}{\beta-2}\cdot x\cdot f(x)\right]_0^{f^{-1}(z)} = \tfrac{\beta-1}{\beta-2}\cdot f^{-1}(z)\cdot f(f^{-1}(z)),$$

which evaluates to $\frac{\beta-1}{\beta-2}n\left(\frac{\delta}{z}\right)^{\beta-1}\cdot z$. Dividing both sides by z yields (1).

For (2), note that

$$\mathbb{E}[\deg(v)] = \sum_{u\neq v}\min\{w_u w_v/W, 1\} \leq \sum_{u\in V} w_u w_v/W = w_v.$$

For the other direction, note that the above inequality only loses at the terms with $u = v$ and with u such that $w_u w_v/W > 1$. We thus have

$$w_v - \mathbb{E}[\deg(v)] \leq w_v^2/W + \sum_{u\in V,\, w_u > W/w_v} w_u w_v/W.$$

Observing that this sum is of the same form as in (1) with $z = W/w_v = \Theta(\delta n/w_v)$, we obtain

$$w_v - \mathbb{E}[\deg(v)] \leq \mathcal{O}\left(w_v^2/W + n(w_v/n)^{\beta-1}\right) = \mathcal{O}\left(w_v^2/W + n^{2-\beta}w_v^{\beta-1}\right).$$

Finally, for (3) we sum up (2) over all vertices v to obtain

$$W \geq \sum_{v \in V} \mathbb{E}[\deg(v)] \geq W - \mathcal{O}\left(\sum_{v \in V} \left(w_v^2/W + n^{2-\beta} w_v^{\beta-1}\right)\right).$$

We further bound this error term by replacing sums by integrals as above. This yields

$$\sum_{v \in V} w_v^2 \leq f(1)^2 + \int_1^n f(x)^2 dx = f(1)^2 + \left[\tfrac{\beta-1}{\beta-3} x f(x)^2\right]_1^n = \mathcal{O}(f(1)^2),$$

where we used $2 < \beta < 3$. Using $f(1) = \delta n^{1/(\beta-1)}$ we thus obtain

$$\sum_{v \in V} w_v^2/W = \mathcal{O}(\delta^2 n^{2/(\beta-1)}/(\delta n)) = \mathcal{O}(\delta n^{(3-\beta)/(\beta-1)}).$$

We similarly bound

$$\sum_{v \in V} n^{2-\beta} w_v^{\beta-1} = \delta^{\beta-1} n^{3-\beta} \sum_{v=1}^{n} v^{-1} = \mathcal{O}(\delta^{\beta-1} n^{3-\beta} \log n).$$

Hence, the total error is bounded by $\mathcal{O}(\delta n^{(3-\beta)/(\beta-1)} + \delta^{\beta-1} n^{3-\beta} \log n)$. In our regime $\beta < 3$ and $\delta = n^{o(1)}$, this evaluates to $o(n) = o(W)$, and thus we even have $\sum_{v \in V} \mathbb{E}[\deg(v)] \geq W(1 - o(1))$. □

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 is crucial for the deanonymization algorithm, as it uses vertex weights to judge the information given by an edge. Intuitively, an edge to an identified high weight vertex is less useful (since the vertex is likely to be connected to many other nodes) than an edge to an identified low weight vertex (since this narrows down possible candidates for matching).

Our bounds on vertex weights also yield 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, ρ_1, and ρ_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 ρ_2 (and ρ_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 the quotient σ_1/σ_2 (e.g. from $|V_1|/|V_2|$), but our algorithms do not need them. Note that in our model it is hard to estimate the parameters σ_1, σ_2.

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,

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

if node v survives in G_i. Chernoff bound shows that this degree is concentrated. This allows us to compute intervals for the weights w_v for all nodes v in G_i.

Lemma 3 *Let $i \in \{1, 2\}$. Given $\deg_i(v)$ (and ρ_i and an approximation of n) we can compute $0 \le \underline{w}_{v,i} \le \bar{w}_{v,i}$ such that w.h.p. for all $v \in V$ we have*

1. $\underline{w}_{v,i} \le \sigma_i w_v \le \bar{w}_{v,i}$,
2. $\bar{w}_{v,i} = \mathcal{O}\big(\sigma_i w_v + \frac{1}{\rho_i} \log n\big)$, *and*
3. $\underline{w}_{v,i} = \Omega(\sigma_i w_v) - \mathcal{O}\big(\frac{1}{\rho_i} \log n\big)$.

Proof Let

$$\mu := \mathbb{E}[\deg_i(v) \mid v \in V_i] = \rho_i \sigma_i \mathbb{E}[\deg(v)].$$

By Chernoff bound (Lemma 1) we obtain

$$\Pr\left[|\deg_i(v) - \mu| > t \mid v \in V_i\right] \le \exp(-t^2/(2\mu + t/3)).$$

Solving a quadratic equation shows that this probability bound is n^{-c} at

$$t = c \log(n)/6 + \sqrt{(c \log(n)/6)^2 + 2\mu c \log n} \le 2c \log n + \tfrac{1}{3}\mu,$$

where the second step follows from $A + \sqrt{A(A+B)} \le 12A + B/36$ for any $A, B \ge 0$. From Lemma 2(2) we obtain $\mu \le \rho_i \sigma_i w_v$, which yields w.h.p.

$$|\deg_i(v) - \mu| \le 2c \log n + \tfrac{1}{3}\mu \le 2c \log n + \tfrac{1}{3}\rho_i \sigma_i w_v.$$

Moreover, from Lemma 2(2) we see that $\mu = \rho_i \sigma_i \mathbb{E}[\deg(v)]$ differs from $\rho_i \sigma_i w_v$ only by $\rho_i \sigma_i \cdot \mathcal{O}(w_v^2/W + n^{2-\beta} w_v^{\beta-1}) = \rho_i \sigma_i w_v \cdot \mathcal{O}(\frac{w_v}{W} + (\frac{w_v}{n})^{\beta-2}) = o(\rho_i \sigma_i w_v)$, since the maximum weight is $w_1 = \delta n^{1/(\beta-1)} = o(n) = o(W)$. By triangle inequality, we obtain

$$|\deg_i(v)/\rho_i - \sigma_i w_v| \le \tfrac{2c}{\rho_i} \log n + \left(\tfrac{1}{3} + o(1)\right) \sigma_i w_v.$$

Note that for $A, B, C \geq 0$ the inequality $|A-B| \leq C+(\frac{1}{3}+o(1))B$ implies $|A-B| \leq (1+o(1))(\frac{3}{2}C+\frac{1}{2}A)$. This follows since for $B = xC$ we have $A \geq (1-\frac{1}{x}-\frac{1}{3}-o(1))B$ so that $\frac{3}{2}C+\frac{1}{2}A \geq (\frac{3}{2x}+\frac{1}{2}(1-\frac{1}{x}-\frac{1}{3}-o(1)))B \geq (\frac{1}{x}+\frac{1}{3}-o(1))B = C+(\frac{1}{3}-o(1))B$. Hence,

$$|\deg_i(v)/\rho_i - \sigma_i w_v| \leq (1+o(1))\left(\tfrac{3c}{\rho_i}\log n + \tfrac{1}{2}\deg_i(v)/\rho_i\right).$$

This yields a bound $\bar{w}_{v,i}$ for $\sigma_i w_v$ that we can compute with the given data and that holds with probability at least $1-n^{-c}$, namely

$$\bar{w}_{v,i} = (1+o(1))\left(\tfrac{3}{2}\deg_i(v)/\rho_i + \tfrac{3c}{\rho_i}\log n\right).$$

Note that from this bound we have $\bar{w}_{v,i} \geq \sigma_i w_v$ and $\bar{w}_{v,i} = \mathcal{O}(\frac{1}{\rho_i}\log n + \sigma_i w_v)$. Similarly, we get a lower bound $\underline{w}_{v,i} \leq \sigma_i w_v$ with $\underline{w}_{v,i} = \Omega(\sigma_i w_v) - \mathcal{O}(\frac{1}{\rho_i}\log n)$. □

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

Lemma 4 *Let $i \in \{1,2\}$. Given G_i, we can compute $\underline{W}_i$ such that $\underline{W}_i \leq \sigma_i^2 W \leq (1+o(1))\underline{W}_i$ holds with high probability.*

Proof Let $W_i = \frac{1}{\rho_i}\sum_{u\in V_i}\deg_i(u)$. Let $X_u = 1$ if node u survives in G_i, 0 otherwise. Similarly, let $Y_{uv} = 1$ if edge $\{u,v\}$ is present in G and it would survive in G_i (if both u and v survive), 0 otherwise, so that $\Pr[Y_{uv}=1] = \rho_i \cdot p_{u,v}$. Then we obtain

$$W_i = \tfrac{1}{\rho_i}\sum_{u\neq v\in V} X_u X_v Y_{uv}.$$

It is easy to see that $\mathbb{E}[W_i] = \sigma_i^2\mathbb{E}[\sum_{v\in V}\deg(v)]$, and thus $\sigma_i^2 W \geq \mathbb{E}[W_i] \geq \sigma_i^2 W(1-o(1))$ by Lemma 2(3). To obtain a tight tail bound for W_i, we apply a concentration bound on multivariate polynomials [17]. Let $S = \{X_u \mid u \in V\} \cup \{Y_{uv} \mid u, v \in V\}$ be the set of variables in the polynomial W_i. Let $\frac{\partial W_i}{\partial A}$ be the partial derivative of W_i with respect to the variables in A and define

$$E_j(W_i) = \max_{A\subseteq S:\,|A|=j}\mathbb{E}\left[\frac{\partial W_i}{\partial A}\right].$$

Intuitively, $E_j(W_i)$ yields the "average effect" a group of j variables has on W_i. For a more thorough explanation, we refer the reader to [17]. Computing $E_0(W_i) = \mathbb{E}[W_i] \leq \sigma_i^2 W$, $E_1(W_i) \leq \sigma_i w_1 = \sigma_i \delta n^{\frac{1}{\beta-1}}$, $E_2(W_i) \leq 1$ and $E_3(W_i) = 1$, we obtain the bound

$$\Pr\left[|W_i - \mathbb{E}[W_i]| > c\left(\sigma_i^3 W \delta n^{\frac{1}{\beta-1}}\right)^{1/2}\lambda^3\right] = \mathcal{O}(\exp(-\lambda + 2\log n)),$$

for some constant c. This bound becomes $n^{-c'}$ at $\lambda = \Theta(\log n)$, which implies that w.h.p. we have

$$|W_i - \mathbb{E}[W_i]| = \mathcal{O}\left(\delta\sigma_i^{3/2}(\log n)^3 n^{\frac{\beta}{2\beta-2}}\right) = n^{1-\Omega(1)} = o\left(\sigma_i^2 W\right).$$

Choosing $\underline{W_i} = (1 - o(1))\, W_i$ gives the desired result. □

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 $\bar{w}_{v,i}$ and $\underline{W_i}$ give a result depending on σ_i, the computed upper bound $\bar{p}_{u,v,i}$ on the edge probabilities is oblivious to σ_i.

Corollary 1 *Let $i \in \{1, 2\}$. For any pair of nodes $u, v \in V_i$, we can compute bounds on the edge probability $\bar{p}_{u,v,i} := \min\{\bar{w}_{u,i}\bar{w}_{v,i}/\underline{W_i}, 1\}$ such that w.h.p. we have*

$$p_{u,v} \leq \bar{p}_{u,v,i} = \mathcal{O}\Big(p_{u,v}\Big(1 + \frac{\log n}{\rho_i\sigma_i w_u}\Big)\Big(1 + \frac{\log n}{\rho_i\sigma_i w_v}\Big)\Big).$$

In particular, for $w_u, w_v = \Omega\left(\frac{1}{\rho_i\sigma_i}\log n\right)$ we have $\bar{p}_{u,v,i} = \mathcal{O}(p_{u,v})$.

Corollary 1 allows computing 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 $\rho_1, \rho_2, \sigma_1, \sigma_2, w_1, \ldots, w_n, G_1$ and G_2, and reports estimations $\bar{p}_{u,v,i}$ for all $u, v \in V$ and $i \in \{1, 2\}$ 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.

To argue formally that this is possible, let the random variable X be the (set of) edge probability bounds computed by the above method, and denote by $A(\widetilde{\mathbf{p}})$ the event that our de-anonymization algorithm has a certain quality (like making no wrong identifications and identifying a constant fraction of the vertices) given edge probability estimations $\widetilde{\mathbf{p}}$. Let $\widetilde{P}$ be the set of edge probability bounds satisfying the guarantees in Corollary 1. We assume that for any (adversarially chosen) weight bounds $\widetilde{\mathbf{p}} \in \widetilde{P}$ we have $\Pr[\neg A(\widetilde{\mathbf{p}})] = n^{-\Omega(1)}$. Then plugging our estimation method into our de-anonymization algorithm yields

$$\Pr[\neg A(X)] = \sum_{\widetilde{\mathbf{p}}} \Pr[X = \widetilde{\mathbf{p}}] \cdot \Pr[\neg A(\widetilde{\mathbf{p}}) \mid X = \widetilde{\mathbf{p}}].$$

Now, since our weight estimation method X can be seen as an adversary, we have $\Pr[\neg A(\widetilde{\mathbf{p}}) \mid X = \widetilde{\mathbf{p}}, \widetilde{\mathbf{p}} \in \widetilde{P}] = n^{-\Omega(1)}$. Hence, we have

$$\Pr[\neg A(X)] = n^{-\Omega(1)} + \Pr[X \notin \widetilde{P}] = n^{-\Omega(1)},$$

where the second step uses that our estimation method X satisfies the guarantees $\widetilde{P}$ with high probability.

4 Matching Phase

In the matching phase we assume that we know the identity of some vertices $V_I \subseteq V$ containing the $h = \Omega(\log(n)/\rho_1\rho_2)$ highest weight identifiable nodes, 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

Fix a set V_I of 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 us to check whether two nodes are identical based on their signatures. W.h.p. the test does not identify two non-identical vertices 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 event that no edge is present. Based on these cases we now define

$$Y_u = Y_u^{v_1,v_2} := \begin{cases} \frac{1}{2\bar{p}_{v_1,u,1}}, & \text{if } u \in S_1^{v_1} \cap S_2^{v_2} \quad (A_u) \\ 1 - \frac{\rho_2}{2}, & \text{if } u \in S_1^{v_1} \text{ and } u \notin S_2^{v_2} \quad (B_u^1) \\ 1 - \frac{\rho_1}{2}, & \text{if } u \notin S_1^{v_1} \text{ and } u \in S_2^{v_2} \quad (B_u^2) \\ 1, & \text{otherwise} \quad (C_u). \end{cases}$$

and

$$Y := Y^{v_1,v_2} := \prod_{u \in V_I} Y_u^{v_1,v_2}.$$

Intuitively, $\log(Y_u)$ encodes the evidence of $v_1 = v_2$ given the connections to the identified node u; a common neighbor (A_u) has a positive evidence, $\log(Y_u) > 0$, while a node u connected to only one of the two (B_u^1, B_u^2) has a negative evidence, $\log(Y_u) < 0$. Note that when $v_1 = v_2$ we have $\bar{p}_{v_1,u,1} \approx \bar{p}_{v_2,u,2}$, 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. We prove this intuition in the remainder of this section. First we show that Y is not too large if $v_1 \neq v_2$ (w.h.p.). To this end, we verify $\mathbb{E}[Y_u] \leq 1$, then the statement follows from independence of the edges and Markov's inequality.

Lemma 5 *For any $v_1 \neq v_2 \in V \setminus V_I$ and $t > 0$ we have* $\Pr[Y > t] \leq 1/t$.

Proof For any $v_1 \neq v_2$ by considering the process that yields G_1 and G_2 we can compute

$$\begin{aligned}
\Pr[A_u] &= \rho_1\, p_{v_1,u} \cdot \rho_2\, p_{v_2,u},\\
\Pr[B_u^1] &= \rho_1\, p_{v_1,u}\,(1 - \rho_2\, p_{v_2,u})\\
\Pr[B_u^2] &= \rho_2\, p_{v_2,u}\,(1 - \rho_1\, p_{v_1,u}).
\end{aligned}$$

Note that

$$\begin{aligned}
\mathbb{E}[Y_u] &= \tfrac{1}{2\bar{p}_{v_1,u,1}} \Pr[A_u] + (1 - \tfrac{\rho_2}{2}) \Pr[B_u^1] + (1 - \tfrac{\rho_1}{2}) \Pr[B_u^2] + \Pr[C_u]\\
&= 1 + \big(\tfrac{1}{2\bar{p}_{v_1,u,1}} - 1\big) \Pr[A_u] - \tfrac{\rho_2}{2} \Pr[B_u^1] - \tfrac{\rho_1}{2} \Pr[B_u^2].
\end{aligned}$$

Plugging in $\Pr[A_u]$ and $\Pr[B_u]$, using $p_{v_1,u} \leq \bar{p}_{v_1,u,1}$ and ρ_i, $p_{v_i,u} \leq 1$ yields

$$\begin{aligned}
\mathbb{E}[Y_u] &\leq 1 + \tfrac{1}{2}\rho_1\rho_2(p_{v_2,u} - 2p_{v_1,u}p_{v_2,u} - p_{v_1,u} - p_{v_2,u} + (\rho_1 + \rho_2)p_{v_1,u}p_{v_2,u})\\
&\leq 1 + \tfrac{1}{2}\rho_1\rho_2(-2p_{v_1,u}p_{v_2,u} - p_{v_1,u} + 2p_{v_1,u}p_{v_2,u})\\
&\leq 1.
\end{aligned}$$

By independence of the edges we have $\mathbb{E}[Y] = \prod_{u \in V_I} \mathbb{E}[Y_u] \leq 1$. Markov's inequality now yields the desired bound $\Pr[Y > t] \leq 1/t$. □

The next lemma can be used to show that our test allows identifying 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 $\bar{p}_{v,u,1} = \Omega(1)$, which can be close to 1. These bad nodes do not provide sufficient independence for different nodes, as two indicator random variables with expectation close to 1 are very correlated. Thus, they do not help in separating different vertices. The requirement of the lemma essentially means that the expected number of neighbors is by $\Omega(\log n)$ more than the number of bad nodes, and thus we have $\Omega(\log n)$

neighbors that each give us an "independent bit" for identifying a node, which is enough information to be correct (w.h.p.).

Lemma 6 *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 $p_{u,v} \geq b > 0$, b being a sufficiently small constant. Assume that for $c > 0$ we have*

$$\sum_{u \in V_I} p_{u,v} = \Omega\big(\tfrac{1}{\rho_1\rho_2} c \log n + |B|\big)$$

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

Proof Let $Y = Y^{v,v}$. We use Markov's inequality and independence to show

$$\Pr[Y \leq n^c] = \Pr[Y^{-1/2} \geq n^{-c/2}] \leq n^{c/2}\mathbb{E}[Y^{-1/2}] = n^{c/2} \prod_{u \in V_I} \mathbb{E}[Y_u^{-1/2}].$$

It remains to show $\prod_{u \in V_I} \mathbb{E}[Y_u^{-1/2}] \leq n^{-3c/2}$ to prove the claim. Similarly to the proof of Lemma 5 we compute the probability that one of the events A_u, B_u^1 or B_u^2 occurs. As opposed to Lemma 5, we consider the Y-value for the vertex with the same ID. Therefore, we only have to take into account the probability that the edge $\{u, v\}$ occurs once.

$$\Pr[A_u] = \rho_1\rho_2 p_{v,u}, \quad \Pr[B_u^1] = \rho_1(1-\rho_2)p_{v,u}, \quad \Pr[B_u^2] = \rho_2(1-\rho_1)p_{v,u}.$$

Plugging this into the definition of $\mathbb{E}[Y_u^{-1/2}]$, we obtain

$$\begin{aligned}
\mathbb{E}[Y_u^{-1/2}] = {} & \big(2\bar{p}_{v,u,1}\big)^{1/2} \Pr[A_u] + \big(1 - \tfrac{\rho_2}{2}\big)^{-1/2} \Pr[B_u^1] \\
& + \big(1 - \tfrac{\rho_1}{2}\big)^{-1/2} \Pr[B_u^2] + \Pr[C_u] \\
= {} & 1 + \rho_1\rho_2 p_{v,u} \Big(\big(\sqrt{2\bar{p}_{v,u,1}} - 1\big) \\
& + \tfrac{1-\rho_2}{\rho_2} \Big(\big(1 - \tfrac{\rho_2}{2}\big)^{-1/2} - 1 \Big) + \tfrac{1-\rho_1}{\rho_1} \Big(\big(1 - \tfrac{\rho_1}{2}\big)^{-1/2} - 1 \Big) \Big).
\end{aligned}$$

Bounding $\frac{1-x}{x}\left((1-\frac{x}{2})^{-1/2} - 1\right) \leq \frac{1}{4} - \frac{x}{8}$ for $0 \leq x \leq 1$, we obtain

$$\begin{aligned}
\mathbb{E}[Y_u^{-1/2}] &\leq 1 + \rho_1\rho_2 p_{v,u} \big((2\bar{p}_{v,u,1})^{1/2} - 1 + \tfrac{1}{2} - (\rho_1 + \rho_2)/8 \big) \\
&\leq 1 + \rho_1\rho_2 p_{v,u} \big((2\bar{p}_{v,u,1})^{1/2} - \tfrac{1}{2} \big).
\end{aligned}$$

If $\bar{p}_{v,u,1}$ is less than a sufficiently small constant (which happens if and only if $p_{u,v}$ is less than a sufficiently small constant) then we have

$$\mathbb{E}[Y_u^{-1/2}] \leq 1 - \Omega(\rho_1\rho_2 p_{v,u}) = \exp(-\Omega(\rho_1\rho_2 p_{v,u})).$$

For nodes u with larger $p_{u,v}$, i.e. the nodes in B, using $p_{u,v}$, $\bar{p}_{u,v,1} \leq 1$ we obtain the weaker bound

$$\mathbb{E}[Y_u^{-1/2}] \leq \exp(\rho_1\rho_2 - \Omega(\rho_1\rho_2 p_{v,u})).$$

Hence,

$$\prod_{u\in V_I} \mathbb{E}[Y_u^{-1/2}] \leq \exp\Big(\rho_1\rho_2\Big(|B| - \Omega\Big(\sum_{u\in V_I} p_{u,v}\Big)\Big)\Big).$$

The assumption now yields the desired bound $\prod_{u\in V_I} \mathbb{E}[Y_u^{-1/2}] \leq n^{-3c/2}$. □

The requirement of the above lemma can only hold if the expected degree is $\Omega(\log n)$. Hence, we need a different lemma for small-weight vertices. In this case, we have to take a closer look at $Y^{v,v}$. In the following lemma, the neighbors that yield identifying bits are the neighbors in the set T. The error probability is essentially exponentially small in the expected number μ of neighbors in T.

Lemma 7 *Let $c > 0$ and consider an unidentified vertex $v \in V_\cap \backslash V_I$ with $w_v \leq n^{o(1)}$. 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 $\mu := \rho_1\rho_2\varepsilon|T|$ is at least a sufficiently large constant (depending only on c and β). Then we have*

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

Proof Note that $w_v \leq n^{o(1)}$ implies $p_{u,v} = n^{-\Theta(1)}$ *for any* $u \in V$. In particular, we have no bad nodes (in the sense of Lemma 6).

First, consider the nodes in $V_I\backslash T$ and let $Y' := \prod_{u\in V_I\backslash T} Y_u^{v,v}$. Similar to the proof of Lemma 6, we can show that $\mathbb{E}[Y_u^{-1/2}] = 1 - \Omega(\rho_1\rho_2 p_{v,u}) \leq 1$ so that by Markov's inequality and independence

$$\Pr[Y' \leq n^{-2c}] = \Pr[(Y')^{-1/2} \geq n^c] \leq n^{-c}\mathbb{E}[(Y')^{-1/2}] = n^{-c} \prod_{u\in V_I\backslash T} \mathbb{E}[Y_u^{-1/2}],$$

which is bounded by n^{-c}. Hence, the contribution of the vertices in $V_I\backslash T$ to Y is a factor of at least n^{-2c} with probability $1 - n^{-c}$.

It remains to show that for $Y_T := \prod_{u\in T} Y_u^{v,v}$ we have $\Pr[Y_T > n^{3c}] \geq 1 - \exp(-\Omega(\mu))$, then in total we obtain $\Pr[Y > n^c] \geq 1 - n^{-c} - \exp(-\Omega(\mu))$.

Let the random variable X denote the number of vertices in T that are adjacent to v in both G_1 and G_2, $X := |T\cap(N_1(v)\cap N_2(v))|$. Similarly, let $X_1 := |T\cap(N_1(v)\backslash N_2(v))|$ and $X_2 := |T \cap (N_2(v)\backslash N_1(v))|$. Note that $X_1 \preceq \text{Bin}\big(|T|, \mathcal{O}(\rho_1\varepsilon)\big)$ and $X_2 \preceq \text{Bin}\big(|T|, \mathcal{O}(\rho_2\varepsilon)\big)$. The Chernoff bound now yields $X_i = \mathcal{O}(|T|\rho_i\varepsilon)$ with probability at least $1-\exp(-\Omega(|T|\rho_i\varepsilon)) = 1-\exp(-\Omega(\mu))$. In this case, we obtain for $i \in \{1, 2\}$

$$(1 - \tfrac{\rho_{3-i}}{2})^{X_i} \geq \exp(-\mathcal{O}(\rho_1\rho_2|T|\varepsilon)) = \exp(-\mathcal{O}(\mu)).$$

Similarly, we have $X \succeq \mathrm{Bin}\left(|T|, \Omega(\rho_1\rho_2\varepsilon)\right)$. Since $\mathrm{Bin}(n, p) = \Omega(np)$ holds with probability at least $1 - \exp(-\Omega(np))$, we have $X = \Omega(\mu)$ with probability at least $1 - \exp(-\Omega(\mu))$. In total, we obtain with probability at least $1 - \exp(-\Omega(\mu))$

$$Y_T = n^{\Theta(1)\cdot X}(1 - \tfrac{\rho_2}{2})^{X_1}(1 - \tfrac{\rho_1}{2})^{X_2} \geq n^{\Omega(\mu)}\mathrm{e}^{-\mathcal{O}(\mu)} \geq n^{3c},$$

since μ is at least a sufficiently large constant (and for n large enough). □

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 algorithm 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 identifiable 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 $c > 0$ is 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.

Algorithm 1 De-anonymization using Y-tests

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

Note that this algorithm is oblivious to the node subsampling probabilities σ_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

$$\mathcal{O}\left(\log n \sum_{v_1,v_2} (\deg_1(v_1) + \deg_2(v_2))\right) = \mathcal{O}(nm \log n),$$

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 Sect. 5 how to decrease this to quasilinear runtime.

Using Lemma 5 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 8 *Algorithm 1 does not identify any two non-identical vertices with probability more than* $1 - \mathcal{O}(n^{2-c} \log n)$.

Proof Fix two non-identical vertices $v_1 \neq v_2$, $v_1 \in V_1$, $v_2 \in V_2$. Then the probability of identifying them in round r, i.e., the probability of $Y^{v_1,v_2} > n^c$, is at most n^{-c} by Lemma 5. Thus, the probability of identifying v_1 and v_2 in any round is at most $\sum_{r=1}^{\mathcal{O}(\log n)} n^{-c} = \mathcal{O}(n^{-c} \log n)$. The claim now follows from a union bound over all pairs of non-identical vertices $v_1 \in V_1$, $v_2 \in V_2$.

Note that here we use independence between the event of a node u, $v_1 \neq u \neq v_2$, being already identified and the edges among v_1, v_2, u being present, in order to apply Lemma 5. This independence holds because the decision of identifying u is made purely by looking at its edges to the already identified vertices, not at the ones to v_1, v_2. □

4.3 Quality Analysis

It remains to show that the algorithm identifies most vertices. We do this by examining the propagation of identified vertices in the graphs. For this, we define $L_j := \{2^{j-1}, \ldots, 2^j - 1\}$, $j = 1, \ldots, \log n$ as the jth layer of vertices. Note that this splits the nodes according to their weight. Indeed, from the definition of the weights w_v we see that all nodes in layer L_j have weight $\Theta(\delta(n/2^j)^{1/(\beta-1)})$. Thus, the sizes of the layers are exponentially increasing with j, while the weights of all nodes in a layer are roughly the same and exponentially decreasing with j. Moreover, 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 Chernoff bound we have $|\tilde{L}_j| = \Omega(\sigma_1\sigma_2|L_j|)$.

We proceed in three steps. First, we show that given the seed nodes, there is some layer k that gets identified with high probability. We choose k such that the estimated edge probability $\bar{p}_{v,h,i}$ of every vertex $v \in L_k$ with the hth highest weight identifiable vertex, for any $i \in \{1, 2\}$, is at most a sufficiently small constant, and k is minimal with this property. We show that this specific layer k satisfies $|L_k| \geq n^{3-\beta-o(1)}$, which in particular implies $|\tilde{L}_k| = \Theta(\sigma_1\sigma_2|L_k|)$. 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 $1 - \alpha_j \geq 1 - \exp(-\Omega(\rho_1\rho_2\sigma_1\sigma_2\delta))$. This holds independent 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 note that these vertices could already be identified 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 high-degree 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.

4.4 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 $h = \gamma^2 \frac{1}{\rho_1\rho_2} \log n$ where γ is a sufficiently large constant, and let h^* be the hth highest weight identifiable vertex. W.h.p. we have $h^* = \Theta(\frac{h}{\sigma_1\sigma_2})$. Let $\ell := \gamma \frac{1}{\sigma_1\sigma_2\rho_1\rho_2} \log n$. Choose a layer L_k such that for any $v \in L_k$ we have $p_{v,\ell} \le b$, where $b = \Theta(1)$ is the constant from Lemma 6, so that w.h.p. we have $|B| = \mathcal{O}(\sigma_1\sigma_2\ell) = \mathcal{O}(\gamma \frac{1}{\rho_1\rho_2} \log n)$ *bad nodes*. For any $v \in L_k$, for our weight distributions one can show that

$$p_{v,h^*} = p_{v,\ell} \cdot (\ell/h^*)^{1/(\beta-1)} = \Theta\left(\gamma^{-1/(\beta-1)}\right).$$

Hence, any node $1 \le u \le h^*$ has $p_{v,u} \ge p_{v,h^*} = \Theta(\gamma^{-1/(\beta-1)})$. Summing up over all $h = |V_I|$ given seed nodes, we have

$$\sum_{u\in V_I} p_{v,u} \ge \Omega\left(\gamma^{2-1/(\beta-1)} \frac{1}{\rho_1\rho_2} \log n\right)$$

Since $\gamma^{2-1/(\beta-1)} = \gamma^{1+\Omega(1)}$ and γ is sufficiently large, for any arbitrarily large hidden constant we have

$$\sum_{u\in V_I} p_{v,u} = \Omega\left((\gamma + c)\frac{1}{\rho_1\rho_2} \log n\right) = \Omega\left(\frac{1}{\rho_1\rho_2} c \log n + |B|\right),$$

which proves that the assumption of Lemma 6 is fulfilled and we identify v in the first round with high probability.

4.5 Second Step

Consider any following level $k < j \le \log n - \Omega(\log\log n)$ (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 $\bar{p}_{u,v,1} = \mathcal{O}\big(p_{u,v} + \frac{1}{\delta n} \log^2 n\big)$. Note that since $j \le \log n - \Omega(\log\log n)$ we have $w_v = \log^{\Omega(1)} n$ so that $\bar{p}_{u,v,1} = \mathcal{O}(p_{u,v})$. We can apply Lemma 6 with $|B| \le \mathcal{O}(\frac{1}{\rho_1\rho_2} \log n)$ (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 $|\tilde{L}_{j-1}| = \Omega(\sigma_1\sigma_2 2^j)$ we obtain

$$\sum_{u\in V_I} p_{u,v} \geq \Omega\big(\sigma_1\sigma_2 2^j \delta n^{(3-\beta)/(\beta-1)}\, 2^{-2j/(\beta-1)}\big) = \Omega\big(\sigma_1\sigma_2\delta \log^{\Omega(1)} n\big),$$

which is larger than $\Omega(\frac{1}{\rho_1\rho_2}\log n)$ since $\rho_1\rho_2\sigma_1\sigma_2\delta$ is at least a sufficiently large constant. Hence, Lemma 6 implies that we identify all vertices in $\tilde{L}_j$ with high probability.

For $\log n - o(\log n) \leq j \leq \log n$, so that $w_v = n^{o(1)}$, we instead use Lemma 7 to show that any vertex $v \in \tilde{L}_j$ is identified after round $j-k+1$ with a probability of at least $1-\alpha_j \geq 1-\exp(-\Omega(\rho_1\rho_2\sigma_1\sigma_2\delta))$. We again consider the edges of v into $T := V_I \cap \tilde{L}_{j-1}$ and obtain

$$\mu := \rho_1\rho_2\varepsilon|T| = \Omega\big(\rho_1\rho_2\sigma_1\sigma_2 2^j \delta n^{(3-\beta)/(\beta-1)}\, 2^{-2j/(\beta-1)}\big) = \Omega(\rho_1\rho_2\sigma_1\sigma_2\delta).$$

Hence, the assumption of Lemma 7 amounts to $\rho_1\rho_2\sigma_1\sigma_2\delta$ being at least a sufficiently large constant, and we identify each vertex in $\tilde{L}_j$ with probability at least $1-\alpha_j := 1-\exp(-\Omega(\mu)) - n^{-c} \geq 1-\exp(-\Omega(\rho_1\rho_2\sigma_1\sigma_2\delta))$.

4.6 Third Step

Now we identified most vertices in layers $\tilde{L}_j$, $j \geq k$ within the first $\log(n)-k+1$ rounds with high probability. It remains to show that we identify any vertex $v \in \tilde{L}_j$, $j < k$, at the latest in round $\log(n)-k+2$ with high probability. Such a vertex v has a probability to connect to any vertex $u \in \tilde{L}' := \tilde{L}_{\log n}$ of $\Theta(w_v/n) = \Theta(\delta n^{(2-\beta)/(\beta-1)} 2^{-j/(\beta-1)})$, and there are $\Theta(\sigma_1\sigma_2 n)$ identified vertices in $\tilde{L}'$ with high probability. Summing up over these vertices, we get

$$\sum_{u\in V_I} p_{u,v} = \Omega(\sigma_1\sigma_2 w_v) = \Omega\left(\sigma_1\sigma_2\delta n^{1/(\beta-1)} 2^{-j/(\beta-1)}\right) = \Omega\left((n/2^j)^{1/(\beta-1)}\right),$$

where the last step follows from $\rho_1\rho_2\sigma_1\sigma_2\delta \leq \sigma_1\sigma_2\delta$ being at least a sufficiently large constant. Moreover, any vertex u having probability larger than a constant to connect to v (i.e. a bad vertex) has weight at least $w_u = \Omega(W/w_v) = \Omega(n^{(\beta-2)/(\beta-1)}\, 2^{j/(\beta-1)})$. The number of such vertices is

$$|B| = \mathcal{O}\left(\delta^{\beta-1} n^{3-\beta}\, 2^{-j}\right).$$

Since $3-\beta < 1/(\beta-1) < 1$ for $2 < \beta < 3$, we have $\sum_{u\in V_I} p_{u,v} \gg |B| + \frac{1}{\rho_1\rho_2}\log n$ so that can apply Lemma 6 and thereby identify layer $\tilde{L}_j$ with high probability.

This finishes the analysis of the algorithm. We proved that for all layers $\tilde{L}_j$ each vertex $v \in \tilde{L}_j$ is identified in $\mathcal{O}(\log n)$ rounds with probability at least $1-\alpha_j \geq 1-\exp(-\Omega(\rho_1\rho_2\sigma_1\sigma_2\delta))$. Moreover, w.h.p. we identify at least a $1-\exp(-\Omega(\rho_1\rho_2\sigma_1\sigma_2\delta))$ fraction of the identifiable vertices.

In total, we showed the following.

Theorem 2 *Assume we are given the identity of a set V_I of vertices containing the h highest weight identifiable nodes, where $h = \Omega(\frac{1}{\rho_1\rho_2}\log n)$ with sufficiently large hidden constant. Then Algorithm 1 w.h.p. makes no wrong identifications and successfully matches a fraction of $1 - \exp(-\Omega(\rho_1\rho_2\sigma_1\sigma_2\delta))$ of the identifiable vertices. It runs in time $\mathcal{O}(nm\log n)$.*

In the next section we show how to decrease the running time to quasilinear, at the cost of increasing the number of highest-weight seed nodes to n^ε.

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 [16]. We assume to have identified the $h = n^{2\varepsilon}$ highest weight identifiable vertices for any constant $\varepsilon > 0$. We first ignore the issue of weights being estimated by an adaptive adversary (and just assume that we are given upper and lower bounds for all weights as in Lemma 3 without revealing any edges), later we sketch how to make our algorithm formally correct in the adaptive adversary case.

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 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,

$$T_{v,i} := \{u \in V_I \mid \bar{p}_{v,u,i} \le n^{-\varepsilon}\}.$$

We compute the first C/ε vertices $(u_1, \ldots, u_{C/\varepsilon}) =: M_v^i$ in $N_i(v)\cap T_{v,i}$ with respect to the order π (for some constant $C \ge 2$ 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 $\bigcup_v T_{v,i}$), so that the first neighbor of v is simply the first entry of its adjacency list. In the analysis we show that for a *good* (see Eq. 1 in Sect. 5.2 for a definition) vertex $v \in V_\cap$ the sets M_v^1, M_v^2 are equal with probability $\Theta((\rho_1+\rho_2)^{\Theta(1/\varepsilon)})$, while for non-identical vertices $v_1 \ne v_2$ 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 $m = \mathcal{O}((\rho_1+\rho_2)\delta n) = \mathcal{O}(\delta n)$.

As we will see in Sect. 5.2, roughly the same quality analysis as in the last section goes through, with the necessary number of rounds growing to $\Theta((\min\{\rho_1,\rho_2\})^{-\Theta(1/\varepsilon)}\log n)$. As $\varepsilon > 0$ is a constant, this evaluates to $\mathcal{O}(\min\{\rho_1,\rho_2\}^{-\mathcal{O}(1)}\log n)$. 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

Algorithm 2 Fast de-anonymization using Y-tests

Input: graphs G_1, G_2, identified vertices V_I containing the $n^{2\varepsilon}$ highest weight identifiable vertices
for $r = 1, 2, \ldots, \Theta((\rho_1 + \rho_2)^{-\Theta(1/\varepsilon)}) \log n$ **do**
  choose a random permutation π of V_I
  for all $v \in V \setminus V_I$ and $i \in \{1, 2\}$ **do**
    if $|N_i(v) \cap T_{v,i}| \geq C/\varepsilon$ **then**
      compute M_v^i, i.e., the first C/ε vertices in $N_i(v) \cap T_{v,i}$ 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**
    compute Y^{v_1,v_2}
    if $Y^{v_1,v_2} > n^c$ **then**
      $V_I := V_I \cup \{v_1, v_2\}$. ▷ We identified $v_1 = v_2$

probability grows to $\mathcal{O}(n^{2-c} \min\{\rho_1, \rho_2\}^{-\mathcal{O}(1)} \log n))$. In total we get an expected runtime of $\mathcal{O}(\min\{\rho_1, \rho_2\}^{-\mathcal{O}(1)} \delta n \log n)$.

5.1 Analysis of Hash Collisions

Lemma 9 *In any round of the above algorithm, consider vertices $v_1 \neq v_2$. A hash collision of the form $(v_1, 1)$ and $(v_2, 2)$ happens with probability:*

$$\Pr\left[M_{v_1}^1 = M_{v_2}^2\right] = \mathcal{O}(1/n).$$

In other words, the negative case of a hash collision between different vertices $v_1 \neq v_2$ always has small probability. This directly yields that the expected cost of Y-tests performed in one round is at most

$$\sum_{\substack{v_1 \neq v_2 \\ v_i \in V_i}} \mathcal{O}\left(\tfrac{1}{n}\right) (\deg_1(v_1) + \deg_2(v_2)) + \sum_{v \in V_\cap} \mathcal{O}(\deg_1(v) + \deg_2(v)) = \mathcal{O}(m).$$

Proof Let us analyze the probability of $v_1 \neq v_2$ forming a hash collision. Let $N_i := N_i(v_i) \cap T_{v_i,i}$. Since $M_{v_i}^i$ consists of the first C/ε vertices in N_i with respect to the random order π, we have

$$\Pr[M_{v_1}^1 = M_{v_2}^2] = \prod_{k=0}^{C/\varepsilon - 1} \frac{|N_1 \cap N_2| - k}{|N_1 \cup N_2| - k},$$

in particular this probability is 0 if $|N_1 \cap N_2| < C/\varepsilon$. In the following we ignore the fact that $\bar{w}_{v,i}$ (and thus the sets $T_{v_i,i}$) are chosen by an adaptive adversary. Doing so allows us to assume that any $u \in T_{v_i,i}$ is connected to v_i independently with probability $p_{v_i,u} \leq n^{-\varepsilon}$. See Sect. 5.3 for the details on how to handle this subtle issue.

First let $v_1 \neq v_2$, without loss of generality we can assume $v_1 < v_2$. Note that

$$|N_1 \cap N_2| \sim \sum_{u \in T_{v_1,1} \cap T_{v_2,2}} \mathrm{Bin}(\rho_1\, p_{v_1,u} \cdot \rho_2\, p_{v_2,u}) \preceq \sum_{u \in T_{v_1,1}} \mathrm{Bin}(\rho_1\, p_{v_1,u} \cdot n^{-\varepsilon}) =: X$$

$$|N_1 \cup N_2| \geq |N_1| \sim \sum_{u \in T_{v_1,1}} \mathrm{Bin}(\rho_1\, p_{v_1,u}) =: X'.$$

This allows us to show that $\Pr[M^1_{v_1} = M^2_{v_2}] = \mathcal{O}(\frac{1}{n})$ for $C \geq 2$ as follows. We do a case distinction on the expected size of X. If $\mathbb{E}[X] \leq n^{-\varepsilon/2}$, then by Chernoff bound (specifically the second bound from Lemma 1) we have

$$\Pr[X > C/\varepsilon] \leq \Big(\frac{\mathrm{e}\,\mathbb{E}[X]}{C/\varepsilon}\Big)^{C/\varepsilon} = \mathcal{O}(n^{-C/2}) = \mathcal{O}(\tfrac{1}{n}).$$

Hence, for such $v_1 < v_2$ the probability of having at all C/ε common neighbors is $\mathcal{O}(\frac{1}{n})$. If, on the other hand, $\mathbb{E}[X] > n^{-\varepsilon/2}$, then in particular $\mathbb{E}[X'] > n^{\varepsilon/2}$, so by Chernoff bound we observe that X' is well concentrated and we have $X' = \Omega(\mathbb{E}[X']) = \Omega(n^{\varepsilon}\mathbb{E}[X])$ with high probability. Moreover, again by Chernoff (specifically the second bound from Lemma 1) we have

$$\Pr[X > \tfrac{C}{\varepsilon} n^{\varepsilon/2}\mathbb{E}[X]] \leq \Big(\frac{\varepsilon \mathrm{e}}{C n^{\varepsilon/2}}\Big)^{\frac{C}{\varepsilon} n^{\varepsilon/2}\mathbb{E}[X]} = \Big(\frac{\mathcal{O}(1)}{n^{\varepsilon/2}}\Big)^{C/\varepsilon} = \mathcal{O}(n^{-\frac{C}{2}}) = \mathcal{O}(\tfrac{1}{n}).$$

Thus, with probability $1 - \mathcal{O}(\frac{1}{n})$ we have $X/X' = \mathcal{O}(n^{-\varepsilon/2})$, so that

$$\Pr[M^1_{v_1} = M^2_{v_2}] \leq \prod_{k=0}^{C/\varepsilon - 1} \frac{X-k}{X'-k} \leq \Big(\frac{X}{X'}\Big)^{C/\varepsilon} = \mathcal{O}(\tfrac{1}{n}).$$

In total we obtain $\Pr[M^1_{v_1} = M^2_{v_2}] = \mathcal{O}(\frac{1}{n})$. □

Lemma 10 *In any round of the above algorithm, consider a vertex v and let $N_i := N_i(v) \cap T_{v,i}$ for $i \in \{1, 2\}$. We say that v is* good *if*

$$|N_1 \cap N_2| \geq 2C/\varepsilon \quad \text{and} \quad \frac{|N_1 \cap N_2|}{|N_1 \cup N_2|} = \Omega(\min\{\rho_1, \rho_2\}). \tag{1}$$

If v is identifiable and good, a hash collision of the form $(v, 1)$ and $(v, 2)$ happens with probability:

$$\Pr[M^1_v = M^2_v] = \Omega((\min\{\rho_1, \rho_2\})^{C/\varepsilon}).$$

Note that this positive case of a hash collision between equal vertices $v_1 = v_2 = v$ does not always have large probability, but needs the additional technical assumption that the vertex is good.

Proof Let $v_1 = v_2 = v$ be identifiable and good. Then the probability of $(v_1, 1)$ and $(v_2, 2)$ forming a hash collision is

$$\begin{aligned}\Pr[M^1_{v_1} = M^2_{v_2}] &= \prod_{k=0}^{C/\varepsilon-1} \frac{|N_1 \cap N_2| - k}{|N_1 \cup N_2| - k} \\ &\geq \prod_{k=0}^{C/\varepsilon-1} \frac{|N_1 \cap N_2|}{2|N_1 \cup N_2|} \\ &= \Omega((\min\{\rho_1, \rho_2\})^{C/\varepsilon}).\end{aligned}$$

□

In order to see that the number of good vertices is high in expectation, we again first ignore the issue that we do not know the edge probabilities $p_{v,u}$ exactly, but only have upper bounds $\bar{p}_{v,u,1}$, $\bar{p}_{v,u,2}$ given by an adaptive adversary. For details on how to cope with this issue, see Sect. 5.3. Thus, assume for now that $\bar{p}_{v,u,1} = \bar{p}_{v,u,2} = p_{v,u}$ for all vertices u, v, in particular $T_{v,1} = T_{v,2}$. If v has an expected number of k neighbors in $T_{v,1} = T_{v,2}$, then $\mathbb{E}[|N_1 \cap N_2|] = \rho_1\rho_2 k$ and $\mathbb{E}[|N_1 \cup N_2|] = (1 - (1 - \rho_1)(1 - \rho_2))k = \mathcal{O}((\rho_1 + \rho_2)k)$. If $\rho_1\rho_2 k \gg C/\varepsilon$, by Chernoff with probability at least $1 - \exp(-\Omega(\rho_1\rho_2 k))$ the vertex v is good. In particular, if a constant fraction of the layer below v is identified, then we have $k = \Omega(\delta)$, so that v is good with probability $1 - \exp(-\Omega(\rho_1\rho_2\delta))$ (assuming that $\rho_1\rho_2\delta$ is sufficiently large). Thus, identifying only good vertices does not change the amount of identified vertices asymptotically.

5.2 Quality Analysis

In this section we sketch an analysis of the expected number of vertices identified by our algorithm. We again split up the analysis into three steps.

In the first step we start with at least the $n^{2\varepsilon}$ highest weight identifiable vertices being identified. We choose k such that every vertex v in layer $\tilde{L}_k$ satisfies

$$v = \Theta(\delta^{\beta-1} n^{3-\beta+\varepsilon(\beta-2)}),$$

with a sufficiently large hidden constant. It is easy to check that all but the n^{ε} highest weight identifiable vertices have probability less than $n^{-\varepsilon}$ to connect to v. Moreover, there are no vertices having at least constant probability to connect to v, and the last $n^{2\varepsilon}/2$ of the highest weight identifiable vertices have probability $\Theta(n^{-\varepsilon\beta/(\beta-1)}) \gg n^{-2\varepsilon}$ to connect to v. Thus, v has an expected number of $n^{\varepsilon'}$ neighbors in $T_{v,i}$ for some $\varepsilon' > 0$ and any $i \in \{1, 2\}$, specifically we have w.h.p. $|N_i(v) \cap T_{v,i}| > n^{\varepsilon''}$ for some $\varepsilon'' > 0$. This shows that every vertex $v \in L_k$ indeed gets hashed and is good. This also allows us to apply Lemma 6 meaning that any vertex in L_k whose two copies have a hash collision gets identified with high probability. Finally, note that every vertex in L_k has a hash collision with its copy within the first $\Theta((\min\{\rho_1, \rho_2\})^{-\Theta(1/\varepsilon)} \log n)$ rounds with high probability by Lemma 10. This shows that we identify all vertices in L_k w.h.p. within the first $\Theta((\min\{\rho_1, \rho_2\})^{-\Theta(1/\varepsilon)} \log n)$ rounds.

In the second step we assume that we have identified a constant fraction of some layer L_j, $j \geq k$. Then by Lemma 10 in $\Theta((\min\{\rho_1, \rho_2\})^{-\Theta(1/\varepsilon)})$ rounds we positively test a constant fraction of the vertices in the next layer L_{j+1}, so that after $\Theta((\min\{\rho_1, \rho_2\})^{-\Theta(1/\varepsilon)} \log n)$ rounds we have identified a constant fraction of all layers L_j, $j > k$. After another $\Theta((\min\{\rho_1, \rho_2\})^{-\Theta(1/\varepsilon)} \log n)$ rounds, for every good vertex $v \in L_j$ we tested its two copies in G_1 and G_2 (w.h.p.), implying that we identified all good vertices with a probability of at least $1 - \exp(-\Omega(\rho_1\rho_2\delta))$. As only a small fraction of the vertices is not good, this yields the same asymptotic guarantee on the number of identified vertices as from Algorithm 1.

The third step is again that all layers L_j, $j < k$, will be identified w.h.p. once we know a constant fraction of the vertices with constant expected degree. By the same argument as in the second step, after $\Theta((\min\{\rho_1, \rho_2\})^{-\Theta(1/\varepsilon)} \log n)$ rounds, a constant fraction of the vertices in $L_{\log n}$ is identified. Since we hash each vertex with its copy w.h.p. in the second $\Theta((\min\{\rho_1, \rho_2\})^{-\Theta(1/\varepsilon)} \log n)$ rounds, we therefore identify all vertices in layer j w.h.p. by the same argument as in Sect. 4.3.

5.3 Adaptive Adversary

Our sketched algorithm has one major technical difficulty. For analyzing the (random) sequence M_v^i in $N_i(v) \cap T_{v,i}$ we would like to assume that v has $u \in T_{v,i}$ as neighbor independently with probability $p_{u,v} \leq n^{-\varepsilon}$. However, the edge probability estimations (and, thus, the sets $T_{v,i}$) are chosen by an adaptive adversary. One could even imagine that among the vertices with $p_{u,v} \approx n^{-\varepsilon}$ the ones with $\bar{p}_{u,v,1} > n^{-\varepsilon}$ all have no edges to v and the remaining ones all have an edge to v, making them highly dependent. As we will see, this effect is insignificant to our algorithm as long as the number vertices u with $p_{u,v} \ll n^{-\varepsilon}$ is sufficiently large (intuitively since then the random sequence M_v^i does not contain too many vertices with $p_{u,v} \approx n^{-\varepsilon}$). We have to adjust the algorithm slightly as follows.

Recall that we defined $T_{v,i} := \{u \in V_I \mid \bar{p}_{u,v,i} \leq n^{-\varepsilon}\}$. As ε is a sufficiently small constant one can show that around $n^{-\varepsilon}$ the estimate $\bar{p}_{u,v,i}$ is a constant approximation to $p_{u,v}$. In particular we have $\underline{R}_v \subseteq T_{v,i} \subseteq \overline{R}_v$ for

$$\overline{R}_v := \{u \in V_I \mid p_{u,v} \leq n^{-\varepsilon}\},$$
$$\underline{R}_v := \{u \in V_I \mid p_{u,v} \leq c^{-1}n^{-\varepsilon}\},$$

for some constant $c \geq 1$. Similarly we obtain

$$\underline{T}_{v,i} \subseteq \underline{R}_v \subseteq T_{v,i} \subseteq \overline{R}_v \subseteq \overline{T}_{v,i},$$

where

$$\overline{T}_{v,i} := \{u \in V_I \mid \bar{p}_{u,v,i} \leq c'n^{-\varepsilon}\},$$
$$\underline{T}_{v,i} := \{u \in V_I \mid \bar{p}_{u,v,i} \leq c'^{-1}n^{-\varepsilon}\},$$

for some $c' \geq 1$. We set

$$\underline{s}_{v,i} := \sum_{u \in \underline{T}_{v,i}} \bar{p}_{u,v,i}, \qquad \overline{s}_{v,i} := \sum_{u \in \overline{T}_{v,i}} \bar{p}_{u,v,i}.$$

Now we change Algorithm 2 such that we hash vertex $v \in V_i$ only if

$$\underline{s}_{v,i} \geq n^{\varepsilon}(\overline{s}_{v,i} - \underline{s}_{v,i}). \tag{2}$$

To implement this check efficiently, we always store V_I and $V \setminus V_I$ in two sorted orders, with respect to $\bar{w}_{u,1}$ and $\bar{w}_{u,2}$, e.g., in balanced search trees that also allow computing partial sums of $\bar{w}_{u,1}$ and $\bar{w}_{u,2}$. Then we can quickly determine $\underline{s}_{v,i}, \overline{s}_{v,i}$, as this simply amounts to searching for the largest $u \in V_I$ with $\bar{p}_{v,u_i} = \bar{w}_{v,i}\bar{w}_{u,i}/\underline{W}_i \leq n^{-\varepsilon}$, computing the partial sum over $\bar{w}_{u,i}$ up to this point and multiplying by $\bar{w}_{v,i}/\underline{W}_i$. In fact, determining $\underline{s}_{v,i}, \overline{s}_{v,i}$ for all v can be done in time $\mathcal{O}(n)$.

In the following we sketch how the analysis of hash collisions changes. Consider a vertex v in G_i. The algorithm computes the first C/ε vertices M_v^i in $N_i(v) \cap T_{v,i}$. We want to show that most vertices in M_v^i are contained in $\underline{R}_v$. As this set does not use the weight estimations and does not reveal edges, the neighbors of v in $\underline{R}_v$ are nicely independently distributed. By condition (2), the expected number of neighbors of v in $\underline{R}_v$ is much larger than its expected number of neighbors in $\overline{R}_v \setminus \underline{R}_v$. Similar to the hash collision analysis in Sect. 5.1, one can show that with probability at least $1 - \mathcal{O}(\frac{1}{n})$ we have

$$|N_i(v) \cap (\overline{R}_v \setminus \underline{R}_v)| \leq 2/\varepsilon \quad \text{or} \quad |N_i(v) \cap \underline{R}_v| = \Omega(n^{\varepsilon/2}) \cdot |N_i(v) \cap (\overline{R}_v \setminus \underline{R}_v)|$$

(depending on $\mathbb{E}[|N_i(v) \cap (\overline{R}_v \setminus \underline{R}_v)|]$ being smaller or larger than $n^{-\varepsilon/2}$). In both cases, the first $C/\varepsilon = 4/\varepsilon$ vertices in $N_i(v) \cap T_{v,i}$ with respect to π contain at most $2/\varepsilon$ vertices in $T_{v,i} \setminus \underline{R}_v$ with probability $1 - \mathcal{O}(\frac{1}{n})$. The remaining vertices, at least $2/\varepsilon$, are randomly sampled from $N_i(v) \cap \underline{R}_v$, which are nicely independently distributed vertices on which we can do the analysis from Sect. 5.1. Repeating this analysis shows that for $v_1 \neq v_2$ the probability of a hash collision is at most $\mathcal{O}(\frac{1}{n})$.

For $v_1 = v_2 = v$ we note that as long as

$$|N_i(v) \cap \underline{R}_v| \geq 5/\varepsilon, \tag{3}$$

the whole sequence M_v^i is contained in $\underline{R}_v$ with probability $\Omega(1)^{4/\varepsilon} = \Omega(1)$. In this case we can again repeat the analysis from Sect. 5.1 showing that $v_1 = v_2$ form a hash collision with a probability of at least $\Theta(\min\{\rho_1, \rho_2\})^{-\Theta(1/\varepsilon)}$. Moreover, note that if a constant fraction of the layer below v is identified and $\rho_1\rho_2\delta$ is sufficiently large, then condition (3) holds with a probability of at least $1 - \exp(-\Omega(\rho_1\rho_2\delta))$. Thus, adding (3) to the definition of a good vertex suffices.

6 Lower Bound

In this section we show that our algorithm identifies an asymptotically optimal fraction of the vertices.

Theorem 3 *Consider a de-anonymization algorithm A that w.h.p. makes no wrong identifications. Then there is a constant $c > 0$ such that A identifies no more than a fraction of $1 - \exp(-c\rho_1\rho_2\sigma_1\sigma_2\delta)$ identifiable vertices in expectation.*

Proof We prove this statement by case distinction. Recall that $V_\cap := V_1 \cap V_2$ describes the identifiable vertices.

If $\rho_1\rho_2\sigma_1\sigma_2 \geq \frac{1}{2}$, we compute the expected number of isolated vertices in the original graph G. These vertices cannot be matched. Consider the vertices $V_s := \{\frac{n}{2}+1, \ldots, n\}$ and $\tilde{V}_s := V_s \cap V_\cap$. Observe that w.h.p. $|\tilde{V}_s| = \Omega(|V_\cap|)$. Using Lemma 11(1) below, a node $v \in \tilde{V}_s$ is isolated with probability at least $\exp(-\Omega(w_v))$. Together with the fact that $w_v = O(\delta)$ for $v \in V_s$, and the case assumption that $\rho_1, \rho_2, \sigma_1, \sigma_2$ are constant, a node $v \in \tilde{V}_s$ is isolated with probability at least

$$\exp(-\Omega(w_v)) = \exp(-\Omega(\delta)) = \exp(-\Omega(\rho_1\rho_2\sigma_1\sigma_2\delta)).$$

Hence, there are at least $n \cdot \exp(-\Omega(\rho_1\rho_2\sigma_1\sigma_2\delta))$ isolated vertices in G in expectation, which are impossible to distinguish.

If $\rho_1\rho_2\sigma_1\sigma_2 < \frac{1}{2}$, we compute the number of vertices that leave no common edge in the subsampled graphs. Consider the set of identifiable nodes among the $n/2$ lowest weight vertices, i.e. $\tilde{V}_s = V_\cap \cap V_s$. We claim that there exist $\Omega(|V_\cap|)$ nodes $u, v \in \tilde{V}_s$ such that u, v are consecutive (i.e. there is no $w \in \tilde{V}_s$ such that $u < w < v$) and $|u - v| = \mathcal{O}(\frac{1}{\sigma_1\sigma_2})$. To see this, observe that $|\tilde{V}_s| \geq \Theta(\sigma_1\sigma_2 n)$ w.h.p. by the Chernoff bound. Now, a simple counting argument shows that the number of consecutive nodes u, v with distance $|u - v| \geq \frac{c}{\sigma_1\sigma_2}$ can at most be $n/(c/\sigma_1\sigma_2) = c^{-1}\sigma_1\sigma_2 n$. Note that the weights of two such vertices u, v are $\Theta(\delta)$ and they differ by at most $\mathcal{O}(\delta/(\sigma_1\sigma_2 n))$.

Assume that u has no common neighbor with v in G, which happens at least with probability $1 - o(w_u w_v) = 1 - o(1)$ by Lemma 11(2) below. Now we assume that u, v have $\mathcal{O}(\delta)$ neighbors in G which happens at least with constant probability, as we assumed $\delta = \Omega(1)$. Then, the chance that u (and v, respectively) has no edge that stays in both G_1, G_2 is $(1 - \rho_1\rho_2\sigma_1\sigma_2)^{\mathcal{O}(\delta)} = \exp(-\mathcal{O}(\rho_1\rho_2\sigma_1\sigma_2\delta))$.

Overall, these assumptions hold with probability at least $\exp(-\mathcal{O}(\rho_1\rho_2\sigma_1\sigma_2\delta))$. Intuitively, in this case it is impossible to distinguish u and v (at least this is not possible with a high probability guarantee), meaning that both u and v cannot be identified by any algorithm. To make this formal, let N_1^u be the neighbors of u in G_1, and similarly define N_2^u, N_1^v, N_2^v. Then, under the assumptions we have made, the probability of observing neighbor sets $(N_1^u, N_2^u, N_1^v, N_2^v)$ is approximately the same as the probability of observing $(N_1^u, N_2^v, N_1^v, N_2^u)$, namely their quotient is

$$\prod_{i \in U_2} \frac{p_{u,i}}{p_{v,i}} \cdot \prod_{i \in V_2} \frac{p_{v,i}}{p_{u,i}} = \left(1 \pm \mathcal{O}(\delta/(\sigma_1\sigma_2 n))\right)^{|U_2|+|V_2|} = 1 \pm \mathcal{O}(\delta^2/(\sigma_1\sigma_2 n)) = 1 \pm o(1).$$

Hence, if we swap the neighborhoods of u and v in G_2 to obtain G_2', then the probabilities of the instances (G_1, G_2) and (G_1, G_2') are approximately equal. Let $A(G_1, G_2, u)$ be any algorithm that outputs a match for node u, and assume the above conditions hold. Then

$$\Pr[A(G_1, G_2, u) = u] = (1 \pm o(1)) \Pr[A(G_1, G_2', u) = v],$$

as G_2 and G_2' are isomorphic. This implies that A matches correctly at most with probability $1/2 + o(1)$, and therefore not with high probability.

As u and v were two arbitrary consecutive identifiable vertices with small weight, we observe that an expected fraction of $\exp(-\mathcal{O}(\rho_1\rho_2\sigma_1\sigma_2\delta))$ of the vertices cannot be identified, proving the claim. □

Assume that u has no common neighbor with v in G, which happens at least with probability

$$1 - \sum_{i \in V \setminus \{u,v\}} p_{u,i}\, p_{v,i} = 1 - \Theta\left(\sum_{i \in V \setminus \{u,v\}} w_i^2/n^2\right) = 1 - \Theta\left(\delta n^{-\frac{2\beta-4}{\beta-1}}\right) = 1 - o(1).$$

Lemma 11 *Consider a Chung–Lu graph $G = (V, E)$. (1) Any node v is isolated with probability at least $\exp(-\Omega(w_v))$. (2) Vertices $u \neq v$ have no common neighbor with probability at least $1 - o(w_u w_v)$.*

Proof For (1), a node v is isolated with probability at least

$$\prod_{u \in V} \left(1 - \mathcal{O}\Big(\frac{w_u w_v}{W}\Big)\right) \geq \exp\Big(-\Omega\Big(\sum_{u \in V} \frac{w_u w_v}{W}\Big)\Big) \geq \exp(-\Omega(w_v)).$$

For (2), vertices $u \neq v$ have no common neighbor with probability

$$1 - \sum_{i \in V \setminus \{u,v\}} p_{u,i}\, p_{v,i} = 1 - \Theta\Big(w_u w_v \sum_{i \in V \setminus \{u,v\}} w_i^2/n^2\Big) = 1 - \Theta\left(w_u w_v \delta n^{-\frac{2\beta-4}{\beta-1}}\right),$$

which is $1 - o(w_u w_v)$. □

7 Experiments

The focus of this paper lies on theoretical algorithm analysis. To check our theory for robustness, however, we conducted an 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.

We evaluated the algorithm on Chung–Lu graphs. The results are shown in Fig. 1. We choose as seeds the nodes with the highest degree in the graph. The plots indicate that our quality bounds hold up well in practice, as we identify 95% of the nodes with

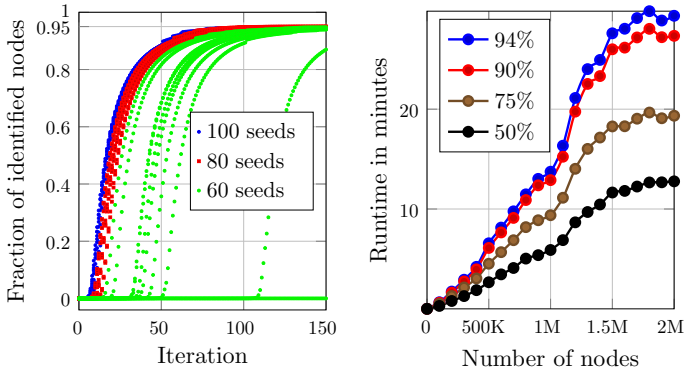

Fig. 1 Algorithm performance on Chung–Lu graphs with $\beta = 2.5$, $\delta = 30$ and subsampling probabilities $p_i = 0.5$, $q_i = 0.85$. The left plot shows recall vs. iterations for 30 different runs with $n = 10^6$ nodes. The precision is $1 - 10^{-5}$ for all runs. The right plot shows the runtime needed to identify a certain fraction of the nodes (see legend)

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 of $n^{1+\Omega(1)}$.

Finally, we investigated the robustness of our algorithm with respect to the underlying graph model. We ran it on different random graph models (Preferential Attachment [7], Affiliation Networks[5] [19]) and even subsampled real graphs (Facebook,[6] Orkut[7]). The results are presented in Table 1. Note that the purpose of these experiments is to demonstrate robustness to the underlying graph model, however, since the subsampling procedure still strictly follows our model, they have limited significance for real-world instances of the deanonymization problem.

We further point out that the Facebook network seems to be a particularly challenging instance for deanonymization, as evidenced by [18], who also performed experiments on Facebook and Affiliation Networks. Similar to us, they achieve a recall of 0.6 and 0.9, respectively. However, subtracting the substantial amount of seed nodes used in their approach (10% of the identifiable nodes) and only counting node pairs actually identified by the algorithm, we find that our approach ultimately identifies equally or more additional node pairs. 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

[5] 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.

[6] http://socialnetworks.mpi-sws.org/datasets.html.

[7] http://snap.stanford.edu/data/.

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

Graph model								
Name	n	m	$\rho_1 \cdot \rho_2$	$\sigma_1 \cdot \sigma_2$	Seeds	Recall	Prec.	Runtime
Chung–Lu	2M	80M	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 s
Facebook	63K	1.5M	0.58	1	50	0.50	0.95	6 s
Orkut	3M	117M	0.56	0.81	1000	0.89	0.88	54 min

Recall is defined as the fraction of identified nodes out of the $|V_1 \cap V_2|$ identifiable nodes. Precision is the fraction of correctly identified pairs over all identified pairs

(e.g. independent edge probabilities), the algorithm performs reasonably well also for other graph topologies.

8 Conclusion

We present a new method for de-anonymizing scale-free networks with two crucial improvements in comparison 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 asymptotic. Whereas recent experiments by Korula and Lattanzi [18] required large compute clusters, our algorithm can handle graphs with millions of vertices in less than an hour on a single machine. 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 approach needs only n^ε highest-weight seed nodes, while all previous algorithms with proven runtime and quality use $\Theta(n)$ random 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.

Acknowledgements We thank Silvio Lattanzi from Google Inc. for fruitful discussions, sharing their data sets, and sending us a preliminary version of [18] at the early stages of this project. Karl Bringmann is a recipient of the *Google Europe Fellowship in Randomized Algorithms*, and this research is supported in part by this Google Fellowship. Tobias Friedrich received funding from the German Research Foundation (DFG) under Grant Agreement No. FR 2988 (ADLON).

References

1. Aiello, W., Chung, F., Lu, L.: A random graph model for massive graphs. In: 32nd Annual ACM Symposium on Theory of Computing (STOC), pp. 171–180 (2000)
2. Aiello, W., Chung, F., Lu, L.: A random graph model for power law graphs. Exp. Math. **10**(1), 53–66 (2001)
3. Alon, N., Spencer, J.H.: The Probabilistic Method, 3rd edn. Wiley, Hoboken (2008)
4. Amini, H., Fountoulakis, N.: What I tell you three times is true: bootstrap percolation in small worlds. In: 8th International Workshop on Internet and Network Economics (WINE), pp. 462–474 (2012)
5. Arvind, V., Köbler, J., Kuhnert, S., Vasudev, Y.: Approximate graph isomorphism. In: 37th International Symposium on Mathematical Foundations of Computer Science (MFCS), pp. 100–111. Springer (2012)
6. Backstrom, L., Dwork, C., Kleinberg, J.: Wherefore art thou r3579x? Anonymized social networks, hidden patterns, and structural steganography. Commun. ACM **54**(12), 133–141 (2011)
7. Barabási, A.-L., Albert, R.: Emergence of scaling in random networks. Science **286**, 509–512 (1999)
8. Bringmann, K., Friedrich, T., Krohmer, A.: De-anonymization of heterogeneous random graphs in quasilinear time. In: 22nd European Symposium on Algorithms (ESA). Lecture Notes in Computer Science. Springer (2014)
9. Chung, F., Lu, L.: Connected components in random graphs with given expected degree sequences. Ann. Comb. **6**(2), 125–145 (2002)
10. Chung, F., Lu, L.: The average distances in random graphs with given expected degrees. Proc. Natl. Acad. Sci. (PNAS) **99**(25), 15879–15882 (2002)
11. Chung, F., Lu, L.: The average distance in a random graph with given expected degrees. Internet Math. **1**(1), 91–113 (2004)
12. Dereich, S., Mönch, C., Mörters, P.: Typical distances in ultrasmall random networks. Adv. Appl. Prob. **44**(2), 583–601 (2012)
13. Dwork, C.: Differential privacy. In: 33rd International Colloquium on Automata, Languages, and Programming (ICALP), pp. 1–12 (2006)
14. Fountoulakis, N., Panagiotou, K., Sauerwald, T.: Ultra-fast rumor spreading in social networks. In: 23rd Symposium Discrete Algorithms (SODA), pp. 1642–1660 (2012)
15. Friedrich, T., Krohmer, A.: Parameterized clique on scale-free networks. In: 23rd International Symposium on Algorithms and Computation (ISAAC), pp. 659–668 (2012)
16. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: 30th Annual ACM Symposium on Theory of Computing (STOC), pp. 604–613 (1998)
17. Kim, J.H., Vu, V.H.: Concentration of multivariate polynomials and its applications. Combinatorica **20**(3), 417–434 (2000)
18. Korula, N., Lattanzi, S.: An efficient reconciliation algorithm for social networks. In: 40th International Conference on Very Large Data Bases (VLDB), pp. 377–388 (2014)
19. Lattanzi, S., Sivakumar, D.: Affiliation networks. In: 41st Annual ACM Symposium on Theory of Computing (STOC), pp. 427–434 (2009)
20. McGregor, A., Mironov, I., Pitassi, T., Reingold, O., Talwar, K., Vadhan, S.P.: The limits of two-party differential privacy. In: 51th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 81–90 (2010)
21. Mitzenmacher, M.: A brief history of generative models for power law and lognormal distributions. Internet Math **1**(2), 226–251 (2004)
22. Narayanan, A., Shmatikov, V.: De-anonymizing social networks. In: 30th IEEE Symposium on Security and Privacy (SP), pp. 173–187 (2009)
23. Newman, I., Sohler, C.: Every property of hyperfinite graphs is testable. In: 43rd Annual ACM Symposium on Theory of Computing (STOC), pp. 675–684 (2011)
24. Newman, M.E.J.: The structure and function of complex networks. SIAM Rev. **45**(2), 167–256 (2003)
25. Novak, J., Raghavan, P., Tomkins, A.: Anti-aliasing on the web. In: 13th International Conference on World Wide Web (WWW), pp. 30–39 (2004)
26. Rao, J.R., Rohatgi, P.: Can pseudonymity really guarantee privacy? In: 9th USENIX Security Symposium (USENIX), pp. 85–96 (2000)
27. Sala, A., Zhao, X., Wilson, C., Zheng, H., Zhao, B.Y.: Sharing graphs using differentially private graph models. In: ACM SIGCOMM Conference on Internet Measurement Conference (IMC), pp. 81–98 (2011)

28. van der Hofstad, R.: Random graphs and complex networks. www.win.tue.nl/~rhofstad/NotesRGCN.pdf (2014)
29. Vijayraghavan, A., Wu, Y., Yoshida, Y., Zhou, Y.: Graph isomorphism: approximate and robust. Unpublished manuscript, available from the authors (2013)
30. Wondracek, G., Holz, T., Kirda, E., Kruegel, C.: A practical attack to de-anonymize social network users. In: IEEE Symposium on Security and Privacy (SP), pp. 223–238 (2010)
31. Zafarani, R., Liu, H.: Connecting corresponding identities across communities. In: 3rd International Conference on Weblogs and Social Media (ICWSM), pp. 354–357 (2009)
32. Zheleva, E., Getoor, L.: 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), pp. 531–540 (2009)