On the Complexity of Sampling Vertices Uniformly from a Graph

Flavio Chierichetti[1]
Dipartimento di Informatica, Sapienza University, Rome, Italy
flavio@di.uniroma1.it
https://orcid.org/0000-0001-8261-9058

Shahrzad Haddadan[2]
Dipartimento di Informatica, Sapienza University, Rome, Italy
shahrzad.haddadan@uniroma1.it
https://orcid.org/0000-0002-7702-8250

Abstract

We study a number of graph exploration problems in the following natural scenario: an algorithm starts exploring an undirected graph from some *seed* vertex; the algorithm, for an arbitrary vertex v that it is aware of, can ask an oracle to return the set of the neighbors of v. (In the case of social networks, a call to this oracle corresponds to downloading the profile page of user v.) The goal of the algorithm is to either learn something (e.g., average degree) about the graph, or to return some random function of the graph (e.g., a uniform-at-random vertex), while accessing/downloading as few vertices of the graph as possible.

Motivated by practical applications, we study the complexities of a variety of problems in terms of the graph's mixing time $\mathrm{t_{mix}}$ and average degree $\mathrm{d_{avg}}$ – two measures that are believed to be quite small in real-world social networks, and that have often been used in the applied literature to bound the performance of online exploration algorithms.

Our main result is that the algorithm has to access $\Omega\left(\mathrm{t_{mix}}\,\mathrm{d_{avg}}\,\epsilon^{-2}\ln\delta^{-1}\right)$ vertices to obtain, with probability at least $1-\delta$, an ϵ additive approximation of the average of a bounded function on the vertices of a graph – this lower bound matches the performance of an algorithm that was proposed in the literature.

We also give tight bounds for the problem of returning a close-to-uniform-at-random vertex from the graph. Finally, we give lower bounds for the problems of estimating the average degree of the graph, and the number of vertices of the graph.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Social Networks, Sampling, Graph Exploration, Lower Bounds

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.149

Related Version A full version of the paper is available at `https://arxiv.org/abs/1710.08815`.

Acknowledgements The authors would like to thank Anirban Dasgupta, Ravi Kumar, Silvio Lattanzi and Tamás Sarlós for several useful discussions.

[1] [Supported in part by the ERC Starting Grant DMAP 680153, by a Google Focused Research Award and by the SIR Grant RBSI14Q743.]
[2] [Supported in part by the ERC Starting Grant DMAP 680153, by a Google Focused Research Award and by the SIR Grant RBSI14Q743.]

1 Introduction

Hundreds of millions of people share messages, videos and pictures on Google+ and Facebook each day – these media have an increasingly high political, economical, and social importance in today's world. Data miners have consequently devoted significant amounts of attention to the study of large social networks.

In data mining, one often seeks algorithms that can return (approximate) properties of online social networks, so to study and analyze them, but without having to download the millions, or billions, of vertices that they are made up of. The properties of interest range from the order of the graph [18, 17], to its average degree (or its degree distribution) [10, 13, 16], to the average clustering coefficient [22, 23] or triangle counting [2], to non-topological properties such as the average score that the social network's users assign to a movie or a song, or to the fraction of people that *like* a specific article or page. All these problems have trivial solutions when the graph (with its non-topological attributes) is stored in main memory, or in the disk: choosing a few independent and uniform at random vertices from the graph, and computing their contribution to the (additive) property of interest, is sufficient to estimate the (unknown) value of the graph property – the empirical average of the contributions of the randomly chosen vertices will be close to the right value with high probability, by the central limit theorem.

In applications, though, it is often impossible to have random access to the (vertices of the) graph. Consider, for instance, an online undirected social graph, such as the Facebook friendship graph. An algorithm can *download* a webpage of a given (known) user `alice` from this social graph (e.g., `http://sn.com/user.php?id=alice`), parse the HTML, and get the URLs of the pages of her friends (e.g., `http://sn.com/user.php?id=bob`, `http://sn.com/user.php?id=charles`, etc.) and that user's non-topological attributes (e.g., the set of movies she likes) – an algorithm, though, cannot download the webpage of a vertex without knowing its URL: thus, to download a generic vertex `zoe` from the graph, the algorithm first needs to download all the vertices in (at least) one path from the seed vertex (e.g., `alice`) to `zoe`.

Clearly, given enough many resources, the algorithm could crawl the whole social network (that is, download each of the social network's vertices), and then reduce the problem of computing the online graph property to the centralized one – unfortunately, it is practically infeasible to download millions, or billions, of vertices from a social network (the APIs that can be used to access the network usually enforce strict limits on how many vertices can be downloaded per day). Several techniques have been proposed in the literature for studying properties of online graphs – almost all of them assume to have access to a random oracle that returns a random vertex of the graph according to a certain distribution (usually, either uniform, or proportional to the degree), e.g., [5, 12, 18, 10, 16].

When running algorithms on online social networks, it is often hard or impossible to implement a uniform-at-random random oracle, and to get samples out of it – the complexity of this oracle is one of the main problems that we tackle in this paper.

In practice, an algorithm is given (the URL of) a seed vertex (or, some seed vertices) of the social network; the algorithm has to download that seed vertex, get the URLs of its neighbors, and then decide which of them to download next; after having downloaded the second node, the algorithm (might) learn of the existence of some other URLs/vertices, and can then decide which of the known (but unexplored) URLs to download – and so on, and so forth, until the algorithm can return a good approximation of the property to be estimated.

The natural cost function in this setting is the (random) number of vertices that the algorithm has to download, or *query*, before making its guess – the cost function is usually bounded in terms of properties of the graph (e.g., its order, its average degree, etc.), and in terms of the quality of the algorithm's guess.

Many problems of this form can be found in the literature. In this paper, we consider two natural problems, that are at the heart of many others, and whose complexity (as far as we know) was open before this work:

- the "average score problem": assuming that each vertex holds some score in $[0, 1]$, compute an approximation of the average of the scores;
- the "uniform at random sample": return a random vertex from the graph whose distribution is approximately uniform.

In a sense, the latter problem is technically more interesting than the former (a solution to the latter provides a solution to the former). In practice, though, the average score problem is much more significant (and ubiquitous), given its many applications [11, 1, 14, 21] (e.g., computing the favorability rating of a candidate, or the average star-rating of a movie). These problems are similar in flavor to some graph property testing problems [15]. Observe that our setting is similar, but different, than various graph property testing settings – many of the existing graph property testing algorithms require, as a primitive, the ability to sample a uniform at random vertex. Our work can be seen as a way of implementing that primitive using the oracles mentioned above. (See also, [4]).

A number of algorithms have been proposed for the uniform-at-random sample problem [7, 18] – the best known algorithms require roughly $\tilde{O}\left(\mathrm{t_{mix}} \cdot \mathrm{d_{avg}}\right)$ vertex queries/downloads to return a vertex whose distribution is (close to) uniform at random , where $\mathrm{t_{mix}}$ is the mixing time of the lazy random walk on the graph, and $\mathrm{d_{avg}}$ is its average degree [7][3]. These algorithms do not use any knowledge of the average degree of the graph $\mathrm{d_{avg}}$, but need to know a constant approximation of its mixing time $\mathrm{t_{mix}}$. To our knowledge, the best lower bound for the uniform-at-random sample problem before this work, was $\Omega(\mathrm{t_{mix}} + \mathrm{d_{avg}})$ [7] – one of the main results of this paper is (i) an almost tight lower bound of $\Omega(\mathrm{t_{mix}} \cdot \mathrm{d_{avg}})$ for this problem, for wide (in fact, polynomial) ranges of the two parameters.[4] The lower bound holds even for algorithms that know constant approximations of $\mathrm{d_{avg}}$.

Our lower bound construction for the uniform-at-random sample problem also provides (ii) a tight lower bound of $\Omega\left(\mathrm{d_{avg}}\,\mathrm{t_{mix}}\right)$ for the average score problem – in fact, we resolve the complexity of the average score problem by showing that our lower bound coincides with the complexity of some previously proposed algorithms, whose analysis we improve.

The same lower bound construction further resolves (iii) the complexity of the average-degree estimation problem, and (iv) entails a non-tight, but significant, lower bound for the problem of guessing the graph order (that is, the number of vertices in the graph).

It is interesting to note that all the algorithms that were proposed require $O(\log n)$ space, while our lower bounds hold for general algorithms with no space restriction. Thus, the problems we consider can be solved optimally using only tiny amounts of space.

---

[3] Real-world social networks are known to have a small average degree $\mathrm{d_{avg}}$; their mixing time $\mathrm{t_{mix}}$ has been observed [20] to be quite small, as well.

[4] Observe that the two parameters have to obey some bound for such a lower bound to hold – in general, any problem can be solved by downloading all the n vertices of the graph: thus, if $\mathrm{t_{mix}} \cdot \mathrm{d_{avg}} > \omega(n)$, one can solve the uniform-at-random sample problem with less than $o(\mathrm{t_{mix}} \cdot \mathrm{d_{avg}})$ vertex queries.

2 Preliminaries

Consider a connected and undirected graph $\mathcal{G} = \langle V_{\mathcal{G}}, E_{\mathcal{G}} \rangle$ with no self-loops (e.g., the Facebook friendship graph), and a function on its vertices $\mathcal{F} : V_{\mathcal{G}} \to [0,1]$.[5] We aim to estimate the average value of this function, i.e., $f_{avg} = \sum_{v \in V_{\mathcal{G}}} \mathcal{F}(v)/n$ where $n = |V_{\mathcal{G}}|$.

Motivated by applications, we assume that accessing the graph is a costly operation, and that there is little or no information about its global parameters such as the average degree, the number of vertices or the maximum degree. However, we can access a "friendship" oracle: that is, an oracle which, given a vertex $v \in V_{\mathcal{G}}$, outputs references (their ids, or their URLs) to its neighbors $N_v = \{u \in V_{\mathcal{G}} | (v,u) \in E_{\mathcal{G}}\}$. In such a setting, it is natural to approximate f_{avg} by taking samples from a Markov chain based on the graph structure (see, e.g., [9, 8]). A simple random walk on the graph, though, will not serve our purposes since it samples vertices with probability proportional to their degree, while our goal is to take a uniform average of the values of $\mathcal{F}$.

On a graph $\mathcal{G} = \langle V_{\mathcal{G}}, E_{\mathcal{G}} \rangle$, a lazy simple random walk is a Markov chain which being at vertex $v \in V_{\mathcal{G}}$, stays on v with probability $1/2$ and moves to $u \in N_v$ with probability $1/(2\deg(v))$. Given that $\mathcal{G}$ is connected, the lazy random walk will converge to its unique stationary distribution which we denote by Π^1 and which is equal to $\Pi^1(v) = \deg(v)/2|E_{\mathcal{G}}|$, $\forall v \in V_{\mathcal{G}}$.

By $\mathrm{t}_{\mathrm{mix}}(\mathcal{G})$ we refer to the mixing time of the lazy random walk on $\mathcal{G}$, which is the minimum integer satisfying: for any $\tau \geq \mathrm{t}_{\mathrm{mix}}(\mathcal{G})$, $\left|X^\tau - \Pi^1\right|_1 \leq 1/4$, where X^τ is the distribution of the lazy walk at time τ, and $|\cdot|_\ell$ is the ℓ-norm of a vector. Note that by the theory of Markov chains, by taking $\tau \geq \mathrm{t}_{\mathrm{mix}}(\mathcal{G}) \log(1/\epsilon)$ we have $\left|X^\tau - \Pi^1\right|_1 \leq \epsilon$. We denote the uniform distribution on vertices of $\mathcal{G}$ by Π^0, i.e., $\Pi^0(v) = 1/|V_{\mathcal{G}}|, \forall v \in \mathcal{G}$. In general, we denote a distribution on $V_{\mathcal{G}}$ weighing each vertex $v \in V_{\mathcal{G}}$ proportional to $\deg(v)^\zeta$ by Π^ζ. We may drop all the subscripts when doing so does not cause ambiguity.

Following the framework of [7], we consider two measures of time complexity. First the number of downloaded vertices, and second the number of steps the algorithm takes to produce the output. Note that accessing an already downloaded vertex has a negligible cost, and hence, the most relevant cost of the algorithm is the number of downloaded vertices. As mentioned in the introduction, the algorithms considered in [7] and in this paper, only require space to store constantly many vertices, while our lower bound results hold regardless of the space complexity of the algorithms.

Our Contribution. We begin by discussing the problem of producing an approximately-uniform sample vertex from an unknown graph (Problem 1); showing that some algorithm presented in the literature are optimal (Theorem 1). Then, we proceed to the problem of estimating f_{avg} for a function $\mathcal{F} : V_{\mathcal{G}} \to [0,1]$ (Problem 2). We extend the positive results of [7]; we particularly study one algorithm, the "Maximum Degree algorithm", which we show to be suboptimal in the number of downloaded vertices. This algorithm requires knowledge of some constant approximation of the graph's mixing time, and and upper bound on its maximum degree. We also show new lower bounds for constant approximations of the order and the average degree of a graph. A summary of our contribution is presented in Table 1. We remark that, in practice, only an upper bound on the mixing time is available. In the equations of Table 1, $\mathrm{t}_{\mathrm{mix}}$ can be substituted with any upper bound on the mixing time of the graph.

[5] Note that from any bounded function we can get a function with range $[0,1]$, through a simple affine transform. Therefore, our results can be trivially extended to functions with any bounded range.

Table 1 Upper bounds and lower bounds on number of queried vertices for algorithms which explore the graph using a neighborhood oracle and a seed vertex. As mentioned before, $\mathrm{t_{mix}}$ is the mixing time of the lazy random walk on the graph, $\mathrm{d_{avg}}$ is its average degree, $\mathcal{D}$ is an upper bound on its maximum degree, Π^1 is its stationary distribution, and ϵ and δ are the precision parameters. The lower bounds for estimating the number of vertices and the average degree hold for any constant approximation.

	Upper Bound	Lower Bound
Average of a Bounded Function	$O(\mathrm{t_{mix}}\,\mathrm{d_{avg}}\log(\delta^{-1})\epsilon^{-2})$ (Theorem 2, with an Algorithm of [7])	$\Omega(\mathrm{t_{mix}}\,\mathrm{d_{avg}}\log(\delta^{-1})\epsilon^{-2})$ (Theorem 3)
Uniform Sample	$O(\mathrm{t_{mix}}\,\mathrm{d_{avg}}\log(\epsilon^{-1}))$ ([7])	$\Omega(\mathrm{t_{mix}}\,\mathrm{d_{avg}})$ (Theorem 1)
Number of Vertices	$O(\mathrm{t_{mix}}\max\{\mathrm{d_{avg}}, \lvert\Pi^1\rvert_2^{-1/2}\}\log(\delta^{-1})\log(\epsilon^{-1})\epsilon^{-2})$ ([18])	$\Omega(\mathrm{t_{mix}}\,\mathrm{d_{avg}})$ (Theorem 4)
Average Degree	$O(\mathcal{D}^2\mathrm{t_{mix}}\,\mathrm{d_{avg}}\log(\delta^{-1})\epsilon^{-2})$ (Application of Theorem 2)	$\Omega(\mathrm{t_{mix}}\,\mathrm{d_{avg}})$ (Theorem 4)

In Section 3, we prove our lower bound results on the number of oracle calls for the following problems: sampling a vertex, learning the order, and the average degree of the graph. Estimations of these parameters in a graph are intertwined meaning that a knowledge about one of them the complexity of estimating the other one changes. For instance, Goldreich and Ron [16] show that, if a uniform sample generator is accessible at zero cost (alternatively, if the order of graph is precisely known), then the average degree is computable in $\sqrt{|V_{\mathcal{G}}|/\,\mathrm{d_{avg}}}$ steps. Our lower bounds for the aforementioned problems hold if the algorithm has no $\epsilon-$approximation of the order, and of the average degree of the graph. On the other hand, the lower bound we obtain for an $\epsilon-$approximation of a bounded function's average holds even if the graph's structure is precisely known.

Number of downloads to produce a close-to-uniform sample. We prove a lower bound of $\Omega(\mathrm{t_{mix}}\,\mathrm{d_{avg}})$, thus, showing that the rejection algorithm and the maximum degree algorithm suggested in the literature [7] can be optimal (Theorem 1). We observe that the two algorithms require some knowledge of $\mathrm{t_{mix}}$. In most scenarios, only an upper bound on $\mathrm{t_{mix}}$ is available. Thus, it is more accurate to claim that the two algorithms are optimal when some constant approximation of the mixing time is available.

Number of downloads to estimate the number of vertices. The problem of estimating the order of a graph is widely studied [9, 18]. Katzir et al. [18] (2011) propose an algorithm that, having access to an oracle that produces random vertices from the graph's stationary distribution, requires $\max\{\frac{1}{|\Pi^1|_2}, \mathrm{d_{avg}}\}(\frac{1}{\epsilon^2\delta})$ samples to obtain an ϵ approximation with probability at least $1-\delta$. It has been shown the number of samples in Katzir's algorithm is necessary ([17]). The Katzir et al. algorithm implies an upper bound of $\mathrm{t_{mix}}\max\{\frac{1}{|\Pi^1|_2}, \mathrm{d_{avg}}\}(\frac{\log(\epsilon^{-1})}{\epsilon^2\delta})$ vertex queries to obtain an ϵ approximation with probability at least $1-\delta$ in our friendship-oracle model. In Theorem 4 we present a lower bound on the number of accesses to the vertices, to get a constant approximation of the graph's order in our friendship-oracle model. Our lower bound is tight for the graphs that satisfy $\frac{1}{|\Pi^1|_2} < \mathrm{d_{avg}}$ – that is, the graphs whose variance

of the degree distribution is greater than n.[6] This class include, say, all the graphs having a power-law degree distribution with exponent smaller than 3/2 (e.g., social networks [19]).

We remark that this lower bound problem is still open for graphs with $\frac{1}{|\Pi^1|_2} > \mathrm{d_{avg}}$, for example regular sparse graphs. Recently, Ben-Hamou et al. studied the problem of estimating the size of a regular graph when having access to an oracle analogous to ours, and presented an constant approximation algorithm querying $(t_{\mathrm{unif}})^{3/4}\sqrt{n}$ vertices, where t_{unif} is the minimum integer satisfying: $\tau \geq t_{\mathrm{unif}}(\mathcal{G})$, $\max_{v\in V} |X^\tau(v)/\Pi^1(v) - 1| \leq 1/4$ [3].

Number of downloads to estimate the average degree. There are quite a few results on estimating the average degree of a graph. The first one by Feige et al. [13] introduced a sublinear algorithm of complexity $\sqrt{|V_\mathcal{G}|}$ for a 2-approximation. Goldreich et al. [16] extends Feige et al.'s result and presents an $(1 \pm \epsilon)$ approximation algorithm with running time $O(1/\epsilon)\sqrt{|V_G/\,\mathrm{d_{avg}}|}$ – they also prove a lower bound on the number of samples of $\sqrt{|V_G|/\,\mathrm{d_{avg}}}$ – both of [13] and [16] assume to have access to an oracle capable of producing a uniform at random vertex. Recently, Dasgupta et al. [10] showed that by sampling $O(\log(\mathcal{D})\log\log(\mathcal{D}))$ vertices of a graph from some weighted distribution[7] one can obtain a $(1\pm\epsilon)$ approximation of its average degree, where $\mathcal{D}$ is an upper bound on the maximum degree. By factoring in the the cost of sampling, the complexity becomes $O(\mathrm{t_{mix}}\log(\mathcal{D})\log\log(\mathcal{D}))$. Taking $\mathcal{D} = n$ and adding the cost of estimating the graph size, takes the upper bound to: $O\left(\mathrm{t_{mix}}\left((\log(n)\log\log(n)) + \mathrm{d_{avg}} + \frac{1}{|\Pi^1|_2}\right)\right)$.

In Theorem 4 of this paper we show that by downloading $o(\mathrm{t_{mix}}\,\mathrm{d_{avg}})$ vertices, it is impossible for an algorithm to have any constant approximation of the average degree $\mathrm{d_{avg}}$ with probability more than some constant.

Finally, our main result is the following lower bound – unlike the above three lower bounds, this one holds even if we know exactly the graph's structure.

Number of downloads to find an ϵ, δ approximation for the average of a bounded function. In Theorem 3, we show that an algorithm requires $\Omega\left(\mathrm{t_{mix}}\,\mathrm{d_{avg}}(1/\epsilon^2)\log(1/\delta)\right)$ vertex downloads to produce an $\epsilon-$additive approximation of f_{avg}, with probability at least $1-\delta$. This lower bound, together with Theorem 2, allows us to conclude that the "maximum degree algorithm" is an optimal algorithm for this problem. Note that this algorithm has to have some upper bound $\mathcal{D}$ on the maximum degree of the graph. In many situations, one can assume that this information is available – for instance $\mathcal{D} \leq n$ and, in many cases, one can assume to have a constant approximation to the order of the graph (for instance, in Facebook, one could claim that $\mathcal{D}$ is no larger than the world's population.) Observe that the maximum degree algorithm suffers no loss in getting a large $\mathcal{D}$, as opposed to a tighter one, since $\mathcal{D}$ does not impact the upper bound on the number of downloaded vertices.

2.1 Statement of Problems and Results

▶ **Problem 1.** *Input: A seed vertex $s \in V$ in graph $\mathcal{G} = \langle V_\mathcal{G}, E_\mathcal{G}\rangle$. Output: A random vertex $v \in V_\mathcal{G}$ whose distribution is at total variation distance at most ϵ from the uniform one on $V_\mathcal{G}$.*

---

[6] Let pr_k be the fraction of vertices with degree k. We have $\left|\Pi^1\right|_2 = n\sum_{i=1}^n \mathrm{pr}_k \frac{k^2}{4|E|^2} = \frac{1}{n\,\mathrm{d_{avg}}^2}\sum_{i=1}^n \mathrm{pr}_k k^2$. Thus, to have $1/\sqrt{|\Pi^1|_2} \leq \mathrm{d_{avg}}$, it is necessary and sufficient to have $\sum_{i=1}^n k^2\mathrm{pr}_k > n$.

[7] Dasgupta et al. use an oracle samples each $v \in V_\mathcal{G}$ proportional to $\deg(v) + c$ for some constant c. Note that for $c = 0$ this distribution will be the same as the stationarity.

Several algorithms have been proposed for Problem 1 [7, 18] – we will specifically consider the "maximum degree sampling" algorithm, the "rejection sampling" algorithm, and the "Metropolis Hasting" algorithm.

The efficiency of the three algorithms has been studied in terms of the number of their running time (or the number of steps they make on the Markov chain they are based on) and, more importantly, on the number of *queries*[8] (or downloaded vertices) that the algorithm performs. The rejection sampling and maximum degree algorithms produce a close-to-uniform random vertex by querying[9] $\tilde{O}\left(\mathrm{t_{mix}}\,\mathrm{d_{avg}}\right)$ distinct vertices from the graph, where $\mathrm{t_{mix}}$ is the mixing time of a simple random walk on $\mathcal{G}$, and $\mathrm{d_{avg}}$ is the average degree of $\mathcal{G}$. In terms of space complexity, each of these algorithms is based on a simple random walk on $\mathcal{G}$ and thus only require space to save constant number of vertices.

One of the main results of this paper is Theorem 1, which shows the optimality of the maximum degree, and of the rejection sampling, algorithms for Problem 1 – their running time. We observe that our lower bound holds regardless of the amount of space available to the algorithm.

▶ **Theorem 1.** *For any large enough constant c, and for any arbitrary n, $d = \omega(\log n)$, and $t = o(\frac{n}{d^2})$ there exists a distribution over graphs $\mathcal{G} = \langle V, E\rangle$, each having mixing time $\Theta(t)$, $\mathbb{E}(|V|) = \Theta(n)$, $\mathrm{d_{avg}} = \mathbb{E}_{v\in V}(\deg_v) = \Theta(d)$, such that any algorithm $\mathcal{A}$ that queries less than $\mathrm{d_{avg}}\,\mathrm{t_{mix}}/c$ vertices of $\mathcal{G}$, and that returns some random vertex of $\mathcal{G}$ with distribution $\Pi_{\mathcal{A}}$, is such that*[10] $\mathbb{E}\left[\left|\Pi_{\mathcal{A}} - \Pi^0\right|_1\right] \geq \frac{24}{100} - \frac{202}{c-1}$.

The same lower bounds also hold if one aims to obtain the generic Π^ζ distribution [11] *: if $\zeta > 1$, and d and t satisfy $d = o(t^{\frac{\zeta-1}{\zeta}})$, then any algorithm $\mathcal{A}$ that queries less than $\mathrm{d_{avg}}\,\mathrm{t_{mix}}/c$ vertices of $\mathcal{G}$, and that returns some random vertex of $\mathcal{G}$ with distribution $\Pi_{\mathcal{A}}$, is such that: $\mathbb{E}\left[\left|\Pi_{\mathcal{A}} - \Pi^\zeta\right|_1\right] \geq \frac{24}{100} - \frac{202}{c-1}$.*

The above theorem, and the other lower bound results that we mention in this section, will be proved in Section 3.

Then, we consider the problem of finding the average of a function $\mathcal{F}$ defined on vertices of a graph and ranging in $[0, 1]$.

▶ **Problem 2.** *Input: A seed vertex $s \in V$ in graph $\mathcal{G} = \langle V, E\rangle$ – each vertex v holds a value $0 \leq \mathcal{F}(v) \leq 1$ which we learn upon visiting it. Output: $\bar{f}$ such that $\mathbb{P}\left(|\bar{f} - f_{avg}| \leq \epsilon\right) \geq 1 - \delta$.*

Note that having a uniform sampler (the maximum degree or rejection sampling algorithm of [7]), we can have an ϵ approximation of f_{avg} with probability $1-\delta$ by taking $O(\epsilon^{-2}\log(\delta^{-1}))$ independent samples which are ϵ close to uniformity. In total, the number of queries will be $O(\mathrm{t_{mix}}\,\mathrm{d_{avg}}\log(\delta^{-1})\epsilon^{-2}\log(\epsilon^{-1}))$. Here we propose a slight variation of the "maximum degree" algorithm to obtain a tight upper bound. We improve the analysis of the "maximum degree algorithm" in Theorem 2 – its performance beats the other natural three algorithms, and the main result of this paper is that this performance is optimal (Theorem 3).

The proof of the following Theorem is omitted from this extended abstract.

---

[8] A vertex is "queried", when the set of its neighbors is obtained from the oracle for the first time – equivalently, when it is downloaded.

[9] To get ϵ close to the uniform distribution we need $O\left(\mathrm{t_{mix}}\,\mathrm{d_{avg}}\log(\epsilon^{-1})\right)$ downloads.

[10] Observe that the expected ℓ_1 distance between the distributions is over the random variable $\Pi_{\mathcal{A}}$.

[11] The Π^ζ distribution is the distribution in which the probability of a vertex $v \in V$ is proportional to $deg(v)^\zeta$.

Algorithm 1 The Maximum Degree Algorithm.

Input: Seed vertex $s \in V_{\mathcal{G}}$, $\mathrm{t}'_{\mathrm{mix}}$ a constant approximation of $\mathrm{t}_{\mathrm{mix}}$, and an upper bound $\mathcal{D}$ on d_{max}

Output: An ϵ additive approximation of f_{avg} with probability at least $1-\delta$

Consider the maximum degree Markov chain: at vertex $v \in V$ go to the generic $u \in N_v$ with probability $1/\mathcal{D}$, otherwise stay at v.
$T \leftarrow \mathrm{t}'_{\mathrm{mix}}\mathcal{D}/\mathrm{d}_{\min}$
Starting from s, run the chain for $T \cdot \left(1+\epsilon^{-2}\ln\delta^{-1}\right)$ steps – let $v_0 = s, v_1, v_2, \ldots$ be the states that are visited by the walk
$S \leftarrow 0, t \leftarrow 0, i \leftarrow 0$
while $t < \frac{\mathrm{d}_{\mathrm{avg}}}{d_{min}}(\mathrm{t}'_{\mathrm{mix}}/\epsilon^2)\log(1/\delta)$ **do**
  $i \leftarrow i+1$
  if $v_i \neq v_{i-1}$ **then**
    $t \leftarrow t+1$
  $S = S + \mathcal{F}(v_i)$
return S/i

▶ **Theorem 2.** *Consider a graph $\mathcal{G} = \langle V, E\rangle$, and a function $\mathcal{F} : V \to [0,1]$. Let $\mathrm{t}_{\mathrm{mix}}$ be the mixing time of the simple lazy random walk on $\mathcal{G}$. Let $\bar{f}$ be the value returned by Algorithm 1. We assume the algorithm knows a constant approximation to $\mathrm{t}_{\mathrm{mix}}$. i.e. $\mathrm{t}'_{\mathrm{mix}} = \Theta(\mathrm{t}_{\mathrm{mix}})$. Then,*

$$\mathbb{P}\left(|\bar{f} - f_{avg}| \geq \epsilon\right) \leq \delta \tag{1}$$

This algorithm queries $\Theta(\mathrm{t}_{\mathrm{mix}}\,\mathrm{d}_{\mathrm{avg}}\,\epsilon^{-2}\log(\delta^{-1}))$ vertices from the graph, and requires space for saving a constant number of them. The number of computational steps it performs is $\Theta(\mathcal{D}\mathrm{t}_{\mathrm{mix}}\epsilon^{-2}\log(\delta^{-1}))$.

The main result of this paper is the following lower bound which complements the upper bound given in the previous theorem:

▶ **Theorem 3.** *For any arbitrary n, $d = \omega(\log n)$, and $t = o(\frac{n}{d^2})$ there exists a distribution over graphs $\mathcal{G} = \langle V, E\rangle$ with mixing time $\Theta(t)$, $\mathbb{E}(|V|) = 4n$, $\mathrm{d}_{\mathrm{avg}} = \mathbb{E}_{v\in V}(\deg_v) = \Theta(d)$, and a function $\mathcal{F} : V \to \{0,1\}$ such that, any algorithm $\mathcal{A}$ as described above which aims to return the average of $\mathcal{F}$, with ϵ precision for arbitrary $0 < \epsilon, \delta < 1$, and queries less than $\Omega(\mathrm{t}_{\mathrm{mix}}\,\mathrm{d}_{\mathrm{avg}}\,\epsilon^{-2}\log(\delta^{-1}))$ vertices of $\mathcal{G}$ fails with probability greater than δ.*

Finally, we consider the problems of obtaining an approximation of the average degree, and the number of vertices, of a graph:

▶ **Problem 3.** *Input: A seed vertex $s \in V$ in graph $\mathcal{G} = \langle V, E\rangle$. Output: an integer $\bar{n}$ such that $\mathbb{P}\left(|\bar{n} - |V|| \leq \epsilon\right) \geq 1-\delta$.*

By a result of Katzir [18], we know by taking $\max\{\mathrm{d}_{\mathrm{avg}}, 1/\sqrt{|\Pi^2|_2}\}\epsilon^{-2}\delta^{-1}$ samples from the stationary distribution we are capable to obtain an ϵ approximation with probability at least $1-\delta$. To implement a sampling oracle using our neighborhood oracle, we can run a Markov chain for $\mathrm{t}_{\mathrm{mix}}\log(\epsilon^{-1})$ steps. Thus, the runtime will be $\mathrm{t}_{\mathrm{mix}}\max\{\mathrm{d}_{\mathrm{avg}}, 1/\sqrt{|\Pi^1|_2}\}\log(\epsilon^{-1})\epsilon^{-2}\delta^{-1}$, which for constant ϵ and δ is $\mathrm{t}_{\mathrm{mix}}\max\{\mathrm{d}_{\mathrm{avg}}, 1/\sqrt{|\Pi^1|_2}\}$. Theorem 4 provides a lower bound for a constant approximation which is as mentioned before tight when the variance of the degree distribution is greater than n.

▶ **Problem 4.** *Input: A seed vertex $s \in V$ in graph $\mathcal{G} = \langle V, E \rangle$. Output: an integer $\bar{d}$ such that* $\mathbb{P}\left(|\bar{d} - \mathrm{d_{avg}}| \leq \epsilon\right) \geq 1 - \delta$.

Normalize the function $deg : V_{\mathcal{G}} \to \mathbb{R}$ by dividing its value to $\mathcal{D}$. By Theorem 2, Algorithm 1 provides an ϵ approximation with probability at least $1 - \delta$ after downloading $O(\mathcal{D}^2\, \mathrm{t_{mix}}\, \mathrm{d_{avg}}\, \epsilon^{-2} \log(\delta^{-1}))$ many vertices – that is, $O(\mathcal{D}^2\, \mathrm{t_{mix}}\, \mathrm{d_{avg}})$ many vertices for constant ϵ and δ. With Theorem 4, we provide a lower bound for a constant approximation.

▶ **Theorem 4.** *For any arbitrary n, $d = \omega(\log n)$, and $t = o(\frac{n}{d^2})$ there exists a distribution of graphs $\mathcal{G} = \langle V, E \rangle$ with mixing time $\Theta(t)$, $\mathbb{E}(|V|) = \Theta(n)$, $\mathrm{d_{avg}} = \mathbb{E}_{v \in V}(\deg_v) = \Theta(d)$ such that, for arbitrary constants $c' > 1$ and large enough c, any algorithm that queries at most $\mathrm{d_{avg}}\, \mathrm{t_{mix}}\, /c$ vertices of the graph, and that outputs an estimation $\bar{n}$ of $n = |V|$ (resp., an estimation $\bar{d}$ of $\mathrm{d_{avg}} = 2|E|/n$), has to satisfy $\max\{\bar{n}/n, n/\bar{n}\} > c'$ (resp., $\max\{\bar{d}/\,\mathrm{d_{avg}}, \mathrm{d_{avg}}\,/\bar{d}\} > c'$), with probability at least $\frac{99}{100} - \frac{202}{c-1}$.*

3 Proof Strategy of the Main Theorems

The proof of our Lower Bounds will be based on the following high-level strategy. Nature will first randomly sample a graph $\mathcal{H}$ according to some distribution; with probability $1/2$, $\mathcal{H}$ will be the unknown graph traversed by the algorithm; with the remaining probability, the algorithm will traverse a graph $\mathcal{G}$ which is obtained from $\mathcal{H}$ by means of a transformation that we call the *decoration construction*. We will prove that, for the right choice of the distribution over $\mathcal{H}$, an algorithm that performs too few queries to the unknown graph will be unable to tell with probability more than $1/2 + o(1)$, whether the unknown graph it is traversing is distributed like $\mathcal{H}$, or like $\mathcal{G}$.

The decoration construction will guarantee that the properties (e.g., number of nodes, average degree, or even the values assigned by the bounded function to the vertices) will be quite far from each other in $\mathcal{H}$ and $\mathcal{G}$. This will make it impossible for the algorithm to get good approximation of any of those properties – we will also show it impossible for the algorithm to return a close to uniform-at-random vertex (essentially because the decoration construction will add a linear number of nodes to $\mathcal{H}$, and the algorithm will be unable to visit any of them with the given budget of queries.)

We present a roadmap of our proof strategy here, and omit the detailed proofs in this extended abstract. We start by describing the *decoration construction* which, given any graph $\mathcal{H}$, produces a graph $\mathcal{G}$ with similar mixing time and average degree, but with a linear number of "hidden" new vertices. After presenting the definition for the decoration construction, in Definition 6 we introduce a class of graphs to which we apply this construction. These graphs' mixing time and average degree can be set arbitrarily. Later, in Lemma 7 we prove that if an algorithm, equipped only with the neighborhood oracle traverses a graph of this type and queries few vertices of it, it will not be capable of finding any of its hidden vertices. This is our main lemma from which Theorems 1, 3, and 4 can be concluded. We now proceed to the formal definitions:

▶ **Definition 5** (The Decoration Construction)**.** Let $\mathcal{H} = \langle V, E \rangle$ be an arbitrary graph. We construct $\mathcal{G}$ from $\mathcal{H}$ in the following way:

Take $t := \mathrm{t_{mix}}(\mathcal{H})$, and mark any vertex $v \in V$ with probability $1/t$. For any marked vertex $v \in V$, add a vertex v^* and connect it to v via an edge. For a constant c_1, attach $c_1 t - 1$ new degree one vertices to v^* – this makes the degree of v^* equal to $c_1 t$. Let this new graph be $\mathcal{G}$. We denote the set of marked vertices by MARKED and the set $\mathcal{G} \setminus \mathcal{H}$ by STARRED.

By saying a vertex v is STARRED (MARKED) we mean $v \in$ STARRED ($v \in$ MARKED), and to indicate their numbers we use a preceding #. We call the STARRED vertices with degree $c_1 t$ the STARRED centers. Note that to any STARRED vertex we can associate a unique MARKED vertex.

We now introduce the random graph to which we will apply the decoration construction, and which will be at the heart of our lower bounds.

▶ **Definition 6.** We define the graph $\mathcal{H}_{n,d,\psi}$ as follows: given arbitrary parameters n, d, and $0 < \psi < 1$, take two Erdös-Rényi graphs $H_1 = \langle V_{H_1}, E\rangle$ and $H_2 = \langle V_{H_2}, E\rangle$ with parameters $\langle n, d/n\rangle$. Choose ψn vertices uniformly at random from V_{H_1} namely $v_1, v_2, \dots v_{\psi n}$, and then ψn vertices uniformly at random from V_{H_2} namely $u_1, u_2, \dots n_{\psi n}$. Select a uniformly random permutation σ of ψn numbers and put an edge between the vertices v_i and $u_{\sigma(i)}$ for each $1 \le i \le \psi n$.

The decoration construction does not change the mixing time of $\mathcal{H}$ drastically, in fact, $\mathrm{t_{mix}}(\mathcal{H}) \le \mathrm{t_{mix}}(\mathcal{G}) \le c\,\mathrm{t_{mix}}(\mathcal{H})$ for some constant c. The $\mathcal{H}_{n,d,\psi}$s are useful in our proofs, for the reason that, by changing the parameters n, d, and ψ in certain ranges these graphs acquire arbitrary $\mathrm{t_{mix}}$ and $\mathrm{d_{avg}}$, yet their behaviour remains similar to Erdös-Rényi random graphs.

We now present the following lemma (proof ommited), which states if few queries are performed by an algorithm, then the probability of finding the STARRED vertices is tiny. Then, we conclude our main theorems; Theorem 1, 3, and 4:

▶ **Lemma 7.** *Consider arbitrary n, $d > \omega(\log n)$, $\Omega(\log n) < t < o(n/d^2)$, so that $t/d = \Omega(1)$, take $\mathcal{G}$ to be the graph obtained from the decoration construction applied to $\mathcal{H}_{n,d,d/t}$[12]. We have, $\mathrm{t_{mix}}(\mathcal{G}) = \Theta(t)$ and $\mathrm{d_{avg}} = \Theta(d)$.*

If, instead, $t = O(\log n/\log d)$, and $d = \Theta(\log^d n)$, take $\mathcal{G}$ to be the decorated version of an Erdös-Rényi graph with parameters $\langle n, d/n\rangle$.

Then,

- *if an algorithm traverses the edges of $\mathcal{G}$ and queries at most td/c vertices of $\mathcal{G}$; c being a constant, then with probability at least $99/100 - 202/(c-1)$ there is no* STARRED *vertex among its queried vertices.*
- *If an algorithm traverses the edges of $\mathcal{G}$ and queries $q \le \frac{n}{cd}$ vertices of $\mathcal{G}$; c being a constant, then with probability $1 - o(1)$ the expected number of* STARRED *centers which have been queried is less than $\frac{8c}{c-1}\left(\frac{q}{dt}\right)$.*

We can finally prove our three main Theorems:

*** Proof of Theorem 1.** Consider the two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, $\mathcal{G}_1$ being the graph of Lemma 7 with $c_1 = 1$ and $\mathcal{G}_2$ the same graph without the STARRED vertices (the graph before the decoration construction). Any algorithm which queries less than $\mathrm{t_{mix}}\,\mathrm{d_{avg}}/c$ vertices of $\mathcal{G}_1$ or $\mathcal{G}_2$ will fail to distinguish between them with probability at least $\frac{1}{100} + \frac{202}{c-1}$. Let $\Pi^0_{\mathcal{G}_i}$ be the uniform distribution on vertices of $\mathcal{G}_i$; $i = 1, 2$. In $\mathcal{G}_1$ with high probability we have at least $2|V_{\mathcal{G}_1}|$ STARRED vertices.

- *Part 1.* Note that $|\Pi^0_{\mathcal{G}_1}, \Pi^0_{\mathcal{G}_2}|_1 \ge 1/4$. Thus, if the natures selects $\mathcal{G}_1$, any algorithm $\mathcal{A}$ which aims to outputs $\Pi^0_{\mathcal{G}_1}$ will return a distribution $\Pi_{\mathcal{A}}$ satisfying $|\Pi^0_{\mathcal{G}_1}, \Pi_{\mathcal{A}}|_1 \ge 1/4 - \frac{1}{100} - \frac{202}{c-1}$.

[12] If $c > d/t \ge 1$, for constant c, let $\psi = d/tc$.

- *Part 2.* For $\zeta > 1$, let $\Pi^{\zeta}_{\mathcal{G}_i}$ be the probability distribution on vertices of $\mathcal{G}_i$; $i = 1, 2$ weighing each vertex v proportional to $deg(v)^{\zeta}$. If we take a sample from distribution $\Pi^{\zeta}_{\mathcal{G}_1}$ it will be STARRED vertex with probability $(t^{\zeta-1}+1)/2(t^{\zeta-1}+d^{\zeta}+1)$. Thus, for $t^{\zeta-1} \geq d^{\zeta}$ we have, $|\Pi^{\zeta}_{\mathcal{G}_1}, \Pi^{\zeta}_{\mathcal{A}}|_1 \geq 1/4$. Thus, $|\Pi^{0}_{\mathcal{G}_1}, \Pi_{\mathcal{A}}|_1 \geq 24/100 - 202/c$. □

* **Proof of Theorem 4.** Take the two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, $\mathcal{G}_1$ being the graph of Lemma 7, and $\mathcal{G}_2$ the same graph without the STARRED vertices (the graph before the decoration construction). We have: $\mathbb{E}(|V_{\mathcal{G}_2}|) = n$, $\mathbb{E}(|V_{\mathcal{G}_1}|) = (1+c_1)n$, and $\mathbb{E}(\mathrm{d_{avg}}(\mathcal{G}_2)|) = d$, $\mathbb{E}(|\,\mathrm{d_{avg}}(\mathcal{G}_1)|) = \frac{d+c_1}{1+c_1}$. □

* **Proof of Theorem 3.** Take $c_1 = 1$, and let $\mathcal{G} = \langle V_{\mathcal{G}}, E_{\mathcal{G}} \rangle$ be the graph constructed as in Lemma 7. Consider two functions $\mathcal{F}_1 : V_{\mathcal{G}} \to [0,1]$ and $\mathcal{F}_2 : V_{\mathcal{G}} \to [0,1]$. Let the function $\forall v \in V_{\mathcal{G}} \setminus \text{STARRED}, \mathcal{F}_1(v) = \mathcal{F}_2(v) = 0$. For any $v \in$ STARRED we set $\mathcal{F}_1(v) = 1$ with probability $1/2 + \epsilon$ and $\mathcal{F}_2(v) = 1$ with probability $1/2 - \epsilon$.

Note that $|\mathcal{F}_1 - \mathcal{F}_2|_1 \geq \epsilon$, and by employing the following classical result [6], with probability $1 - o(1)$ we will not be able to distinguish between $\mathcal{F}_1$ and $\mathcal{F}_2$.

▶ **Lemma 8** ([6]). *Consider a $\left(\frac{1}{2}-\epsilon, \frac{1}{2}+\epsilon\right)$-biased coin (that is, a coin whose most likely outcome has probability $\frac{1}{2}+\epsilon$. To determine with probability at least $1-\delta$ what is the most likely outcome of the coin, one needs at least $\Omega(1/\epsilon^2 \log(1/\delta))$ coin flips.*

By Lemma 7 with probability $1 - o(1)$, the expected number of STARRED centers will be $\frac{8q}{dt(1-o(1))}$. Let QUERIED be the set of queried vertices by the algorithm. If $q = \omega(dt)$

$$\mathbb{P}\left(\#\ \text{QUERIED} \cap \text{STARRED}_{\text{ center}} \geq 48q/dt\right) \leq e^{-16q/dt} \leq o(1).$$

Therefore, since in order to distinguish between $\mathcal{F}_1$ and $\mathcal{F}_2$ with probability at least $1-\delta$, we need at to see at least $\Omega\left(\log(1/\delta)(1/\epsilon^2)\right)$ starred centers, or equivalently $\Omega\left(dt \log(1/\delta)(1/\epsilon^2)\right)$ queries. □

4 Conclusion

In this paper we have studied the complexity of computing a number of functions of online graphs, such as online social networks, in terms of their average degree and their mixing time. We have obtained a tight bound for the problem of computing the average of a bounded function on the vertices of the graph (e.g., the average star rating of a movie), and a near-tight bound for the problem of sampling a close-to uniform-at-random vertex (many algorithms in the literature assume to have access to such an oracle), and lower bounds for the problems of estimating the order, and the average degree of the graphs.

It will be interesting to pursue the study of these online graphs problems in order to bridge the gap between theoretical algorithms, and applied ones. Besides the obvious questions (what are the optimal bounds for estimating the order and the average degree of a graph?), an interesting open problem is to understand which structural properties of online social networks could be used by algorithms to improve the complexity of the various problems that practitioners have been considering.

References

1 Yong-Yeol Ahn, Seungyeop Han, Haewoon Kwak, Sue Moon, and Hawoong Jeong. Analysis of topological characteristics of huge online social networking services. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 835–844, New York, NY, USA, 2007. ACM. doi:10.1145/1242572.1242685.

2 Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. Efficient algorithms for large-scale local triangle counting. *ACM Trans. Knowl. Discov. Data*, 4(3):13:1–13:28, October 2010.

3 Anna Ben-Hamou, Roberto I. Oliveira, and Yuval Peres. *Estimating graph parameters via random walks with restarts*, pages 1702–1714. SIAM, 2018. doi:10.1137/1.9781611975031.111.

4 E. Blais, C. Canonne, S. Chakraborty, G. Kamath, and C. Seshadhri. Property testing review,the latest in property testing and sublinear time algorithms(blog post). URL: https://ptreview.sublinear.info/?p=918.

5 Marco Bressan, Enoch Peserico, and Luca Pretto. Simple set cardinality estimation through random sampling. arXiv:1512.07901, 2015.

6 Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, 53(1):17–25, 1995. doi:10.1016/0020-0190(94)00171-T.

7 Flavio Chierichetti, Anirban Dasgupta, Ravi Kumar, Silvio Lattanzi, and Tamás Sarlós. On sampling nodes in a network. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 471–481, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee. doi:10.1145/2872427.2883045.

8 Kai-Min Chung, Henry Lam, Zhenming Liu, and Michael Mitzenmacher. Chernoff-Hoeffding bounds for markov chains: Generalized and simplified. In Thomas Wilke Christoph Dürr, editor, *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*, volume 14, pages 124–135, Paris, France, 2012. LIPIcs. URL: https://hal.archives-ouvertes.fr/hal-00678208.

9 Colin Cooper, Tomasz Radzik, and Yiannis Siantos. Estimating network parameters using random walks. *Social Network Analysis and Mining*, 4(1):168, 2014. doi:10.1007/s13278-014-0168-6.

10 Anirban Dasgupta, Ravi Kumar, and Tamas Sarlos. On estimating the average degree. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 795–806, New York, NY, USA, 2014. ACM. doi:10.1145/2566486.2568019.

11 Anirban Dasgupta, Ravi Kumar, and D. Sivakumar. Social sampling. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 235–243, New York, NY, USA, 2012. ACM. doi:10.1145/2339530.2339572.

12 Talya Eden and Will Rosenbaum. On sampling edges almost uniformly. arXiv:1706.09748, 2017.

13 Uriel Feige. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM J. Comput.*, 35(4):964–984, 2006. doi:10.1137/S0097539704447304.

14 Minas Gjoka, Maciej Kurant, Carter T. Butts, and Athina Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *Proceedings of the 29th Conference on Information Communications*, INFOCOM'10, pages 2498–2506, Piscataway, NJ, USA, 2010. IEEE Press. URL: http://dl.acm.org/citation.cfm?id=1833515.1833840.

15 Oded Goldreich. *Introduction to Testing Graph Properties*, pages 105–141. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. doi:10.1007/978-3-642-16367-8_7.

16 Oded Goldreich and Dana Ron. Approximating average parameters of graphs. *Random Struct. Algorithms*, 32(4):473–493, 2008. doi:10.1002/rsa.v32:4.

17 Varun Kanade, Frederik Mallmann-Trenn, and Victor Verdugo. How large is your graph? *CoRR*, abs/1702.03959, 2017. URL: http://arxiv.org/abs/1702.03959, arXiv:1702.03959.

18 Liran Katzir, Edo Liberty, Oren Somekh, and Ioana A. Cosma. Estimating sizes of social networks via biased sampling. *Internet Mathematics*, 10(3-4):335–359, 2014. doi:10.1080/15427951.2013.862883.

19 Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 177–187, New York, NY, USA, 2005. ACM. doi:10.1145/1081870.1081893.

20 Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and MichaelW̃. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Math.*, 6(1):29–123, 2009. URL: http://projecteuclid.org/euclid.im/1283973327.

21 Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 29–42, New York, NY, USA, 2007. ACM. doi:10.1145/1298306.1298311.

22 Thomas Schank and Dorothea Wagner. Approximating clustering coefficient and transitivity. *Journal of Graph Algorithms and Applications*, 9:265–275, 2005. doi:10.7155/jgaa.00108.

23 C. Seshadhri, Ali Pinar, and Tamara G. Kolda. Wedge sampling for computing clustering coefficients and triangle counts on large graphs. *Stat. Anal. Data Min.*, 7(4):294–307, 2014. doi:10.1002/sam.11224.