Interactive Proofs for Social Graphs

Liran Katzir
lirank@google.com
Google Research

Clara Shikhelman
clara.shikhelman@gmail.com
Chaincode Labs

Eylon Yogev
eylony@gmail.com
BU and TAU

November 15, 2020

Abstract

We consider interactive proofs for social graphs, where the verifier has only oracle access to the graph and can query for the i^{th} neighbor of a vertex v, given i and v. In this model, we construct a doubly-efficient public-coin two-message interactive protocol for estimating the size of the graph to within a multiplicative factor $\varepsilon > 0$. The verifier performs $\widetilde{O}(1/\varepsilon^2 \cdot \tau_{mix} \cdot \Delta)$ queries to the graph, where τ_{mix} is the mixing time of the graph and Δ is the average degree of the graph. The prover runs in quasi-linear time in the number of nodes in the graph.

Furthermore, we develop a framework for computing the quantiles of essentially any (reasonable) function f of vertices/edges of the graph. Using this framework, we can estimate many health measures of social graphs such as the clustering coefficients and the average degree, where the verifier performs only a small number of queries to the graph.

Using the Fiat-Shamir paradigm, we are able to transform the above protocols to a non-interactive argument in the random oracle model. The result is that social media companies (e.g., Facebook, Twitter, etc.) can publish, once and for all, a short proof for the size or health of their social network. This proof can be publicly verified by any single user using a small number of queries to the graph.

Keywords: interactive proofs; social graphs; succinct arguments

Contents

1 Introduction

Social networks have become a large and influential part of the everyday lives of billions of people. The study and analysis of social networks is a modern approach to understand human relationships and, as such, it has gained a vast amount of attention from researchers in areas spanning from psychology and public policy to game theory and computer science. As many of these networks contain immense amounts of data [CEKLM15], new tools ought to be developed to facilitate new demands.

From a computational point of view, a social network is modeled as a graph. Very abstractly, a node represents an individual (or an entity), and an edge represents a relationship between two individuals (we note that a social graph can represent other entities as well, such as companies, objects of interest, or virtual asserts such as a webpage and more). This abstraction is limited yet has been shown very fruitful for studying social relationships with applications ranging from economics to health. Social networks usually share common structural properties such as homophily, the existence of clusters, the small-world phenomena, heterogeneous distributions of friends, and community structure [Bre12; EK10; Kle00]. Some of these properties can be explained using known measures of graphs such as small mixing time (e.g., a random walk converges rapidly to its stationary distribution), small average distance between a pair of nodes (the "small world" phenomenon), and small average degree.

The company or the network's provide (the entity in hold of the data of the network) in most cases regularly publishes data (either to the public or to a group of interest) regarding the number of (active) users and other "health" measures for various commercial and sociological purposes. However, as a community, it is crucial to have an *independent* estimate of these measures, and in particular, one that does not have a blind trust in the provider's reports, which are amenable by financial or political incentives. For example, Facebook has acquired WhatsApp at a steep price of 16 billion dollars which was computed by a 40 dollar evaluation per user of the platform [Wha], giving WhatsApp incentive to increase their network size in the reports[1].

Two main challenges stand in the path to performing independent estimates of social graphs health measures. First, the graphs are huge, which makes it infeasible to simply obtained the data on a standard machine and perform arbitrary computations. Second, and perhaps more importantly, the data is usually not freely available to obtain, but instead, is accessible via the *public interface of the network*. Typically, and as considered in this work, the interface includes queries to check membership of a specific node's ID, querying for the node's neighbors, fetching meta-data as its degree, and so on. Furthermore, for security and data proprietary reasons, access to this interface is throttled where it is forbidden to perform a large number of queries in a short time.

As a result, to face these challenges, there has been a line of work focused on estimating the size of social graphs, and other measures, via its public interface while performing few queries as possible. The main ingredient in these works is that the public interface allows performing a "random walk" in the graph and since such graphs have good mixing-time one can reach the stationary distribution with only a few queries. For example, the public interfaces have been used in [GKBM10; HRH09; RT10] to estimate the assortativity coefficient, degree distribution, and clustering coefficients of online social networks, as well as in [YW11; HRH09; KLSC14; KBM12; KH15] to estimate the number of registered users.

Similar techniques are used for search engines as well, as they provide a public interface as part of

[1]We are not suggesting that WhatsApp behave untruthfully, but merely that they had the incentive to do so.

their service. Web results (documents) which fit a search engine queries induces a bipartite query-document graph structure. Search engine public interfaces have been used in [BYG08; Bro+06; BYG11] to estimate corpus size, index freshness, and density of duplicates using search engine queries. In [BYG09] the authors provide a way to estimate the impression rank of a webpage. In [ZLAZ11] the number of YouTube videos is estimated by sampling link prefixes.

One of the most basic property of a social graph that one would study is the *size* of the graph (the number of nodes in the network). If the graph has n nodes, then using birthday paradox techniques, the number of queries required to estimate n is roughly $\sqrt{n}$, assuming queries return a uniform node at random [YW11]. In a more recent work, [KLSC14] have shown that the random walks provide a biased sampling of the nodes in the graph which, under certain assumptions on the graph, allow to get a biased version of the birthday paradox that uses only (roughly) $O(n^{1/4})$ queries. In [KMV17] it is shown that this bound is tight, if one insists on solely using the public interface. This is a huge improvement over a naïve $O(n)$ solution but still leaves much to be desired. This leads us to ask:

Can we estimate social graph measures using **few** *queries to the network while having* **no** *trust in the graph's provider?*

We propose a way to, on the one hand, use the power of the network provider (e.g., Facebook, Twitter, YouTube, Linkedin, etc.) for computation, and on the other hand, put no trust in them. That is, we propose *interactive proof for social graphs*, as a solution to this problem and as a generalization of the known interactive protocols [GMR89]. In this model, we request the network's provider to not only provide measures such as the size of the graph but, in addition, provide a "proof" of their claims. The proof will come in the form of an interactive protocol between a verifier (a weak computation device that can perform a small number of queries to the public interface) and a prover (the network's provider that has full access to the graph).

As the verifier cannot query the entire graph, we cannot hope to compute the precise graph size. Instead, we relax this requirement and settle for an approximation of the size of any measure in mind. As is often the case in interactive proofs, we require two properties from the interactive protocol. Completeness: if the prover is honest and provides the right measures with the prescribed proof, then the verifier will accept the claims (with high probability). Soundness: if a cheating prover submits claims that are *far* from the correct ones, then no matter what proof it provides, the verifier will reject its claims (with high probability). This way, we can efficiently estimate the graph's size without putting any trust in the network itself.

Our model assumes that the graph itself is fixed, and we only interact with a prover that has access to this graph but cannot make changes to it. One could imagine that a social network could create an alternative fake view of the network with a large number of fake users and connections which would fool the verifier. However, there are several reasons why this would not happen and we stress two points.

First, it is impossible to distinguish between queries issued by a verifier and by legitimate users. Thus, in order for the prover to cheat the cheating prover must consistently change the graph for all users, and not only for a specific verifier. Second, without our protocol, the social network can cheat simply by publishing wrong reports about their network (which happens in many cases). Using our protocol, any cheating report has to be materialized and maintained in the network in a way that affects all users. While this is still possible, it puts a huge burden on the cheating party.

1.1 Our results

We develop a framework for interactive proofs for social graphs. The first main result is a public-coin interactive protocol for estimating the size of a social graph, up to a small multiplicative error ε for any $\varepsilon > 0$. Let G be a graph of size n and let $\widetilde{n}$ be the claimed size. The protocol verifies that $\widetilde{n} \in (1 \pm \varepsilon)n$. The verifier performs a small number of queries (depending on the mixing-time and average degree of the graph), and the prover is quasi-linear in the size of the graph. This improves upon all previous works in terms of the number of queries (however, with the help of a prover).

We stress that we have no precise definition of a "social graph" and instead our results apply to any graph where the complexity of the protocol depends on different measures of the graph (e.g., mixing time) which are relatively small for social graphs. We show the following theorem.

Theorem 1.1 (informal). *Let G be a graph with n nodes, mixing time τ_{mix} and average degree Δ, and let $\varepsilon > 0$. There is a two message public-coin interactive proof for estimating the size of a graph in the social graph model, within an error of ε where the verifier's query complexity, running-time and the communication complexity are all bounded by $\widetilde{O}(1/\varepsilon^2 \cdot \tau_{mix} \cdot \Delta)$, and the prover runs in time $\widetilde{O}(n \cdot 1/\varepsilon^2)$.*

The theorem above is shown via a more general technique of estimating the size of any set S where the verifier has limited access to the set, in our case, the set will be the set of vertices in the graph. The verifier has membership queries to the set and an efficient procedure that can sample a uniform element from the set (which in our case corresponds to a random walk in the graph). The precise theorem is given in Theorem 3.1.

Estimating the size of a social graph (or in general a set) is perhaps the first and most basic property one would like to know about the graph. However, many other more involved complexity measures are desired as well, including the average degree, the local clustering coefficients, and others. Towards this end, we develop a general framework for estimating a large class of measures that include the most popular and studied ones, such as the examples given above.

For essentially any computable function f which is applied on a single vertex of the graph, let $A_f(q)$ a value that is the q-th quantile of the vertices of the graph when applying the function f. As the first theorem, this theorem, too, is shown via a more general approach to estimate the average value $f(x)$ for any set S.

Theorem 1.2 (informal). *Let G be a graph with n nodes and mixing time τ_{mix}, and let f be a computable function. There is a two-message protocol that (given n) computes $\widetilde{A}_f(q)$ such that*

$$\widetilde{A}_f(q) \in [A_f((1-\varepsilon)q), A_f((1+\varepsilon)q)] \ ,$$

with soundness error $1/3$, completeness error $1/3$, communication complexity $\widetilde{O}(1/\varepsilon^2)$, verifier query complexity $\widetilde{O}(1/\varepsilon^2 \cdot \tau_{mix})$, and prover running-time $\widetilde{O}(1/\varepsilon^2 \cdot n)$.

The precise theorem (for general sets) is given in Theorem 4.1.

Applications of our framework. The structure of social networks is studied via a set of common structural measures which reflect the health and authenticity of the network (see [CRTB06] for a survey of major structural measurement). In [MMGDB07] a large-scale measurement study and analysis of the structure of multiple online social networks. Our framework is useful for estimating various different structural measures of social graphs. The core measures are average degree; degree distribution; and local clustering coefficient distribution. [CRTB06]. In Section 5 we elaborate on these applications.

Social graphs and society. Social media companies are subject to severe public critique in the past years. As the understanding of the effects of social media grows, the companies are expected to fight bots pretending to be users, echo chambers, and other unfavorable phenomena (see, e.g., [GDFMGM18; ACELP13] and references therein). The protocol proposed in this paper can be used by the companies to prove that the issues are under control, and by society to hold the companies accountable. In Section 5.6 we elaborate on this further.

Non-interactive arguments for social graphs. One important property of our protocols is that they are public-coin protocols, meaning that the verifier has no secret random coins and the messages it sends are merely the random coins he flips. This has two main benefits. First, it ensures that the protocol is secure even if the social graph observes the verifier's queries. The verifier might perform queries that are later not sent to the prover and leak information about its private randomness. Since the social graph is the entity that implements these oracles to the graph, these private queries are indeed leaked to it. Our public-coin protocol removes any concern of this type.

Second, and even more importantly, public-coin protocols can be transformed, via the Fiat-Shamir transformation, to non-interactive arguments systems, in the random oracle model. This is achieved by having the prover use the random oracle for getting the random challenge for a specific prover message. The result is a short (e.g., polylog-sized) "proof" that can be published once and for all by a social graph company (Facebook, Twitter, etc.) and can be later *publicly* verified by any single user using only few (i.e., polylog) queries to the network (in the random oracle model, the scheme is even more practical and does not hide any large constants). These proofs can be used for publishing the company's annual report and even possibly used in court.

1.2 Related work

Goldwasser and Sipsers [GS89] presented a general technique that transforms any interactive protocol to a public coin protocol. At the heart of their transformation is a public-coin interactive proof for lower bounding the size of a specific set that is given implicitly to the verifier (he has only membership access to the set). Our work strengthens this result in several ways. First, we provide an upper bound together with the lower bound, which gives a complete estimation of the size of the set (in [GS89] a lower bound was sufficient for their proof). Second, we provide an arbitrary approximation with ratio ε where in [GS89] the ratio was assumed to be 2. We note that the protocol given in [GS89] does not work for better approximations ratios (it is not merely a matter of better analysis, but a protocol change is required).

There is a generic way to reduce a general approximation ratio of ε to 2, by taking the Cartesian product of the set with itself. That is, if S is the set, we estimate the size of S^k which contains all k tuples from S. The main problem with this is that the set S^k has size $|S|^k$, and thus, the running time of the prover is at least $|S|^k$, which is not realistic for most sets S and appropriate values k.

Fortnow [For87] also gave a protocol for estimating the size of a general set (he gives a lower bound and an upper bound), however, in his work the approximation ratio assumed is super-constant (in fact, his protocol allows distinguishing between a set of size n and a set of size n^2). Moreover, the upper bound he gives is a *private-coin* protocol where our protocol is public-coin. As discussed, a public-coin protocol is desirable, in part for applying the Fiat-Shamir transformation to get a corresponding non-interactive argument. Note that one cannot simply apply the [GS89] transformation to make the protocol of [For87] public-coin as this transformation is proven in the standard model (where the verifier has full access to the input) and, moreover, this transformation does not preserve the running-time of the prover (where we aim to have an efficient and practical prover).

Interactive proofs of proximity. In our protocol, the verifier is sublinear in its input and soundness condition only asserts that what we compute is close to the real value. This can be modeled as an *interactive proof of proximity* (IPP) [EKR04; RVW13]. One can view IPPs as the property testing analogue of interactive proofs. More concretely, consider the representation of the graph as a long indicator vector of size $|U|$ where U is the universe of all possible vertex IDs. Then, what we develop here is an IPP for the hamming weight of this vector, which corresponds to the size of the graph. However, there are critical differences in our goal compared to what is known for IPPs. Our protocols achieve a multiplicative error ε with respect to the *size of the graph*. On the other hand, in an IPP, the distance and error are with respect to $|U|$, the size of this long vector. Thus, any hamming weight IPP would not be useful in our setting.

1.3 Paper organization

In Section 2 we give a formal definition of the graph query model in which our interactive protocols take place, and provide additional required preliminaries.

In Section 3 we show our first main result which is a general protocol for estimating the size of any set, where the verifier is given only oracle access to the set. The proof is split into two parts, one for the lower bound and one for the upper bound (each in its own subsection).

Then, in Section 4 we build upon this protocol and construct a general framework for estimating a large class of functions. The section begin with an overview of our techniques and then the formal statement and proof.

In Section 5 we combine the previous sections (that apply to any set) and show how to instantiate them on a social graph using random walks and get our final theorem statements. Additionally, we demonstrate the usefulness of our framework and show how to estimate various different social graph measures.

Finally, in Section 6 we show how to apply the Fiat-Shamir transformation to get a corresponding non-interactive version of our protocols, in the random oracle model.

2 Model definition and preliminaries

Graph notations. Throughout the paper, we will consider a social graph. The graph is denoted by G, its set or vertices is denoted by V and the set of edges is denoted by E. Usually, n represents the size of the graph, i.e., the number of vertices in the graph, where m usually represents the number of edges. Using this notation, $|V| = n$ and $|E| = m$. Additionally, we denote by d_i the degree of vertex i and the sum of degrees by $D = \sum_{i=1}^{n} d_i = 2|E|$. The maximum degree of a graph is noted by $d_{\max} = \max_{i=1}^{n} d_i$.

2.1 Interactive proofs

An interactive protocol (P, V) for a language L is a pair of interactive Turing machines; for a instance x we denote by $\langle P(x), V(x)\rangle$ the output of V in an interaction between P and V on common input x.

Definition 2.1. *An interactive protocol (P, V) is an interactive argument system for a language $L \subset \{0,1\}^*$, if V is PPT and the following conditions hold:*

- ***Completeness**: For every $x \in L$:* $\Pr[\langle P(x), V(x)\rangle = 1] \geq 2/3$.
- ***Soundness**: For every P^*, for every $x \notin L$:* $\Pr[\langle P^*(x), V(x)\rangle = 1] \leq 1/3$.

We refer to (P, V) as a public-coin proof system if V simply sends the outcomes of its coin tosses to the prover (and only performs computation to determine its final verdict). Note that the soundness error and completeness error can be arbitrary amplified to 2^{-k} by repeating the protocol in parallel $O(k)$ times.

2.2 The graph query model

In this paper, we construct interactive protocols for languages where the verifier is sublinear and has specific oracle access to the input. In particular, we consider languages of graphs. Let U be a universe of elements (e.g., U is the set of all possible IDs of a vertex in the graph). Let $G = (V, E)$ be a graph where $V \subseteq U$ and let $n = |V|$. Every element in $x \in V$ is described using $\log |U|$ bits and this will be our "word" size $w = \log |U|$. In our model of computation, we assume that algorithms can perform operations on words of size w in constant time. The verifier has oracle access to such a graph G where the oracle ρ_G is given as follows:

$$\rho_G \colon U \times \{0, 1, \ldots, |U|\} \longrightarrow U \cup \{0, 1, \ldots, |U|\} \cup \{\perp\} \ .$$

The oracle ρ_G gets an element x and an index i and returns the i^{th} neighbor of x in the graph G. If x is not in the graph then it returns $\perp$. If x has less than i neighbors then it returns $\perp$. If $i = 0$ then it returns the number of neighbors that x has in the graph (this is why the range of ρ_G is $U \cup 0, 1, ..., |U|$). Formally, we define

$$\rho_G(x, i) = \begin{cases} \perp & \text{if } x \notin V \text{ or } \deg(x) < i \\ \deg(v) & \text{if } x \in V \text{ and } i = 0 \\ u & \text{if } x \in V \text{ and } u \text{ is the } i^{th} \text{ neighbor of } v \end{cases} \ .$$

We consider algorithms that have oracle access to ρ_G. An oracle algorithm A with oracle ρ_G can perform queries to ρ_G and get a response in a single time unit cost. Thus, the running time of A is a bound on the actual computation time and the number of oracle queries it performs. Using this, we can define an interactive protocol between a verifier and a prover where the prover has complete access to the graph and the verifier has only ρ_G query access, and performs a small number of queries.

Definition 2.2 (Interactive proofs in the graph query model)**.** *Let G be a graph and let $\langle P(G), V(G)\rangle$ be an interactive protocol where the instance is the graph G. We say that the protocol is in the* **graph query model** *if the verifier V has only oracle access to ρ_G (the prover has explicit access to G). Additionally, we give the verifier V an arbitrary node x_0 in the graph.*

Formally, we write the completeness and soundness conditions as follows.

- ***Completeness**: For every $G \in L$:*

$$\Pr[\langle P(G), V^{\rho_G}(x_0)\rangle = 1] \geq 2/3 \ .$$

- ***Soundness***: *For every unbounded P^*, and every $G \notin L$,*

$$\Pr[\langle P^*(G), V^{\rho_G}(x_0)\rangle = 1] \le 1/3 \ .$$

2.3 Additional preliminaries

Theorem 2.3 (Chernoff-Hoeffding)**.** *Let $X = \sum_{i=1}^m X_i$ be the mean of m independent Bernoulli (indicator) random variables where $\mathbb{E}[X_i] = p_i$. Let $\mu = \mathbb{E}[X] = \sum_{i=1}^m p_i$. Then,*

1. $\Pr[X \ge (1+\delta)\mu] \le e^{-\frac{\delta^2}{2+\delta}\cdot\mu}$ *for all* $\delta > 0$*.*

2. $\Pr[X \le (1-\delta)\mu] \le e^{-\frac{\delta^2}{2}\cdot\mu}$ *for all* $0 < \delta < 1$*.*

Limited independence. A family of functions $\mathcal{H}$ mapping domain $\{0,1\}^n$ to range $\{0,1\}^m$ is k-wise independent if for every distinct elements $x_1, \dots, x_k \in \{0,1\}^n$, $y_1, \dots, y_k \in \{0,1\}^m$, we have $\Pr_{h \in \mathcal{H}}[h(x_1) = y_1 \wedge \dots \wedge h(x_k) = y_k] \le \frac{1}{2^{km}}$.

Theorem 2.4. *There exists a family $\mathcal{H}$ of k-wise independent functions from $\{0,1\}^n$ to $\{0,1\}^m$ such that choosing a random function from $\mathcal{H}$ requires $O(k \cdot n)$ bits.*

Definition 2.5. *We say that $G : \{0,1\}^d \to \{0,1\}^n$ ε-fools a circuit class $\mathcal{C}$ if for every $C \in \mathcal{C}$ it holds that*

$$\left| \Pr_{s \leftarrow \{0,1\}^d}[C(G(s)) = 1] - \Pr_{x \leftarrow \{0,1\}^n}[C(X) = 1] \right| \le \varepsilon \ . \tag{1}$$

3 Set cardinality interactive proof

In this section, we provide a general interactive protocol for estimating the size of a (large) given set, with a sublinear verifier that has only oracle access to the set. Looking ahead, we will apply this protocol on the nodes (and edges) of the social graph. The task is formulated as follows. There is a universe U and a subset $S \subset U$ of interest. Our goal is to (approximately) compute the size of the set to within a multiplicative error ε. The prover has complete access to the set S while the verifier has only limited access to S given by two methods. The first method is a *membership* oracle where the verifier can specify an element $x \in U$ and gets as response whether $x \in S$ or not. The second method is an efficient random algorithm $\mathcal{D}$ that samples a *uniformly* random element in the set. We say that $\mathcal{D}$ is a sampler for the set S that uses ℓ bits if $\mathcal{D}$ can sample a uniform element in S using at most ℓ random bits.

One can easily observe that it is impossible to verify that indeed the size of the set S is *exactly* n while using a sublinear (in n) number of queries. However, here our goal is to verify that the size of the set is *approximately* n. That is, for every $\varepsilon > 0$ we want a protocol that assures that a given alleged size $\widetilde{n}$ satisfies $\widetilde{n} \in (1 \pm \varepsilon)|S|$ with probability at least 2/3.

To formally define this as an interactive proof for a specific language, we define the language $L_{S,\varepsilon}$

$$L_{S,\varepsilon} = \{(S, \widetilde{n}) : (1-\varepsilon)|S| \le \widetilde{n} \le (1+\varepsilon)|S|\} \ .$$

The main focus is to minimize the number of queries the verifier performs (both membership queries and sampling queries). For practical use cases, we additionally want the running-time of the prover to actually be quasi-linear in the size of the set S (which is achieved in our protocol). In the following theorem the notation $\widetilde{O}$ hides polylog(n) factors.

Theorem 3.1. *Let S be a set of size n with a sampler $\mathcal{D}$ that uses ℓ random bits. Let $\varepsilon > 0$ be a parameter and let $L_{S,\varepsilon}$ be the language defined as above. Then, there is an public-coin protocol for $L_{S,\varepsilon}$ with the following properties: (1) The protocol has two messages (first the verifier then the prover); (2) The verifier performs at most $\widetilde{O}(1/\varepsilon^2)$ queries (to both $\mathcal{D}$ and the membership oracle); (3) The verifier's running time and the communication complexity are bounded by $\widetilde{O}(1/\varepsilon^2 \cdot \ell)$; and (4) The prover runs in time $\widetilde{O}(n \cdot 1/\varepsilon^2)$.*

We split the proof into two parts, one for the lower bound and one for the upper bound. The protocols are described separately for ease of presentation and since each might be used independently in different contexts. The final protocol is obtained by simply running the two subprotocols in parallel (without a cost in the round complexity).

Let n be the *real* number of nodes in the graph and let $\widetilde{n}$ be the number of nodes in the graph *claimed* by the prover. Throughout, assume for simplicity that $0 < \varepsilon \leq 1/2$. Recall, our goal is to verify that $\widetilde{n} \in n \pm \varepsilon n$. We begin with the lower bound.

3.1 Lower bound

Overview. The general idea of the protocol is inspired by the Goldwasser-Sipser protocol [GS89]. Goldwasser and Sipser provide a method to compile any private coin protocol to a corresponding public-coin protocol. At the heart of their transformation was a protocol proving that a certain exponentially large set, given implicitly to the verifier, is indeed large enough. The verifier has only membership access to this set.

In the protocol, the verifier sends a (pairwise independent) hash function $h \in \mathcal{H}$ that hashes the elements of the set to a range of size $\widetilde{n}$. The prover is asked to find an element that is mapped to a specific bin (i.e., an element in the range of the hash). Their main observation is that is the set is indeed large enough, then such an element exists with higher probability than if the set is smaller. This suffices for catching a cheating prover (with some probability) and at the end the soundness and completeness of the protocol are amplified.

A better statistical measure. Put in more general terms, the verifier picks a random hash function, and is concerned with how the elements of the set are distributed among the different bins. The Goldwasser-Sipser protocol asks the prover to send a simple "statistical measure" of proving that a random bin is not empty. This measure was sufficient for their protocol as they had a factor of 2 separating from a large set and a small set, however, their protocol cannot separate smaller differences in the sizes, and in our case, we only have a factor of $(1 + \varepsilon)$.

There is a generic way to amplify this factor by taking the Cartesian product of the set with itself. That is, if S is the set, we estimate the size of S^k which contains all k tuples from S. The main problem with this approach is that it does maintain the running of the prover which is now at least $O(|S|^k)$ (k is not even a constant in general, and we aim for linear time prover).

Instead, we look at a more involved statistic, namely, *the size of the preimage*. The verifier samples a single hash function $h \in \mathcal{H}$ which is sent to the prover. The prover responds with all the preimages z of fixed output y with respect to h. The verifier asserts that z is indeed a set of preimages, and then counts the number of elements in z and accepts if and only if it is higher than its expectation (minus a small fraction).

The hash family we use is a t-wise independence hash family. Let $\mathcal{H} = \{h \colon U \to [c \cdot \widetilde{n}/k]\}$ be a family of t-wise independent hash functions, where t and c are a parameters that will be defined later. The c parameter is used to ensure the h's co-domain(range) is a power of 2 and also to control the error probability. It would be easier for the reader to think of the hash functions as being completely random, and moreover, the analysis will be first shown under this assumption. Then, in Section 3.3 we show how the same analysis (with negligible changes in the soundness and completeness) holds for our family of t-wise independent hash functions.

The formal description of the protocol is given in Figure 1.

Set cardinality lower bound

Parameters: c and $k = \frac{4^3}{\varepsilon^2}$.

1. V $\Rightarrow$ P: Verifier samples $h \in \mathcal{H}$ where $h \colon U \to [c \cdot \widetilde{n}/k]$, and $y \in [c \cdot \widetilde{n}/k]$ and sends them to the prover.
2. P $\Rightarrow$ V: Prover sends the set of preimages $Z = h^{-1}(y) \cap S$.
3. Verifier checks that:
   (a) $Z \subseteq S$ (by a membership query to each element $z \in Z$); and
   (b) $\forall z \in Z,\ h(z) = y$ (and rejects otherwise).
4. The verifier accepts iff $|Z| > (1 - \varepsilon/4) \cdot c \cdot k$.

Figure 1: Our lower bound protocol for the approximate size of a set.

Formally, this protocol is an interactive protocol for a similar language of $L_{S,\varepsilon}$ only where we consider only the lower bound. Thus, we define

$$L'_{S,\varepsilon} = \{(S, \widetilde{n}) : (1 - \varepsilon)|S| \le \widetilde{n}\} \ .$$

We show the following lemma which specifies the properties of the protocol with respect to parameter c.

Lemma 3.2. *Given a parameter c, and ε the protocol in Figure 1 is an interactive protocol for $L'_{S,\varepsilon}$ with the following properties: soundness error e^{-2c}; completeness error e^{-4c}; total communication $O(c/\varepsilon^2 \cdot \mathrm{polylog}(n))$; verifier running time $O(c/\varepsilon^2 \cdot \mathrm{polylog}(n))$; prover running time $O(1/\varepsilon^2 \cdot n \cdot \mathrm{polylog}(n))$.*

We turn to analyze the protocol (i.e., Figure 1). As mentioned, the completeness and soundness are first analyzed under the assumption that each h is a truly random function. After the proof, in Section 3.3 we show that the limited independence hash function suffices and compute its complexity (both in terms of the key size and the evaluation time), as this affects the communication complexity of the protocol and the verifier's running time.

Completeness. Assume that $\widetilde{n} = n$ and that the prover behaves honestly according to the protocol. Denote the elements of the set by $S = (x_1, \dots, x_n)$. Let Z_i be the indicator $h(x_i) = y$. Namely, $Z_i = 1$ iff $h(x_i) = y$. Then, since the hash function is random we get that $\mathbb{E}[Z_i] = c \cdot k/n$. Clearly $|Z| = \sum_i Z_i$, and thus we get

$$\mathbb{E}[|Z|] = \sum_{i=1}^{n} \mathbb{E}[Z_i] = n \cdot \frac{c \cdot k}{n} = c \cdot k \enspace .$$

Since we assumed that h is a completely random function, we get that we the random variables $Z_1, \dots, Z_n$ are independent. Using a Chernoff-Hoeffding bound (see Theorem 2.3) we entail

$$\Pr[\text{reject}] = \Pr[Z \leq (1 - \varepsilon/4) \cdot c \cdot k] \leq e^{-\frac{\varepsilon^2}{2 \cdot 4^2} \cdot c \cdot k} = e^{-2c} \enspace .$$

Soundness. Assume that the instance is not in the language, that is, $\widetilde{n} > (1 + \varepsilon)n$. Let $\widetilde{Z}$ be the set sent by the prover (i.e., the alleged Z). If the verifier did not reject at step (3) then it must be that $\widetilde{Z} \subseteq S$ and $\widetilde{Z} \subseteq Z$, and therefore $|\widetilde{Z}| \leq |Z|$. Again, let Z_i be the indicator $h(x_i) = y$. In this case, we have that

$$\mathbb{E}[|Z|] = \sum_{i=1}^{n} \mathbb{E}[Z_i] = n \cdot \frac{c \cdot k}{\widetilde{n}} = c \cdot k < c \cdot k \frac{n}{(1 + \varepsilon)n} = \frac{c \cdot k}{1 + \varepsilon} \enspace .$$

We bound from above the probability that the verifier accepts in this case.

$$\begin{aligned}
\Pr[\text{accept}] &\leq \Pr\left[|\widetilde{Z}| > (1 - \varepsilon/4) \cdot c \cdot k\right] \\
&\leq \Pr[|Z| > (1 - \varepsilon/4) \cdot c \cdot k] \\
&= \Pr[|Z| - \mathbb{E}[|Z|] > (1 - \varepsilon/4) \cdot c \cdot k - \mathbb{E}[|Z|]] \\
&\leq \Pr[|Z| - \mathbb{E}[|Z|] > (1 - \varepsilon/4)(1 + \varepsilon)\mathbb{E}[|Z|] - \mathbb{E}[|Z|]] \\
&= \Pr\left[|Z| - \mathbb{E}[|Z|] > (\varepsilon - \varepsilon/4 - \varepsilon^2/4)\mathbb{E}[|Z|]\right] \\
&\leq \Pr[|Z| - \mathbb{E}[|Z|] > \varepsilon/2 \cdot \mathbb{E}[|Z|]] < e^{-4c} \enspace .
\end{aligned}$$

This shows the soundness error and completeness error of the protocol. Notice that one can set c to be a small constant (e.g., $c = 2$) to get the standard 1/3 and 2/3 soundness error and completeness requirements. We now turn to show the other complexity measures of the protocol.

Complexity measures.

- *Queries.* The verifier performs $c \cdot k = O(c/\varepsilon^2)$ membership queries, one to each $z \in Z$. Note that there is no need for the prover to send Z whose size is bigger than $c \cdot k$. That is, if Z is larger than $c \cdot k$ then the prover simply sends an arbitrary subset $Z' \subset Z$ of size $c \cdot k$. This way, the verifier performs at most $O(c \cdot k)$, and it also bounds the communication complexity.

- *Communication.* We bound the total amount of communication in the protocol (both from prover to verifier and from verifier to prover). The verifier sends a hash function with $O(t)$ bits, where $t = \text{polylog}(n)$ is the independence of the hash function. The prover replies with a set that have total size of $O(c \cdot k)$ (see argument above) and thus can be specified using $O(c \cdot k \cdot \log n)$ bits. The total communication is thus $O(c \cdot k \cdot \text{polylog}(n)) = O(c/\varepsilon^2 \cdot \text{polylog}(n))$.

- *Verifier running-time.* The verifier is rather simple and its only computation besides choosing randomness and reading the communication is to compute the hash function h on all $z \in Z$. The hash function can be computed in time polylog(n) for a single element, there are $O(c \cdot k)$ elements and thus the total running time of the verifier is $O(c/\varepsilon^2 \cdot \mathrm{polylog}(n))$.
- *Prover running-time.* The prover runs over all n nodes in the graph and checks if it a preimage of y. This takes time $O(1/\varepsilon^2 \cdot n \cdot \mathrm{polylog}(n))$.

3.2 Upper bound

Overview. In the previous subsection, we have devised a protocol for a lower bound on the set size. In this subsection, we provide a protocol for an *upper bound* on the size of the set. An upper bound for a set is somewhat counter-intuitive, as the prover can always act honestly on a small subset. Since the verifier has limited access to the set, it would be hard to notice this.

Here, we exploit the fact that the verifier has uniform random access to the set. If the prover ignores an ε fraction of the set, and the verifier samples $1/\varepsilon$ random elements then the verifier has a good chance of sampling an element that was ignored by the prover. However, even if this happens, how can the verifier check if the prover included this element in this proof? After all, the proof is very short compared to the large set (the set is of size n and the proof is roughly logarithmic in the size of the set).

A simple but expensive solution. Our protocol is based on the "birthday paradox" and works as follows. We begin with a simple and not query efficient protocol. We have the verifier sample roughly $O(\sqrt{n})$ random elements from the set. If indeed $n = \widetilde{n}$, then we expect to see a collision (i.e., two samples that ended with the same element). However, if the set is much larger, then such a collision is less likely. This is a good protocol, however, the verifier has a very high query complexity.

The actual protocol. Instead, we let the prover simulate this process for us. We ask the prover to perform $O(\sqrt{\widetilde{n}})$ samples and tell us if he saw a collision. This is very naive, as it is hard to force that prover to behave this way. Thus, we again use a random hash function. Here the goal of the hash is fixed some common randomness that is used in order to perform the $O(\sqrt{\widetilde{n}})$ samples.

Let $\mathcal{H} = \{h\colon U \to [\ell]\}$ be a family of hash functions (recall that ℓ is the number of bits used by the algorithm $\mathcal{D}$. Then, the prover uses the values of the hash function $h(i)$ as random bits to use to run the $\mathcal{D}$, and finds $1 \leq i \neq j \leq \widetilde{n}$ such that running the sampler with random coins $h(i)$ results in the same elements as with the coins $h(j)$, namely, $\mathcal{D}(h(i)) = \mathcal{D}(h(j))$. The prover sends i and j to the verifier which can easily verifier that $\mathcal{D}(h(i)) = \mathcal{D}(h(j))$ (and that $i \neq j$) with merely 2 queries!

The description above heavily uses the fact that the hash function h is truly random. Again, in Section 3.3 we show that a limited independent hash function suffices for this analysis as well. Formally, the protocol is given in Figure 2.

An upper bound for estimating the size of a general set

1. V $\Rightarrow$ P: Verifier samples $h \in \mathcal{H}$ and sends it to the prover.
2. P $\Rightarrow$ V: Prover does the following:
   (a) compute randomness $r_i = h(i)$ for all $i \in \sqrt{\widetilde{n}}$.
   (b) let $v_i \leftarrow \mathcal{D}(r_i)$.
   (c) sends the minimal $i \neq j \in \sqrt{\widetilde{n}}$ such that $v_i = v_j$.
3. V: Verifier computes $r_i = \mathcal{D}(h(i))$ and $r_j = \mathcal{D}(h(j))$ and asserts that: (1) $\mathcal{D}(r_i) = \mathcal{D}(r_j)$; (2) $i \neq j$; and (3) $i, j \in [\sqrt{\widetilde{n}}]$ and accepts.

Figure 2: An interactive protocol for ensuring an upper bound of the set size.

Completeness and Soundness. We turn to analyze the protocol. Here, we do not argue completeness and soundness separately but rather in a single joint argument. We denote the completeness by α and the soundness error by β and show that $\alpha - \beta = \gamma$, where γ is not too small. Then, using parallel repetition, we can expand the gap between the completeness and soundness error by an arbitrary constant of our choice. In particular, if γ is the gap, then, to get the desired completeness 2/3 and soundness error 1/3, the number of repetitions required is $O(1/\gamma)$ (see e.g., [HPWP10]). In our case, we show that $\gamma \geq \Omega(\epsilon)$ which means that we need $O(1/\varepsilon)$ repetitions of the protocol we describe above.

Define α and β as follows:

$$\alpha = \Pr\left[\exists i, j \in [\sqrt{\widetilde{n}}] : D(r_i) = D(r_j) \mid \widetilde{n} = n\right] \ ,$$
$$\beta = \Pr\left[\exists i, j \in [\sqrt{\widetilde{n}}] : D(r_i) = D(r_j) \mid \widetilde{n} \leq (1-\varepsilon)n\right] .$$

Note that α and β are indeed the completeness and soundness parameters of the protocol. We consider what is known to be "the birthday paradox". Given m uniformly chosen items $r_1, \ldots, r_m$ from a set of size n we have that

$$\Pr\left[r_1, \ldots, r_m \text{ are distinct}\right] = \prod_{i=0}^{m-1} \left(1 - \frac{i}{n}\right) \ .$$

We are interested in analyzing this formula. Thus, before we continue the proof of the completeness and soundness we provide the following technical lemma.

Lemma 3.3. *For any integer* $1 \leq k \leq \frac{n}{2}$,

$$e^{-\frac{k(k+1)}{2n} - \frac{(k+1)^3}{3n^2}} < \prod_{i=0}^{k} \left(1 - \frac{i}{n}\right) < e^{-\frac{k(k+1)}{2n}} .$$

Proof. First, we observe that

$$\prod_{i=0}^{k} \left(1 - \frac{i}{n}\right) = \exp\left(\sum_{i=0}^{k} \ln\left(1 - \frac{i}{n}\right)\right) \ .$$

Next, we note that $-x - x^2 < \ln(1-x) < -x$ holds for $0 < x \leq 1/2$ (the second inequality even holds for $0 < x < 1$). In addition, for every positive k we have $k(k+1)(2k+1) < 2(k+1)^3$. Finally,

$$-\frac{k(k+1)}{2n} - \frac{k(k+1)(2k+1)}{6n^2} < \sum_{i=0}^{k} \ln\left(1 - \frac{i}{n}\right) < -\frac{k(k+1)}{2n} \ .$$

□

Claim 3.4. *For any $x \geq 1$, $c > 0$ and $\varepsilon > 0$,*

$$ce^{-x} - e^{-\frac{x}{1-\varepsilon}} > e^{-x}\left[c - (1-\varepsilon)\right].$$

Proof.

$$\begin{aligned} ce^{-x} - e^{-\frac{x}{1-\varepsilon}} &= e^{-x}\left[c - e^{-\frac{x}{1-\varepsilon}+x}\right] = e^{-x}\left[c - e^{-x\frac{\varepsilon}{1-\varepsilon}}\right] \\ &> e^{-x}\left[c - (1-\varepsilon)^x\right] \geq e^{-x}\left[c - (1-\varepsilon)\right] \ . \end{aligned}$$

The for first inequality we used $e^{-\frac{z}{1-z}} < 1 - z$ for $z > 0$. The for second inequality we used $x \geq 1$. □

We choose k as the smallest integer such that $k(k+1) \geq 2n$. Using the lemma 3.3 and claim 3.4, we get

$$\begin{aligned} \alpha - \beta &> e^{-\frac{k(k+1)}{2n} - \frac{(k+1)^3}{3n^2}} - e^{-\frac{k(k+1)}{2n(1-\varepsilon)}} \\ &> e^{-\frac{k(k+1)}{2n}}\left(e^{-\frac{(k+1)^3}{3n^2}} - (1-\varepsilon)\right) \\ &> e^{-\frac{k(k+1)}{2n}}\left(1 - \frac{(k+1)^3}{3n^2} - (1-\varepsilon)\right) \\ &= e^{-\frac{k(k+1)}{2n}}\left(\varepsilon - \frac{(k+1)^3}{3n^2}\right) \ , \end{aligned}$$

where the three inequalities follow since: (1) the first inequality is due to lemma 3.3 and the definition of α and β; (2) the second inequality is due to claim 3.4 for $x = \frac{k(k+1)}{2n}$, $c = e^{-\frac{(k+1)^3}{3n^2}}$ and ε (and the choice of k); and (3) the third inequality is due to $e^{-x} > 1 - x$. Since $(k-1)^2 < (k-1)k < 2n \leq k(k+1)$, we get that $k < 1 + \sqrt{2n}$ which yields

1. $\frac{k(k+1)}{2n} < 1 + \frac{2+3\sqrt{2n}}{2n}$;
2. $\frac{(k+1)^3}{3n^2} < \frac{(2+\sqrt{2n})^3}{3n^2} < \frac{2}{\sqrt{n}}$.

For any $n \geq 1000$, we get $e^{-\frac{k(k+1)}{2n}}\left(\varepsilon - \frac{(k+1)^3}{3n^2}\right) > \frac{1}{4}\left(\varepsilon - \frac{2}{\sqrt{n}}\right)$. Assuming $\varepsilon \geq \frac{2}{\sqrt{n}}$, we have $\alpha - \beta \geq \frac{\varepsilon}{12}$, as desired.

Complexity measures. We compute the complexity measures of the underlying protocol and then its cost for the amplified protocol. Recall that we perform parallel repetitions for $O(1/\varepsilon)$ repetitions. All the complexity measures below are actually subsumed by the upper bound.

- *Queries.* The verifier performs only two queries, for the given i and j. After amplification, the number of queries is $O(1/\varepsilon)$.
- *Communication.* The verifier sends h which takes $\operatorname{polylog}(n)$ bits. The prover simply sends i and j which are $O(\log n)$ bits. After amplification, the communication complexity is $O(1/\varepsilon \cdot \operatorname{polylog}(n))$.
- *Verifier running-time.* The verifier's running time is mainly the running time of $\mathcal{D}$, which is $O(\ell \cdot \operatorname{polylog}(n))$, and computing the hash function h which takes time $\operatorname{polylog}(n)$. After amplification, the running time is $O(1/\varepsilon \cdot \ell \cdot \operatorname{polylog}(n))$.
- *Prover running-time.* The prover runs over all $i \in [\sqrt{n}]$ and runs $\mathcal{D}$ on $h(i)$ and searches for collisions. The running time is $O(\sqrt{n} \cdot \operatorname{polylog}(n))$, and after amplification it is $O(1/\varepsilon \cdot \sqrt{n} \cdot \operatorname{polylog}(n))$.

3.3 Using explicit hash functions

The analysis of the above upper bound and lower bound were performed in a model where the sampled hash functions $h \in \mathcal{H}$ were assumed to be truly random functions. This approach is useful for making the analysis easy to read and is a complete proof in the random oracle model. Practically speaking, one could implement the protocol above using a heuristic hash function such as SHA256 and similar implementations as the random oracle. This would heuristically be secure and save a lot in terms of communication since the random oracle serves a large common source of randomness and thus eliminating the need of the verifier to send random coins.

Nevertheless, we show how to use *explicit* limited independent hashes functions to make the analysis *provably* secure where the description of the hash is small (it will be $\operatorname{polylog}(n)$). Thus, our protocol is secure in the standard model with no heuristics or any further assumptions.

We give different arguments for the lower bound and upper bound, but in both cases we rely on pseudorandomness. The main idea in both the lower bound and upper bound is similar and is inspired by [ANS09]. The analysis of the protocols relies on the probability of a certain event. For example, in the lower bound, we relied on the probability of Z to be large enough for completeness and to be small enough for soundness. Instead of using a completely random hash function, we use pseudorandom hash function. The main question is which definition of "pseudo" suffices for our protocol, which we now argue separately.

What we show here is that the event we rely on can be identified by a low depth circuit. Thus, it will suffice to use $\operatorname{polylog}(n)$ wise independent hash function as these fool AC^0 circuits.

Lower bound. Consider the event $\mathcal{E}$ that the set $Z = h^{-1}(y)$ has size at least $(1-\varepsilon/4)\cdot c \cdot k$. The completeness of the protocol shows that if the instance is in the language then the probability of $\mathcal{E}$ is high, and the soundness of the protocol shows that when the instance is not in the language then the probability of this event is low. The probability is taken over a truly random hash function h, where takes at most $|U| \cdot \log n$ bits to describe.

The key point is that this event can be identified by a low space algorithm, and thus we can use a pseudorandom generator to fool it. Define an algorithm C that has gets $x_1, \ldots, x_n$ and y and outputs 1 if and only if

$$|h^{-1}(y)| > (1 - \varepsilon/4) \cdot c \cdot k \ ,$$

where h is a truly random function defined by the random coins of the algorithm. What the analysis of the completeness of the protocol shows is that for any $x_1, \ldots, x_n$ and y it holds that $\Pr[C(x_1, \ldots, x_n, y) = 1] \geq 1 - e^{-4c}$. The soundness shows that in the "no" case it holds that $\Pr[C(x_1, \ldots, x_n, y) = 1] \leq e^{-2c}$

It is easy to see that this algorithm C can be implemented in $O(\log n)$ space. Indeed, the algorithm enumerates over all $x_1, \ldots, x_n$ and for each computes $h(x_i)$ by tossing coins (no need to remember the coins) and counting how many of them equal $h(x_i) = y$ (where or course counting can be done in small space).

Thus, it suffices to use Nisan's pseudorandom generator to fool algorithm C.

Theorem 3.5. *Let $\mathcal{C}$ be the family of algorithms computable in $\log m$ space. There is a PRG $G\colon \{0,1\}^{\log^2 m} \to \{0,1\}^m$ that $1/m$-fools $\mathcal{C}$.*

That is, we do not need the verifier to send $|U| \log n = \mathrm{poly}(n) = m$ random bits. Instead, it suffices for the verifier to send a seed of length $O(\log^2 n)$ to the prover which will then act the same on the m pseudorandom bits which the generator G provides. This has a negligible effects in the completeness and soundness error of the protocol and reduces the description of the hash function to $O(\log^2 n)$ bits.

Upper bound. We now move to the upper bound. In this protocol, we used the truly random property of the hash function where we claimed that for $r_i = \mathcal{D}(h(i))$ it holds that

$$\Pr\left[r_1, \ldots, r_m \text{ are distinct}\right] = \prod_{i=0}^{m-1} \left(1 - \frac{i}{n}\right) \ .$$

To argue this with an explicit hash function, we will the following theorem due to Braverman [Bra10] and its improvement by Tal [Tal17] and Harsha and Srinivasan [HS19].

Theorem 3.6 (Follows from [Bra10; Tal17; HS19])**.** *Let C be an AC^0 circuit of size m and depth d over n bits. Let μ be a distribution that is $r = r(m, d)$-independent over n bits and let U be the uniform distribution over n bits. Then, $|\mathbb{E}_{x \leftarrow \mu}[C(x)] - \mathbb{E}_{x \leftarrow U}[C(x)]| \leq \frac{1}{m}$ where $r = (\log m)^{O(d)}$.*

Let C be a circuit that gets an input a random string $R = r_1, \ldots, r_{\sqrt{n}}$, and works as follows:

$$C(r_1, \ldots, r_{\sqrt{n}}) = 1 \text{ if and only if } \mathcal{D}(r_i) \neq \mathcal{D}(r_j) \text{ for all } i \neq j \ .$$

Recall that $\mathcal{D}$ uses at most ℓ bits of randomness assume, without loss of generality, that $\log m \leq \ell$. Then, the circuit C can be implemented as an AC^0 circuit (constant depth) of size $|C| = 2^{O(\ell)}$ by precomputing $\mathcal{D}(r)$ for all $r \in \{0,1\}^\ell$ and then searching for collisions. Thus, there exists a constant c such that ℓ^c-wise independence $1/n$-fools C. Therefore, we set $\mathcal{H}$ to be a family of ℓ^c-wise independent hash functions from Theorem 2.4. Each function $h \in \mathcal{H}$ maps $[n]$ to $\{0,1\}^\ell$ and can be described using $O(\ell^c \cdot \log n)$ bits. In particular, if $\ell = \mathrm{polylog}(n)$ then each hash function can be described using $\mathrm{polylog}(n)$ bits and the difference in the soundness and completeness is at most $1/n$.

4 The general framework

We have seen an interactive protocol for estimating the size of a set while given only oracle access to the set. In this section, building on this protocol, we extend it to get a more general framework for computing arbitrary function quantiles of the set (the precise definition is below). Then, the framework is used to estimate other interesting measures of the graph, such as degree distribution, the local clustering coefficients distribution and more (see Section 5), given that the size of the set has already been established.

Fix a set S of interest. Let $S_{\leq u}$ be the set of all elements in S whose f value is less or equal u. Formally, $S_{\leq u} = \{x \in S \mid f(x) \leq u\}$. Moreover, we given a similar definition for other operations such as greater than, equal and so on. Formally, for any $\circ \in \{\geq, >, =, <, \leq\}$ we define $S_{\circ u}$ to be $S_{\circ u} = \{x \in S \mid f(x) \circ u\}$.

Given a function f, a set S of known size n, and a parameter q we seek to compute the q-th quantile of f which is a value $A_f(q)$ such that:

$$\begin{aligned} \left|S_{\leq A_f(q)}\right| &\geq n \cdot q; \text{ and} \\ \left|S_{\geq A_f(q)}\right| &\geq n \cdot (1-q) \end{aligned}$$

In particular, if $q = 1/2$ then $A_f(q)$ is the median of the set S with respect to the function f. As before, we will not compute $A_f(q)$ exactly (which is impossible with a sublinear-time verifier), but rather give an approximation $\widetilde{A}_f$ which with high probability satisfies

$$\widetilde{A}_f(q) \in [A_f((1-\varepsilon)q), A_f((1+\varepsilon)q)] \ .$$

Note that the approximation above is with respect to the quantile and not the value $A_f(q)$.

Overview. We begin with a high-level overview of the protocol. We divide the elements into three "buckets" (or bins) according to their $f(x)$ values. Each bucket includes all nodes with values in a specific range, where the ranges are $(-\infty, \widetilde{A}_f(q))$, $[\widetilde{A}_f(q), \widetilde{A}_f(q)]$, and $(\widetilde{A}_f(q), \infty)$. The size of each bucket can be communicated using the lower bound protocol (Section 3.1).

Therefore, our goal now is to learn the size of each bucket. Actually, approximate values suffice as well. Here we reduce the problem to our lower bound protocol. To run the cardinality protocol, we need to implement the two oracle queries for the verifier. The membership queries are easy to implement, as we simply check that the element x is in the set, then compute $f(x)$ and see that its value is in the buckets range (given $\widetilde{A}_f(q)$ the ranges are fixed and known to the verifier).

Multiple q values. The framework can be extended to multiple q values. We can reduce the number of queries if we perform the protocol for buckets induced by the ranges intersection together instead of one-by-one.

An interactive protocol for estimating A_f for a set S and a function f

Parameters: ε, q.

1. P $\Rightarrow$ V: The prover sends the values $\widetilde{A}_f(q)$; $\widetilde{n}_{<q}$; $\widetilde{n}_{=q}$; and $\widetilde{n}_{>q}$, where (allegedly) $\widetilde{A}_f(q) = A_f(q)$; $\widetilde{n}_{<q} = \left|S_{<A_f(q)}\right|$; $\widetilde{n}_{=q} = \left|S_{=A_f(q)}\right|$; and $\widetilde{n}_{>q} = \left|S_{>A_f(q)}\right|$.
2. V: Verifier assets that: (a) $\widetilde{n}_{<q}, \widetilde{n}_{=q}, \widetilde{n}_{>q} \in \mathbb{N}$; (b) $n = \widetilde{n}_{<q} + \widetilde{n}_{=q} + \widetilde{n}_{>q}$; (c) $\widetilde{n}_{<q} + \widetilde{n}_{=q} \geq n \cdot q$; and (d) $\widetilde{n}_{=q} + \widetilde{n}_{>q} > n \cdot (1-q)$.
3. P $\iff$ V: The prover and verifier run the lower bound protocol (Figure 1) on the sets $S_{<A_f(q)}$, $S_{=A_f(q)}$, $S_{>A_f(q)}$ with approximation parameter ε.
4. V returns $\widetilde{A}_f(q)$.

Figure 3: A detailed description of our interactive protocol for estimating the quantile q of a function f on the elements of a given set S of known size n.

Theorem 4.1. *The protocol described above (See Figure 3) asserts that*

$$\widetilde{A}_f(q) \in [A_f((1-\varepsilon)q), A_f((1+\varepsilon)q)] \ ,$$

with soundness error $1/3$, *completeness error* $1/3$, *communication complexity* $\widetilde{O}(1/\varepsilon^2)$, *verifier running-time* $\widetilde{O}(1/\varepsilon^2)$, *and prover running-time* $\widetilde{O}(1/\varepsilon^2 \cdot n)$.

Proof of Theorem 4.1. The lower bound protocol (see Figure 1) is invoked three time. To correct for potentially increased error we use a large enough constant c (as specified in Lemma 3.2) such we can union bound over all invocations of the protocol (e.g., $c \approx 3$). Thereby, soundness is guaranteed with probability $3e^{-2c} < 1/3$ (by union bound) and completeness guaranteed with probability $e^{-4c} < 1/3$.

Given the guarantees of the lower bound protocol we can conclude that:

$$\left|S_{\leq \widetilde{A}_f(q)}\right| = \left|S_{<\widetilde{A}_f(q)}\right| + \left|S_{=\widetilde{A}_f(q)}\right| \geq (1-\varepsilon)\left(\widetilde{n}_{<q} + \widetilde{n}_{=q}\right) \geq (1-\varepsilon)n \cdot q \ .$$

This implies that $\widetilde{A}_f(q) \geq A_f((1-\varepsilon)q)$. On the other hand, we have that

$$\left|S_{\geq \widetilde{A}_f(q)}\right| = \left|S_{>\widetilde{A}_f(q)}\right| + \left|S_{=\widetilde{A}_f(q)}\right| \geq (1-\varepsilon)\left(\widetilde{n}_{=q} + \widetilde{n}_{>q}\right) \geq (1-\varepsilon)n \cdot (1-q) \ .$$

This implies that $\widetilde{A}_f(q) \leq A_f((1+\varepsilon)q)$. Together, this concludes the proof. $\square$

Approximating n. This section assumes that the size n of the set has already been established, and is known to the verifier. If this is not the case, then one can run our protocol for estimating the size of the set first, getting an approximation of n and then running the protocols in this section. Note that if one wants to compute many different functions f, then it suffices to estimate n once, and then run these protocols for any f. For this reason, we did not explicitly include running the cardinality estimation protocol in this section but assumed that it was already established. Finally, note that if one has only an approximation of the size n, then the error of the approximation will be added to the error of the protocol in this section. Thus, in order to get a desired error of ε then one should use $\varepsilon/2$ in each protocol.

5 Applications to social graphs

The framework we developed can be used to estimate the size of any set S. Moreover, the suggested framework can be used to approximate the values distribution of any arbitrary function on elements of S. In order to implement these protocols, the verifier needs to have access to the set via the two methods of membership and random sampling. In this section, we show how to implement this access with a *social graph* and hence get a protocol to estimate the size of a social graph and other complexity measures.

Suppose we have a social graph G with vertex set V and edge set E. We are interested in estimating the size of the graph, that is, we let the set S be the set V of vertices. First, notice that implementing membership access is easy, as such a query is included in the interface of a social graph. Given an element x, we can check if $x \in S$ by checking if x is a valid vertex in V using a single query to the graph. The more involved part is sampling a uniformly random vertex in the graph, as this is not part of the interface given by a social graph. However, using random walks on the graph, we can implement sampling a random vertex using a small number of queries to the graph. This strongly relates to the mixing time of the graph, and we elaborate below on how to do this and its cost.

5.1 Generating random samples

Let $G = (V, E)$ be an undirected graph with n vertices, and let d_x be the degree of a node $v_x \in V$. A random walk with r steps on G, denoted by $R = (x_1, x_2, \ldots, x_r)$, is defined as follows: start from an arbitrary node v_{x_1}, then choose a uniformly random neighbor (i.e., x_{i+1} is chosen with probability $\frac{1}{d_{x_i}}$) and repeat this process $r - 1$ times. As r grows to infinity, the probability that the last step of this random walk lands on a specific node v_i, i.e., $\Pr[x_r = i]$, converges to $p_i \triangleq d_i/D$. The vector $\pi = (p_1, p_2, \ldots, p_n)$ is called the stationary distribution of G. We recommend the book [LPW08] and additionally the survey [LLE96] for an excellent overview on random walks and their properties.

The actual number of steps needed to converge to the stationary distribution depends on what is called as *the mixing time* of G. There are several different definitions of mixing time, many of which are known to be equivalent up to constant factors [LPW08]. All definitions take an ε parameter to measure the distance between the stationary and the induced distribution by the random walk. We denote the mixing time of graph G by $\tau_{mix}(\varepsilon)$. We use the following definition:

Definition 5.1. *Let $\vec{p}$ be a distribution over the vertices of the graph G. Let $\pi_r(\vec{p})$ be the distribution of the end point of an r step random walk starting from a vertex chosen in accordance with $\vec{p}$. Then we say that ϵ-mixing time of the graph is $\tau_{mix}(\epsilon)$ if for any $\vec{p}$ we have that the total variation is less than ϵ. Namely,*

$$\left\|\pi_{\tau_{mix}(\epsilon)}(\vec{p}) - \pi\right\|_1 \le \epsilon \ .$$

It is customary to define the mixing time to be $\tau_{mix} := \tau_{mix}(1/4)$. Choosing this (or any other constant) is not very significant as the value of ε affects the value by at most a logarithmic amount [LPW08]:

$$\tau_{mix}(\varepsilon) \le \lceil \log_2 1/\varepsilon \rceil \, \tau_{mix} \ .$$

Social network graphs are known to have low mixing times. Recently, Addario-Berry et al. [ABL12] proved rigorously that the mixing time of Newman-Watts [NW99a; NW99b] small world networks is $\Theta(\log^2 n)$. Mohaisen et al. [MYK10] provide numerical evaluation of the mixing time of several networks. The empirical evidence provided by [MYK10] support the claim that the theoretical argument by Addario-Berry et al. [ABL12] extends to real world social networks. Specifically, in [MYK10, Table 1 and Figure 2] it is shown that to get total variation close to 0, the number of steps should be $r = \log^2 n$ for the Facebook network, $r = 3\log^2 n$ for the DBLP and youtube networks, and $r = 10\log^2 n$ for the Live Journal network.

Sampling a random vertex. There are three popular ways to sample a *node uniformly at random* [CDKLS16] (and matching lower bounds [CH18]). As in this model we do not have any prior knowledge of the graph, we use *rejection sampling.*

In this processes we start by performing a random walk for $r = \tau_{mix}(\epsilon)$ steps. The stationary distribution is proportional to the degree of the vertex, that is $\Pr[x_r = v_i] = d_i/D$. To fix the dependency on the degree, after preforming the random walk, we accept the vertex with probability $1/d_i$ and reject it otherwise. Thus, the expected probability for rejection is $\sum_{i=1}^{n} 1/d_i \cdot d_i/D = n/D = 1/\Delta$, and the expected number of trials until acceptance is Δ. Using this approach to sample random vertices, we get the following theorem.

Theorem 5.2. *Let G be a graph of size n with mixing time τ_{mix} and average degree Δ. For every $\varepsilon > 0$, there is a two-message public-coin interactive protocol to estimate the size of the graph within an error ε, in the graph query model where the verifier's query complexity and communication are bounded by $O(\frac{1}{\varepsilon^2} \cdot \log 1/\varepsilon \cdot \tau_{mix} \cdot \Delta)$ queries, and the prover runs in $\widetilde{O}(n \cdot 1/\varepsilon^2)$ time.*

Sampling a random edge. Sampling an *edge uniformly at random* from the graph can be achieved as follows: (1) generate a random node v_i from the stationary distribution; (2) pick one of v_i's neighbors uniformly at random. The probability for sampling an edge $e = (v_i, v_j)$ is $\frac{d_i}{D} \cdot \frac{1}{d_i} + \frac{d_j}{D} \cdot \frac{1}{d_j} = \frac{1}{|E|}$.

The mixing time of the edges is bounded by the mixing time of the nodes[2]. Note that no rejection is needed here. Thus, an edge is sampled using $O(\tau_{mix})$ queries, without a dependency on the average degree of the graph. We apply the framework on the set of *edges* of the graph and get the following theorem.

Theorem 5.3. *Let G be a graph with m edges and mixing time τ_{mix}. For every $\varepsilon > 0$, there is a two-message public-coin interactive protocol to estimate m within an error ε, in the graph query model where the verifier' query complexity and communication are bounded by $O(\frac{1}{\varepsilon^2} \cdot \log 1/\varepsilon \cdot \tau_{mix})$ queries, and the prover runs in $\widetilde{O}(m \cdot 1/\varepsilon^2)$ time.*

5.2 The average degree

For a graph with n nodes and m edges, the average degree is m/n. The average degree is a crucial property of a social graph ([DKS14]). Estimating the average degree can be done using Theorems 5.2

[2]Since any ε deviation is further divided by the v_i's degree.

and 5.3 when using $\varepsilon' = \varepsilon/4$, where one can obtain the following bounds:

$$\frac{m}{n}(1-2\varepsilon') < \frac{m(1-\varepsilon')}{n(1+\varepsilon')} < \frac{\widetilde{m}}{\widetilde{n}} < \frac{m(1+\varepsilon')}{n(1-\varepsilon')} < \frac{m}{n}(1+4\varepsilon') \ .$$

The number of queries required by the prover for this is then $O(\frac{1}{\varepsilon^2} \cdot \log 1/\varepsilon \cdot \tau_{mix} \cdot \Delta)$.

5.3 Degree distribution

In this subsection, we show how to use our framework to estimate the degree distribution of the graph, given that we have already established the size of the graph. We set the function $f(v) = d(v)$ and the use of the framework (Theorem 4.1) immediately yields the nodes degree distribution. For a parameter b, the framework returns all quantiles $\vec{q} = (1/b, 2/b, \ldots, (b-1)/b)$. This is a very robust surrogate for the full nodes' degree distribution.

Theorem 5.4. *Let G be a graph of size n. Let b be an integer and let $\varepsilon > 0$. Let $A_f(\vec{q}) = (A_f(1/b), A_f(2/b), \ldots, A_f((b-1)/b))$ be the quantiles of the nodes' degree. There is a three-message protocol for estimating $A_f(\vec{q})$ (given n) to within a factor of ε in the graph query model where the verifier performs $\widetilde{O}(\varepsilon^{-2})$ queries to the graph (this also bounds the communication complexity and the verifier's run-time).*

5.4 Local clustering coefficients

Besides the degree distribution, one of the interesting measures for social graph is the distribution of the *local clustering coefficients* [CRTB06; UKBM11]. The local clustering coefficient of a node quantifies how close the sub-graph of the node and its neighbors to being a clique. The notion of social graph composed by small overlapping mini-communities is captured by this node-centric view. In turn, the local clustering coefficients can be used to quantify how close is a graph to a small-world network.

Let N_i be the set of neighbor nodes of the node v_i. The number of edges between any two nodes in N_i is at least 0 and at most $d_i(d_i-1)/2$. The local clustering coefficient C_i measures the fraction between the actual number of edges between nodes in N_i and the maximum number of such edges. Thus, $0 \le C_i \le 1$. Formally,

$$C_i = \frac{|\{(j,k) : (v_j, v_k) \in E, v_k, v_j \in N_i, j \neq k\}|}{d_i(d_i-1)} \ .$$

We set the function $f(v_i) = C_i$ and the use of the framework (Theorem 4.1) immediately yields the local clustering coefficient distribution. The computation of $f(v_i)$ in our model requires $O(d_i^2)$ queries. For simplicity, we assume that we additionally have oracle access to the mutual neighbors of two vertices, or alternatively, can get the full list of neighbors in a single query which reduces the cost to $O(d_i)$ queries. We denote by $d_{\max}$ the maximum degree of any node in the graph.

Theorem 5.5. *Let G be a graph of size n. Let b be an integer and let $\varepsilon > 0$. Let $A_f(\vec{q}) = (A_f(1/b), A_f(2/b), \ldots, A_f((b-1)/b))$ be the quantiles of $\{C_1, C_2, \ldots, C_n\}$. There is a three-message protocol for estimating $A_f(\vec{q})$ (given n) to within a factor of ε in the graph query model where the verifier performs $\widetilde{O}(\varepsilon^{-2} \cdot d_{max})$ queries to the graph (this also bounds the communication complexity and the verifier's run-time).*

5.5 Subpopulations

A very common marketing query asks the number of users with specified properties. For example, the number of 20–30 year-old living in Washington state. In this scenario $f(v) \in \{0, 1\}$. Technically, the framework in Theorem 4.1 can be used as-is. However, since f's range has only two possible values, the framework can be modified to prove a lower bound of the size for each of the preimages.

5.6 Social Graphs and Society

As a result of the growing influence of social networks, the question of the companies obligations and responsibilities is a subject of a heightened discussion. Various companies were required to prove that they are following various laws and guidelines, and to back their claims with data. Our protocol can be applied to some of these claims and we elaborate on two examples that are of particular interest.

Bots and Fake Accounts. Companies, political organizations and other entities are known to use bots to fabricate support, notion of validity or image of popularity. Malicious actors use fake accounts for various attacks on private people or groups of users. In light of public outrage on various events in the recent years, social networks are making efforts to fight this issue.

The detection of fake accounts is the subject of many studies in recent years (see, e.g. [XFH15], [EAKA17]). These studies, together with our protocol, can be used to publicly prove that the proportion of fake users is low, or at least decreasing after certain policies were brought into effect.

For example, in many cases, fake accounts appear in *clusters* that share similar emails, dates of joining the network and other features. One can apply our framework with a function $f(v)$ that returns a "similarity" measure of a vertex v to its neighbors, for some appropriate definition of similarity. In a healthy network we would expect the average similarity to be somewhat high but beneath a certain trivial threshold. We also expect the number of individual vertices with a suspiciously high neighbor-similarity to be low. It is crucial to require social networks both to keep track of these and other "red flags" and *prove* that they are indeed fighting the phenomenon.

Echo Chambers. Social networks are one of the main stages for political debate, compared by many to a virtual "town square" where different people have a chance to debate their opinions. In contrast to this, others claim that instead of introducing various opinions, social networks close people in echo chambers where the only opinions that they are exposed to are similar to theirs.

Extensive research has been done on this subject for various social network (see, e.g., Facebook [QSS16] or Twitter [BJNTB15]). In many cases, it is possible to determine the standing of users on controversial subjects, and explore the connection between various parties. One might make use of use protocol to estimate the number of edges in the graph where each endpoint of the edge has a user with a different stand. This might be useful to track how much diversity are users exposed to in the network.

The above are only two of the many phenomena that social media exhibits. Our protocol allows researchers from various fields to demand reliable data and use it to improve our interaction with social media in the upcoming future.

6 Non-interactive succinct arguments for social graphs

We have described an interactive protocol for estimating the size of a social graph and other complexity measures. Our protocols are AM protocols – they are public-coin and consist of a constant number of rounds. One might ask if this already limited interaction can be further reduced to a completely non-interactive setting where the prover sends a proof and the verifier (probabilistically) decides whether to accepts or reject (an MA protocol). Such a non-interactive protocol is very mush desirable: a social graph provider can publish a proof, once and for all, and any user can later verify the proof on its own, using a small number of queries to the graph. This eliminates the need of the prover to interact with each verifier and to be online on time of verification.

Towards this end, we observe that our protocols can be compiled to non-interactive argument systems in the random oracle model. In such an argument system, proofs of false statements *exist*, but it is *computationally hard* to find them. Here, computation is measured by the number of queries performed to the random oracle. We apply the common approach to eliminate interaction, which is called the Fiat-Shamir transformation or heuristic (first used in [FS86]), that is applicable to any public-coin protocol (this is another reason why we insisted on having a protocols public-coin). In the Fiat-Shamir setting, the parties have access to a random oracle, and the prover is computationally limited: it can only perform a (polynomially) bounded number of queries to the random oracle.

The security of the compiled protocol, in general, is not clear and requires careful analysis [CCHLRRW19; CCRR18; CCHLRR18; KRR17]. However, in our case, since our protocols have only a constant number of rounds (it is either a single round for estimating the size of the graph, or two rounds for the general case), it is easy to argue about its soundness. In general, the compiler uses the random oracle to define the randomness sent by the verifier. Very roughly, on prover message Π, the prover uses the randomness in $\rho(\Pi)$ as the verifier's next message, where ρ is the random oracle (see [Mic00; BCS16] and [NPY18, Section 8.1] for a more precise description). As long as the protocol had negligible soundness, then the compiled protocol will be sound (against cheating prover that can perform at most polynomially many queries to the random oracle). Recall that to achieve soundness $2^{-\lambda}$ it suffices to perform parallel repetition of the protocol $O(\lambda)$, which yields a multiplicative overhead of λ in the communication complexity and query complexity of all protocols. The resulting argument size is simply the communication complexity of the amplified protocol.

The result of this compilation is quite remarkable. A company or social network provider (e.g., facebook, twitter, linkedin, youtube) can provide, in their public report, a proof of the health of its network, in terms of the number of users from different communities and other health measures such as local clustering coefficient, distribution of degrees and so on. The proof is written once in the report without a specific verifier in mind. Then, any individual (a private citizen, shareholders, potential buyer) can look at the report and verify its validity. This might have an effect on the way such provides manage their network, with the aim to more transparent and truthful reports. These reports are also critical for business development issues, choosing between networks for advertisement campaigns or for launching social applications.

Acknowledgments

This work was done (in part) while the second and third authors were visiting the Simons Institute for the Theory of Computing. Eylon Yogev is funded by the ISF grants 484/18, 1789/19, Len Blavatnik and the Blavatnik Foundation, and The Blavatnik Interdisciplinary Cyber Research Center at Tel Aviv University.

References

[ABL12] Louigi Addario-Berry and Tao Lei. "The mixing time of the Newman–Watts small world". In: *SODA*. 2012, pp. 1661–1668.

[ACELP13] Lorenzo Alvisi, Allen Clement, Alessandro Epasto, Silvio Lattanzi, and Alessandro Panconesi. "Sok: The evolution of sybil defense via social networks". In: *2013 ieee symposium on security and privacy*. IEEE. 2013, pp. 382–396.

[ANS09] Yuriy Arbitman, Moni Naor, and Gil Segev. "De-amortized Cuckoo Hashing: Provable Worst-Case Performance and Experimental Results". In: *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*. Ed. by Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas. Vol. 5555. Lecture Notes in Computer Science. Springer, 2009, pp. 107–118.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*. 2016, pp. 31–60.

[BJNTB15] Pablo Barberá, John T Jost, Jonathan Nagler, Joshua A Tucker, and Richard Bonneau. "Tweeting from left to right: Is online political communication more than an echo chamber?" In: *Psychological science* 26.10 (2015), pp. 1531–1542.

[BYG08] Ziv Bar-Yossef and Maxim Gurevich. "Random sampling from a search engine's index". In: *J. ACM* 55.5 (2008).

[BYG09] Ziv Bar-Yossef and Maxim Gurevich. "Estimating the impressionrank of web pages". In: *WWW*. 2009.

[BYG11] Ziv Bar-Yossef and Maxim Gurevich. "Efficient Search Engine Measurements". In: *TWEB* 5.4 (2011).

[Bra10] Mark Braverman. "Polylogarithmic independence fools AC^0 circuits". In: *J. ACM* 57.5 (2010), 28:1–28:10.

[Bre12] Markus Brede. "Networks-An Introduction. Mark E. J. Newman. (2010, Oxford University Press.)" In: *Artificial life* 18 (Feb. 2012), pp. 241–2.

[Bro+06] Andrei Broder et al. "Estimating Corpus Size via Queries". In: Association for Computing Machinery, 2006.

[CCHLRR18] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, and Ron D. Rothblum. "Fiat-Shamir From Simpler Assumptions". In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 1004.

[CCHLRRW19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. "Fiat-Shamir: from practice to theory". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*. 2019, pp. 1082–1090.

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. “Fiat-Shamir and Correlation Intractability from Strong KDM-Secure Encryption”. In: *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I.* 2018, pp. 91–122.

[CDKLS16] 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 2016, Montreal, Canada, April 11 - 15, 2016.* 2016, pp. 471–481.

[CEKLM15] Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. “One Trillion Edges: Graph Processing at Facebook-Scale”. In: *PVLDB* 8.12 (2015), pp. 1804–1815.

[CH18] Flavio Chierichetti and Shahrzad Haddadan. “On the Complexity of Sampling Vertices Uniformly from a Graph”. In: *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic.* 2018, 149:1–149:13.

[CRTB06] Luciano daF. Costa, Francisco A. Rodrigues, Gonzalo Travieso, and P. R. Villas Boas. “Characterization of Complex Networks: A Survey of Measurements”. In: *Advances in Physics* 56.1 (2006), pp. 167–242.

[DKS14] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. “On estimating the average degree”. In: *Proceedings of the 23rd International Conference on World Wide Web.* ACM, 2014, pp. 795–806.

[EAKA17] Buket Erşahin, Özlem Aktaş, Deniz Kılınç, and Ceyhun Akyol. “Twitter fake account detection”. In: *2017 International Conference on Computer Science and Engineering (UBMK).* IEEE. 2017, pp. 388–392.

[EK10] David A. Easley and Jon M. Kleinberg. *Networks, Crowds, and Markets - Reasoning About a Highly Connected World.* Cambridge University Press, 2010.

[EKR04] Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. “Fast approximate probabilistically checkable proofs”. In: *Inf. Comput.* 189.2 (2004), pp. 135–159.

[FS86] Amos Fiat and Adi Shamir. “How to Prove Yourself: Practical Solutions to Identification and Signature Problems”. In: *CRYPTO.* Vol. 263. Springer, 1986, pp. 186–194.

[For87] Lance Fortnow. “The Complexity of Perfect Zero-knowledge”. In: *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing.* STOC ’87. 1987.

[GDFMGM18] Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. “Political discourse on social media: Echo chambers, gatekeepers, and the price of bipartisanship”. In: *Proceedings of the 2018 World Wide Web Conference.* 2018, pp. 913–922.

[GKBM10] Minas Gjoka, Maciej Kurant, Carter T Butts, and Athina Markopoulou. “Walking in Facebook: A Case Study of Unbiased Sampling of OSNs”. In: *Proceedings of IEEE INFOCOM 2010* (2010), pp. 1–9.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. “The Knowledge Complexity of Interactive Proof Systems”. In: *SIAM J. Comput.* 18.1 (1989), pp. 186–208.

[GS89] Shafi Goldwasser and Michael Sipser. “Private Coins versus Public Coins in Interactive Proof Systems”. In: *Advances in Computing Research* 5 (1989), pp. 73–90.

[HPWP10] Johan Håstad, Rafael Pass, Douglas Wikström, and Krzysztof Pietrzak. "An Efficient Parallel Repetition Theorem". In: *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings.* 2010, pp. 1–18.

[HRH09] Stephen James Hardiman, Peter Richmond, and Stefan Hutzler. "Calculating statistics of complex networks through random walks with an application to the on-line social network Bebo". In: *European Physics Journal B* 71.4 (2009), pp. 611–622.

[HS19] Prahladh Harsha and Srikanth Srinivasan. "On polynomial approximations to AC". In: *Random Struct. Algorithms* 54.2 (2019), pp. 289–303.

[KBM12] Maciej Kurant, Carter T. Butts, and Athina Markopoulou. "Graph Size Estimation". In: *CoRR* abs/1210.0460 (2012).

[KH15] Liran Katzir and Stephen J. Hardiman. "Estimating Clustering Coefficients and Size of Social Networks via Random Walk". In: *ACM Trans. Web* 9.4 (Sept. 2015).

[KLSC14] Liran Katzir, Edo Liberty, Oren Somekh, and Ioana A. Cosma. "Estimating Sizes of Social Networks via Biased Sampling". In: *Internet Mathematics* 10.3-4 (2014), pp. 335–359.

[KMV17] Varun Kanade, Frederik Mallmann-Trenn, and Victor Verdugo. "How Large Is Your Graph?" In: *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria.* 2017, 34:1–34:16.

[KRR17] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. "From Obfuscation to the Security of Fiat-Shamir for Proofs". In: *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II.* 2017, pp. 224–251.

[Kle00] Jon M. Kleinberg. "The small-world phenomenon: an algorithmic perspective". In: *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA.* 2000, pp. 163–170.

[LLE96] László Lovász, L. Lov, and Of Erdos. "Random Walks on Graphs: A Survey". In: Jan. 1996, pp. 1–46.

[LPW08] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov Chains and Mixing Times.* American Mathematical Society, 2008.

[MMGDB07] Alan Mislove, Massimiliano Marcon, P. Krishna Gummadi, Peter Druschel, and Bobby Bhattacharjee. "Measurement and analysis of online social networks". In: *Internet Measurement Conference.* 2007.

[MYK10] Abedelaziz Mohaisen, Aaram Yun, and Yongdae Kim. "Measuring the mixing time of social graphs". In: *Internet Measurement Conference.* 2010, pp. 383–389.

[Mic00] Silvio Micali. "Computationally Sound Proofs". In: *SIAM J. Comput.* 30.4 (2000), pp. 1253–1298.

[NPY18] Moni Naor, Merav Parter, and Eylon Yogev. "The Power of Distributed Verifiers in Interactive Proofs". In: *Electronic Colloquium on Computational Complexity (ECCC)* 25 (2018), p. 213.

[NW99a] M.E.J. Newman and DJ Watts. "Renormalization group analysis of the small-world network model". In: *Physics Letters A* 263 (1999), pp. 341–346.

[NW99b] M.E.J. Newman and DJ Watts. "Scaling and percolation in the small-world network model". In: *Physical Review E* 60 (1999), pp. 7332–7342.

[QSS16] Walter Quattrociocchi, Antonio Scala, and Cass R Sunstein. "Echo chambers on Facebook". In: *Available at SSRN 2795110* (2016).

[RT10] Bruno F. Ribeiro and Donald F. Towsley. “Estimating and sampling graphs with multidimensional random walks”. In: *Internet Measurement Conference*. 2010.

[RVW13] Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. “Interactive proofs of proximity: delegating computation in sublinear time”. In: *Symposium on Theory of Computing Conference, STOC’13, Palo Alto, CA, USA, June 1-4, 2013*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM, 2013, pp. 793–802.

[Tal17] Avishay Tal. “Tight Bounds on the Fourier Spectrum of AC0”. In: *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*. 2017, 15:1–15:31.

[UKBM11] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. “The Anatomy of the Facebook Social Graph”. In: *CoRR* abs/1111.4503 (2011).

[Wha] *List of mergers and acquisitions by Facebook*. `https://en.wikipedia.org/wiki/List_of_mergers_and_acquisitions_by_Facebook`. 2019.

[XFH15] Cao Xiao, David Mandell Freeman, and Theodore Hwa. “Detecting clusters of fake accounts in online social networks”. In: *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*. 2015, pp. 91–101.

[YW11] Shaozhi Ye and Shyhtsun Felix Wu. “Estimating the size of online social networks”. In: *IJSCCPS* 1.2 (2011), pp. 160–179.

[ZLAZ11] Jia Zhou, Yanhua Li, Vijay Kumar Adhikari, and Zhi-Li Zhang. “Counting YouTube Videos via Random Prefix Sampling”. In: New York, NY, USA: Association for Computing Machinery, 2011. ISBN: 9781450310130.