Estimating the Number of Induced Subgraphs from Incomplete Data and Neighborhood Queries

Dimitris Fotakis, [1] **Thanasis Pittas,** [2] **Stratis Skoulakis** [3]

[1] National Technical University of Athens, Greece
[2] University of Wisconsin-Madison, WI, USA
[3] Singapore University of Technology and Design, Singapore
fotakis@cs.ntua.gr, pittas@wisc.edu, sskoul@sutd.edu.sg

Abstract

We consider a natural setting where network parameters are estimated from noisy and incomplete information about the network. More specifically, we investigate how we can efficiently estimate the number of small subgraphs (e.g., edges, triangles, etc.) based on full access to one or two noisy and incomplete samples of a large underlying network and on few queries revealing the neighborhood of carefully selected vertices. After specifying a random generator which removes edges from the underlying graph, we present estimators with strong provable performance guarantees, which exploit information from the noisy network samples and query a constant number of the most important vertices for the estimation. Our experimental evaluation shows that, in practice, a single noisy network sample and a couple of hundreds neighborhood queries suffice for accurately estimating the number of triangles in networks with millions of vertices and edges.

1 Introduction

The readily available data for machine learning and network parameter estimation tasks are often incomplete and noisy to the extent that it does not allow for accurate parameter estimation (at least not with reasonable confidence). This can happen for a variety of reasons, e.g., data may be hidden (or massaged) due to privacy or business issues (Chierichetti, Kleinberg, and Oren 2018), data logging may be inaccurate, or the phenomenon represented by the data is only partially observed (see e.g., (Hoogendoorn and Funk 2018, Chapter 3) and (Guptaa and Guptab 2019; Eliassi-Rad, Caceres, and LaRock 2019) for discussion and concrete examples). Additional "clean" data may be available, but at a significant cost (which can be monetary, e.g., (Provost, Melville, and Saar-Tsechansky 2007; Saar-Tsechansky, Melville, and Provost 2009), computational, e.g., (Hoogendoorn and Funk 2018, Chapter 3), or communication, e.g., (Eliassi-Rad, Caceres, and LaRock 2019; Hanneke and Xing 2009; Kim and Leskovec 2011)).

In this work, we consider this very general problem in the specific and important context of network parameter estimation. We show how to efficiently combine (one or two) incomplete and noisy network samples, which are readily available for "free", with few vertex neihgborhood queries, in order to accurately estimate the number of small complete subgraphs of the actual network. Despite the fact that we focus on complete subgraphs (edges, triangles and k-cliques) for simplicity and clarity, our approach can be directly generalized to any fixed small graphlet (in fact, the number of k-cliques is the most difficult to estimate with our approach).

Relation to Previous Work and Motivation. In our network-related setting, (Hanneke and Xing 2009) observed that reliable data collection is among the most difficult challenges in modern network analysis. Hence, (Hanneke and Xing 2009; Kim and Leskovec 2011) introduced the *network completion problem*, where one aims to recover the missing vertices and edges of a partially observed network. They assume that the actual network comes from a known generative model (e.g., the stochastic block model in (Hanneke and Xing 2009) or the Kronecker graphs model in (Kim and Leskovec 2011)) and infer the parameters of the model (and through them, the network's unobserved part) using information from a randomly revealed part of the network.

To avoid the (often not adequately justified) assumption that the actual network comes from a known model, (LaRock et al. 2018; Soundarajan et al. 2015, 2017) considered an orthogonal approach, called *node probing*. They assume query access to the actual network and want to reveal as many vertices and edges of the unobserved network as possible, using a fixed number of queries. The query strategy often follows an adaptive exploration-exploitation pattern and crucially depends on the subset of a node's neighborhood revealed by the queries (see (Eliassi-Rad, Caceres, and LaRock 2019, slide 67) for a summary).

In this work, we take a different approach, bringing together the most interesting and realistic features of network completion and node probing (see also (Bliss, Danforth, and Dodds 2014; Soundarajan et al. 2016) for similar approaches). Similarly to probing, we do not make any assumptions about the network's model, consider an arbitrary *underlying* network and assume query access to it. To keep the model simple, we assume that each query reveals the entire neighborhood of the queried vertex. On the other hand, we want the revealed part of the network to provide enough information, in a statistical sense, about the underlying network. Thus, motivated by network completion, we assume that the revealed part is a random sample of the underlying

network, obtained by a known random process. For technical reasons related to the theoretical analysis, we allow our algorithms to use more than one sample of the underlying network. Moreover, since modern network analysis focuses on parameter estimation (see e.g., (Easley and Kleinberg 2010; Jackson 2008) and the references therein), our goal is to accurately estimate the number of k-cliques and other small induced subgraphs, with one or two incomplete and noisy samples of the underlying network and a small *constant* number of vertex queries.

Network Samples and Neighborhood Queries: Our Model. Formally, we consider an arbitrary fixed undirected *underlying* network (or graph) $G = (V, E)$, with $|V| = n$ nodes (or vertices) and $|E| = m$ edges. A *sample* $G_s = (V, E_s)$ is a spanning subgraph of G, with vertex set V and edge set E_s generated by the following random process: every vertex, independently, becomes *hidden*, with probability p, and *visible*, otherwise. An edge e is present in E_s , if at least one of e's endpoints is visible. Formally, $E_s = \{\{u, v\} \in E \mid \text{at least one of } u \text{ or } v \text{ is visible}\}$. The probability p that a node is hidden is the same for all nodes and known to the algorithm (p could be easily estimated, anyway). We let $\mathcal{P}_{G,p}$ denote the distribution over samples from G defined by the random process above. Our algorithms take one or two independent samples from $\mathcal{P}_{G,p}$ and have query access to the neighborhood $N(v) = \{u \in V \mid \{v, u\} \in E\}$ of any node $v \in V$ in the underlying network G. Note that only neighborhood queries to hidden nodes reveal edges not present in G_s, but the algorithm is not aware of which nodes are visible or hidden.

We focus on estimating the number of small complete subgraphs (edges, triangles and k-vertex cliques, or simply k-cliques) of the underlying network G. An (ϵ, δ)-estimator of the number $\rho_k(G)$ of k-cliques computes an estimate $\hat{\rho}_k(G)$ such that $|\rho_k(G) - \hat{\rho}_k(G)| \leq \epsilon\rho_k(G)$, with probability at least $1-\delta$. When the above guarantee is too restricting, a more relaxed version of it is often used in the subgraph counting literature, where the error can be ϵ-multiplicative in a quantity greater than $\rho_k(G)$ (Theorem 4 uses an (ϵ, δ)-estimator of that type). Our algorithms are deterministic, so the probability is over the random sample(s) of the underlying network. We aim at algorithms that for any underlying graph G and $\epsilon, \delta > 0$, compute an (ϵ, δ)-estimation of $\rho_k(G)$ with at most two samples of G and a polynomial, in $1/\epsilon$ and $1/\delta$, number of queries. Note that this means that the underlying graph is assumed to be independent of the accuracy parameters (e.g. m cannot depend on ϵ).

Notation. We let $d(v)$ (or $d_G(v)$) be the degree of a vertex v (in network G). We let $\mathbb{P}[A]$ be the probability of an event A, and $\mathbb{E}[X]$, $\mathrm{Var}[X]$ the expectation and variance of a random variable X.

Our Contributions: Results and Techniques

On the conceptual side, we introduce a new model of network parameter estimation from incomplete samples and node neighborhood queries. Our model naturally interpolates between network completion and node probing, allows for a wide range of ideas from property testing, statistical estimation and online learning to be applied, and provides a solid framework for theoretical and experimental evaluation of algorithms, based on accuracy, number of queries and number of samples used.

On the technical side, we demonstrate the applicability of the new model with a collection of results about the number of queries required for an (ϵ, δ)-estimation of the number of edges, triangles and k-cliques in the underlying network. Estimating the number of triangles and k-cliques is a task important, general and complex enough to demonstrate the applicability of our approach. For ease of exposition, we first present our methods for the simplest case of estimating the number of edges.

As a warm-up, we focus on the number of edges m in the underlying network and show that its efficient estimation requires combining network samples with neighborhood queries. First, to differentiate our work from a long line of research in property testing, we explain why a naïve query-only based estimation of m requires up to n^ϵ queries in the worst-case, for any $\epsilon < 1/2$. E.g., consider a lollipop graph with a clique of $n^{1-\epsilon}$ nodes and a path of length $n - n^{1-\epsilon}$ attached to it. If we try to estimate m by querying nodes selected uniformly at random (nothing else can be done, because the graph is completely unknown), we need n^ϵ queries in expectation, before the first clique node appears in the sample. Until then, the only reasonable estimation for m is $\Theta(n)$, instead of the real value $\Theta(n^{2(1-\epsilon)})$. We should highlight that query-only based estimators can provide non-trivial efficiency guarantees only if the underlying network is sufficiently dense (i.e., $m = \Theta(n^2)$), or in general, if the value of the estimated parameter is sufficiently large. So, previous work in property testing (see e.g., (Borgs et al. 2006; Alon 2002; Alon and Shapira 2005; Goldreich, Goldwasser, and Ron 1998; Goldreich 2017)) focuses on (ϵ, δ)-estimators with $O(\mathrm{poly}(1/\epsilon))$ queries for normalized values of important network parameters, i.e., edges$/n^2$, k-cliques$/n^k$, max-cut$/n^2$, etc.

Next, we discuss naïve estimators based on network samples only. In a network sample $G_1 = (V, E_1)$, each edge is removed with probability p^2 and $\mathbb{E}[E_1] = m(1 - p^2)$. Thus, $|E_1|/(1-p^2)$ is an unbiased estimator of m. We can generalize this idea to $N \geq 1$ network samples $G_1, \ldots, G_N$. Then, we can compute the union $\hat{E}$ of their edge sets and estimate $\hat{m} = |\hat{E}|/(1 - p^{2N})$. Intuitively, increasing the number of samples corresponds to decreasing p, as now each edge is hidden in all samples with probability p^{2N}. We can prove:

Theorem 1. *Given* $N = \Theta\left(\log_{1/p} \frac{1}{\epsilon^2\delta(1-p)}\right)$ *independent samples* $G_1(V, E_1), \ldots, G_N(V, E_N)$ *of the underlying network* $G(V, E)$, $|\cup_{i=1}^{N} E_i|/(1 - p^{2N})$ *is an* (ϵ, δ)*-estimation of the number of edges* $|E|$.

Interestingly, we can show that without neighborhood queries, $\Omega(\log(1/\epsilon))$ samples are also necessary for an (ϵ, δ)-estimation of the number of edges, no matter the estimator that we use.

Theorem 2. *For all $N = o(\log(1/\epsilon))$ and any estimator $\hat{m}$ using N network samples, there exists an underlying network G with m edges, such that*

$$\mathbb{P}_{G_1,\ldots,G_N \sim \mathcal{P}_{G,p}} [|\hat{m}(G_1,\ldots,G_N) - m| > \epsilon m] \geq 1/3\,.$$

Using many network samples is impractical and expensive (and often infeasible). In most applications (e.g., social networks, biological networks), only few (and usually just one) samples are available. So, we next use two network samples and few neighborhood queries to compute an (ϵ, δ)-estimation of the number of edges, triangles and k-cliques of the underlying network.

In the proof of Theorem 1, the estimation error for one sample is determined by $\sum_v d_G^2(v)/m^2$, which corresponds to the variance of G's degree distribution. Then, the estimator reduces the error through access to many network samples. If the number of samples is fixed, we can reduce the error by accessing the neighborhood of the highest degree nodes in G (or in its sample, since G is not directly available), which contribute most to $\sum_v d_G^2(v)$. This is the intuition behind Algorithm 1, where the estimator is the sum of the edges revealed by the queries plus the naïve estimator for the edges in G_2 not incident to any of the queried nodes. The following is the main result of Section 2:

Theorem 3. *If the network G has $m \geq n \log n$ edges, Algorithm 1 with $q = \Theta(p^2/(\epsilon^2\delta(1-p)^5))$ neighborhood queries computes an (ϵ, δ)-estimation $\hat{m}$ of m.*

Algorithm 1 estimates the number of edges of the underlying network G using a constant number of queries, not depending on n or m. So, we can estimate within 1% the number of edges of a huge (and possibly highly irregular) network, for which 30% - 40% of the edges may be missing from the union of the two samples, with few hundreds of neighborhood queries (see experiments of Section 4)!

The intuition behind the proof of Theorem 3 is that if the error of the naïve estimator is large, a relatively small number of high degree vertices have large contribution to $\sum_v d_G^2(v)/m^2$. Since their degree is significantly higher than average, we can identify a large fraction of them from the sample G_1. Then, querying their neighborhoods reduces quickly the estimation error to an acceptable level.

The reason that Algorithm 1 needs to use two network samples is purely technical. Suppose that we use a single network sample G_1 for both determining the query set Q and estimating the number of G's edges not incident to Q. Then, since the selection of Q is strongly biased, conditional on the event that a node $v \notin Q$, the probability that v is hidden is no longer p (e.g., for a large regular network, the query set would almost certainly consist of visible nodes only). Therefore, scaling up the number of edges not incident to Q by $1/(1-p^2)$ is not an unbiased estimator anymore. In our experiments however, this subtle technical issue does not seem to make any difference. Obtaining strong theoretical guarantees for an estimator with a single sample (which may need to deviate from Algorithm 1, see e.g., the one-sample algorithm in Section 4) is an intriguing open question. We should also mention that in Theorem 3, the condition that the underlying network G has $m \geq n \log n$ edges can be replaced by the milder condition that $\epsilon^2\delta = \Omega(\log^2(n)/n)$.

In Section 3, we generalize the ideas above and present Algorithm 2, which computes an estimation of the number of k-cliques using two network samples and $q = \frac{k^3 2^{\Theta(k)}}{\epsilon^2\delta(1-p)^{\Theta(k)}}$ neighborhood queries (Theorem 4). As a corollary, we get an estimator for the number of triangles with $q = \Theta(1/(\epsilon^2\delta(1-p)^{12}))$ queries. The main idea and the proof outline are similar to those in Algorithm 1 and Theorem 3. The key technical step is again to show that a significant improvement on the estimation error can be obtained by querying the highest degree nodes in the first sample. However, using the second sample to estimate the number of k-cliques not included in the subgraph induced by the query set requires more care and technical work than the correction used in Algorithm 1.

Our approach is general and not restricted to k-cliques. By appropriately modifying the weights in lines 10 and 13 of Algorithm 2, we can estimate the number of any fixed k-vertex graphlet in the underlying network. The number of neighborhood queries is upper bounded by those for k-cliques, in Theorem 4, since the variance of the naïve sample-only-based estimator is maximized for k-cliques (intuitively, estimating the number of k-cliques comprises the most difficult case for our approach).

In Section 4, we experimentally evaluate the performance of our algorithms on synthetic and real-world networks and compare them against other estimation methods. We mostly focus on triangle estimation by a variant of Algorithm 2 that uses only one network sample. The main message is that the number of triangles can be estimated within $0.29\% - 1.66\%$, with one network sample and at most 200 neighborhood queries. Accuracy does not depend on the size of the network, the actual number of triangles, the value of p (we use $p \in [0.15, 0.85]$), and the variance of the degree distribution. The estimation accuracy of the one-sample variant of Algorithm 2 is very close to that of the optimal query set, decided with full knowledge of the underlying network. We also investigate robustness of our approach to different sample generation processes that introduce stronger dependencies among the hidden edges. We observe similar accuracy levels when every node v hides its edges independently with unknown probability p_v and when visible edges are determined by a random walk with jumps.

Other Related Work

Our sample generation model was inspired by the *public-private* model of social networks (Chierichetti et al. 2015; Epasto, Esfandiari, and Mirrokni 2019), with the true network fixed, but unknown, and the sample as the conceptual analogue of the public part (the analogy, however, stops here, since the public part in (Chierichetti et al. 2015) is fixed, while our samples are random.). The most interesting case is when the private part is a subset of a node's neighborhood.

The *network completion* problem with hidden edges has received considerable attention (see e.g., (Clauset, Moore, and Newman 2008; Goldberg and Roth 2003; Hanneke and

Xing 2009; Kim and Leskovec 2011), and also (Chen, Mira, and Onnela 2019) where they consider time-evolving networks). Similar network augmentation problems have been studied in social networks (see e.g., (Koskinen et al. 2013) and the references therein). Our model and approach bear a resemblance to exploring a partially visible network through node probing (Soundarajan et al. 2015, 2017; LaRock et al. 2018; Morales, Caceres, and Eliassi-Rad 2019). However, research in this direction does not assume anything about the visible part of the network, considers different query models, and uses online learning approaches, following an exploration-exploitation pattern. We refer to the recent tutorials by (Eliassi-Rad, Soundarajan, and Bhadra 2018; Eliassi-Rad, Caceres, and LaRock 2019) for a survey.

Among a large volume of research on dealing with incomplete networks, the approach of (Soundarajan et al. 2016; Bliss, Danforth, and Dodds 2014) is closest to ours. Similarly to our model, they consider an arbitrary underlying network and network samples generated by simple random processes. They focus on estimating simple network parameters, such as the number of vertices and edges, the degree distribution or the clustering coefficient, and naturally consider querying nodes with large degree. However, they adopt a purely heuristic viewpoint and do not provide any theoretical guarantees on the performance of their algorithms. In contrast to (Soundarajan et al. 2016; Bliss, Danforth, and Dodds 2014), we adopt a structured sample generation model, prove strong theoretical performance guarantees about the number of queries required for a given accuracy, and consider the more demanding task of estimating the number of k-cliques.

Our work is fundamentally different from (and virtually impossible to fairly compared against) previous work on estimation of graphlet counts in large-scale networks through random node or edge sampling (see e.g., (Ahmed et al. 2014; Ahmed, Willke, and Rossi 2016; Kolda et al. 2014; Lakhotia et al. 2019; Rossi, Zhou, and Ahmed 2018; Seshadhri, Pinar, and Kolda 2013; Jha, Seshadhri, and Pinar 2015)) and on sublinear or streaming algorithms for k-clique counting (see e.g., (Eden et al. 2015; Eden, Ron, and Seshadhri 2018, 2020; Gonen, Ron, and Shavitt 2011; Tsourakakis, Kolountzakis, and Miller 2011)). Our algorithms have access to network samples, which are assumed to be readily available for "free". Therefore, in experimental evaluation, our algorithms need about 200 neighborhood queries to estimate the number of triangles within 1% in networks with millions of nodes, whilst previous work, without access to network samples, requires about 1000 times more queries for estimations of similar accuracy (see Figure 2(a), and e.g., (Rossi, Zhou, and Ahmed 2018, Fig. 3) and (Ahmed et al. 2014, Table 3)). Moreover, we prove that the number of k-cliques can be estimated with a number of neighborhood queries not depending on the order or the size of the network, even for sparse networks. In contrast, e.g., (Eden, Ron, and Seshadhri 2018) requires $O(\frac{n}{C_k^{1/k}} + \frac{m^{k/2}}{C_k})\mathrm{poly}(\log n, 1/\epsilon, k)$ queries, where C_k is the number of k-cliques, which becomes comparable to our query complexity only if the network is dense.

Algorithm 1 Edge Estimator using Two Network Samples and q Neighborhood Queries.

Input: Samples G_1, G_2 of the underlying network G and the number of queries q.
Output: Estimation $\hat{m}$ of the number of edges of G.

1: Let Q be the set of the q highest degree vertices of G_1.
2: Query each $v \in Q$ for its neighborhood $N_G(v)$.
3: $m_q \leftarrow |\{\{u, v\} \in E(G) \mid u \in Q \vee v \in Q\}|$
4: $m_{rest} \leftarrow |E(G_2 \setminus Q)|/(1-p^2)$
5: $\hat{m} \leftarrow m_q + m_{rest}$
6: **return** $\hat{m}$.

2 Estimating the Number of Edges with Neighborhood Queries

We proceed to analyze Algorithm 1, which estimates the number of edges using only two network samples, G_1 and G_2, and a number of vertex neighborhood queries. Next, we outline the proof of Theorem 3.

Since every query essentially removes a term from the error $\sum_{u \in V} d_G^2(u)/m^2$ of the one-sample estimator, we want to query a set of vertices with large degree in the underlying network G. We next show that Algorithm 1 computes an unbiased estimation of the number of edges and upper bound its variance. The notation $Q(G_1)$ below means to highlight that the query set Q is a random variable, whose value is determined by the random sample G_1.

Lemma 1. *The estimation $\hat{m}$ computed by Algorithm 1 is unbiased, i.e., $\mathbb{E}_{G_1, G_2 \sim \mathcal{P}_{G,p}}[\hat{m}] = m$, and*

$$\operatorname*{Var}_{G_1, G_2 \sim \mathcal{P}_{G,p}}[\hat{m}] \leq \frac{p^2}{2(1-p^2)} \operatorname*{\mathbb{E}}_{G_1 \sim \mathcal{P}_{G,p}} \left[\sum_{u \in V \setminus Q(G_1)} d_G^2(u) \right]$$

The key step in the proof of Theorem 3 is to upper bound $\mathbb{E}\left[\sum_{u \in V \setminus Q(G_1)} d_G^2(u)\right]$, which quantifies the contribution of the vertices not in the query set to the estimation error of Algorithm 1. Then, the proof of Theorem 3 follows easily from Lemma 1, Lemma 2 and Chebyshev's inequality.

Lemma 2. *Let $G = (V, E)$ be an underlying network with $m \geq n \log n$ edges and let $G_1 \sim \mathcal{P}_{G,p}$ be a sample of G. Then, for any $\delta, \epsilon > 0$, Algorithm 1 with $q = 1/(\delta\epsilon^2)$ queries has*

$$\operatorname*{\mathbb{E}}_{G_1 \sim \mathcal{P}_{G,p}} \left[\sum_{u \in V \setminus Q(G_1)} d_G^2(u) \right] \leq \frac{4096}{(1-p)^4} \delta\epsilon^2 m^2 \,,$$

where $Q(G_1)$ is the set of the q highest degree vertices in G_1 queried by Algorithm 1.

We next outline the proof of Lemma 2. The main idea is to compare the query set of Algorithm 1 with an optimal query set, consisting of the q highest degree vertices in the underlying graph G. Specifically, we first show that revealing the neighborhood of the $q = 1/(\delta\epsilon^2)$ highest degree vertices of the underlying network G indeed brings the estimation error to the desired level.

Algorithm 2 Estimator of the Number of k-Cliques using Two Samples and q Queries.

Input: Samples G_1, G_2 of the underlying network G and the number of queries q.
Output: Estimation $\hat{\rho}_k$ of the number of k-cliques in G.

Let Q be the set of the q highest degree vertices in G_1 .
Query each $v \in Q$ for its neighborhood $N_G(v)$.
Set $G' \leftarrow G_2 \cup \{\{u, v\} \in E(G) \mid v \in Q \wedge u \in N_G(v)\}$ and $\hat{\rho}_k \leftarrow 0$
for each k-clique C of G' incident to Q **do**
    $i \leftarrow$ number of vertices of C outside Q
    if $i \leq 1$ **then** $\hat{\rho}_k \leftarrow \hat{\rho}_k + 1$
    else $\hat{\rho}_k \leftarrow \hat{\rho}_k + (ip(1-p)^{i-1} + (1-p)^i)^{-1}$
end for
$\hat{\rho}_k \leftarrow \hat{\rho}_k + \frac{\rho_k(G_2 \setminus Q)}{kp(1-p)^{k-1}+(1-p)^k}$
return $\hat{\rho}_k$

Lemma 3. *For any fixed $\delta, \epsilon > 0$, let Q^* be the set of the $q = 1/(\delta\epsilon^2)$ highest degree vertices in the underlying graph G. Then, $\sum_{u \in V \setminus Q^*} d_G^2(u) \leq \delta\epsilon^2 m^2$.*

As a last step, we argue that selecting the q highest degree vertices in the sample network G_1 leads to a similar decrease in the estimation error, because either G is almost regular, in which case querying any set of q vertices is almost equally good, or G is far from being regular, in which case the highest degree vertices can be easily identified from the sample G_1. The latter holds, because the degree of every vertex u in G_1 is either equal to its degree in G, if u is visible, or close to the scaled version $(1-p)d_G(u)$, if u is hidden. The following shows that selecting the q highest degree vertices in G_1, after such a scaling, is not so much different from selecting the q highest degree vertices in G.

Lemma 4 (Re-sorting Lemma)**.** *Let $d_1, d_2, \ldots, d_n \in \mathbb{N}$ such that $d_1 \geq d_2 \geq \cdots \geq d_n$, and let $c_1, c_2, \ldots, c_n \in (0, 1]$. Let $\hat{S}$ be the index set of the $n - q$ smallest products in $\{c_i \cdot d_i\}_{i=1}^n$. For any $a \in (0, 1]$ such that $a \leq c_i$, for all $i = 1, \ldots, q$, $\sum_{i \in \hat{S}} d_i \leq \frac{1}{a} \sum_{i=q+1}^n d_i$.*

3 Estimating the Number of k-Cliques

In this section, we generalize the ideas of Section 2 and estimate the number of k-cliques in the underlying graph G with two samples and $q = \frac{k^3 2^{\Theta(k)}}{\epsilon^2 \delta (1-p)^{\Theta(k)}}$ neighborhood queries. At the conceptual level, the approach is similar to that of Algorithm 1. However, we need to take care of several additional technical details not evident when we estimate the number of edges. The following is the main result of this section.

Theorem 4. *Let G be an underlying network with at least $n \log^{k-1} n$ induced stars with $k-1$ leaves. Then, Algorithm 2 with $q = \Theta\Big(\frac{k^3 (16(2k-1))^{2(k-1)}}{(1-p)^{6(k-1)} \epsilon^2 \delta}\Big)$ neighborhood queries computes an estimation $\hat{\rho}_k$ of the number $\rho_k(G)$ of k-cliques in G such that*

$$\mathop{\mathbb{P}}_{G_1, G_2 \sim \mathcal{P}_{G,p}} \Big[|\hat{\rho}_k - \rho_k(G)| > \epsilon s_{k-1}(G) \Big] \leq \delta \,,$$

where $s_{k-1}(G) = \sum_{v \in V} \binom{d(v)}{k-1}$ is the number of induced stars with $k-1$ leaves in the underlying network G.

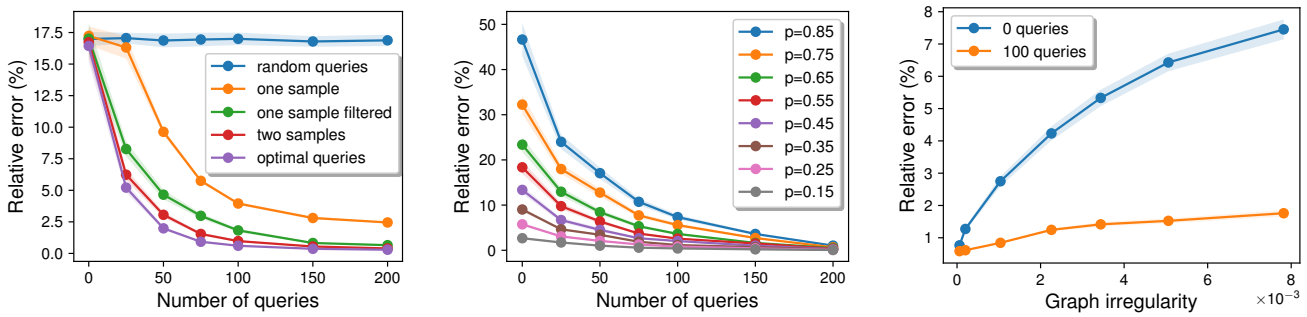
Theorem 4 provides a slightly weaker guarantee than that of Theorem 3, since the estimation error can be up to $\epsilon s_{k-1}(G)$, instead of $\epsilon \rho_k(G)$. For some intuition about that, consider two neighboring vertices u and v that have a large number of neighbors in common. Hence, u and v belong to a large number of triangles that all disappear as soon as both u and v turn hidden. Hence our analysis should consider all stars with $k-1$ leaves in G, which lead to the weaker guarantee of Theorem 4. This is also the guarantee used in closely related previous work (Seshadhri, Pinar, and Kolda 2013; Jha, Seshadhri, and Pinar 2015; Kolda et al. 2014) with the justification that in real world networks, s_{k-1} is typically close to the number of k-cliques because of their asymptotically constant clustering coefficient.

4 Experimental Evaluation

We provide an experimental evaluation of our algorithms on synthetic and real-world networks. Our algorithms were implemented in Python, using SNAP (Leskovec and Krevl 2014), and run on a laptop with a 2.7 GHz Intel Core i7-7500U processor and 8GB RAM. We ignored edge directions and loops, where applicable.

Methodology. Focusing on triangle estimation, we compare different query strategies and variants of Algorithm 2. Having a fixed underlying graph, we generate a sample graph and calculate the absolute relative error of the algorithm's estimation. We repeat this process 100 times, each with a new independently generated network sample, and take the average of all the relative errors, which we plot on the y-axis. We do so for Algorithm 2 and its one-sample variant described below, as well as for random query sets and for the strategy where the highest degree vertices of the actual underlying network are queried (which, of course, requires full knowledge of the underlying network).

One-Sample Algorithm. We mostly evaluate a variant of Algorithm 2 that uses the same network sample for both determining the query set and for the computation of the output value. As discussed in Section 1, after Theorem 3, using the same sample for both tasks introduces bias to the estimator, because conditioned on the event that a vertex v is not selected in the query set, the probability that v is hidden is no longer p. In our one-sample algorithm we use the following heuristic approach, which mitigates the bias: after learning the neighborhood in the underlying network of the vertices in the query set, we order them in non-increasing order of their degrees in the underlying network and keep only the top α-percentile of them in the query set. We remove the remaining $(1-\alpha)$ fraction from the query set and completely ignore the responses to the corresponding queries. Intuitively, this removes some dependency between the query set and the network sample. Thus the probability that each vertex is hidden gets closer to p, as desired. In our experiments, we set $\alpha = 1/2$, without any fine-tuning, to demonstrate that a simple parameter selection works well.



(a) (b) (c)

Figure 1: (a) Comparison of different triangle estimation methods and query strategies on as-caida20071105 dataset with $p = 0.6$. (b) Triangle estimation with one network sample on ego-gplus network for different values of the parameter p. (c) How the irregularity of the underlying network affects the estimation's accuracy. We use $\sum_u d_G^2(u)/m^2$ to quantify irregularity of the underlying network G. For this experiment, we set $p = 0.85$ and use the estimator of Algorithm 1. Shading around the curves is proportional to standard deviation.

Figure 1(a) shows the comparison of these methods on as-caida20071105 dataset (results on the other datasets are similar, see Table 1). The one-sample triangle estimator with the filtering step above becomes comparable to the two-sample estimator which, in its turn, is very close to the estimator querying the truly highest-degree vertices. Random queries do not noticeably improve the estimation as the vertices that crucially affect the variance term are too few to be selected by a random strategy.

Dependence on p. We study how the value of p affects the estimation's accuracy. Figure 1(b) shows different curves, each one representing the one-sample triangle estimator's error for a different value of $p \in [0.15, 0.85]$. Interestingly, the error drops very close to 0 after about 200 queries, even when 85% of the edges are hidden.

The Role of Regularity. While the naïve estimator may be accurate enough for some networks, it may completely fail for others. For the case of edges, this is because its error involves the irregularity measure $\sum_{v \in V} d^2(v)/m^2$. In Figure 1(c), we test our edge estimator on synthetic power-law graphs with different exponents $\gamma \in [2, 5]$. We find that when the input graph is highly regular (i.e. $\gamma = 5$), the naïve estimator is accurate and the queries do not make any significant improvement. However, when the input graph is irregular (i.e. $\gamma = 2$), the naïve estimator has much higher error, but only few queries suffice to dramatically reduce it.

Evaluation on Different Datasets. We evaluate our one-sample estimator on real-world and synthetic networks of different types and sizes. The results are summarized in Table 1. Edge estimation is less demanding and 100 queries suffice for reducing the error below 1.3% in all cases. Another observation is that real-world networks are very different regarding their irregularity measure $\sum_v d_G^2(v)/m^2$. Thus, the naïve estimator performs well for some of them (e.g., roadNet-PA), while for others (e.g., wiki_talk_ar), its error is large, but always improves quickly after some neighborhood queries. Interestingly, the same number of queries suffices to reduce the error for networks with very different size and degree distribution. This allows for an accurate estimation after a small number of queries, even if the underlying network has millions of vertices (e.g., for wiki_talk_ar, the query set includes less than 0.02% of the vertices).

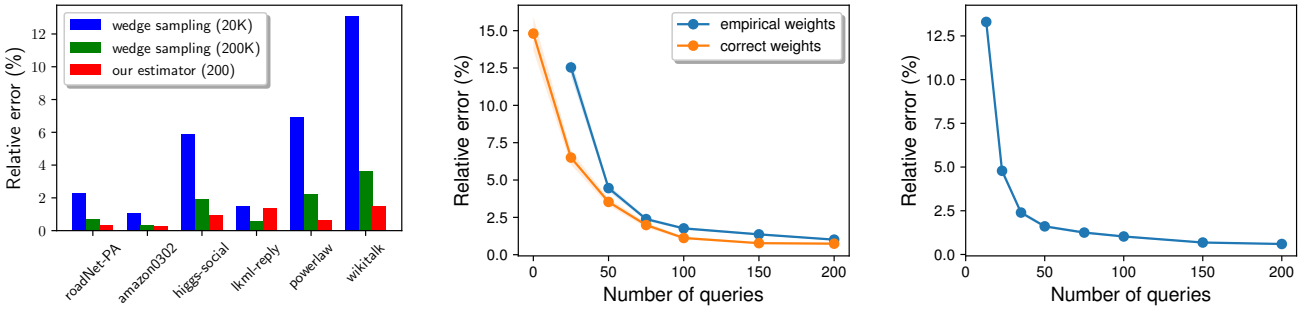
Comparison to Existing Methods. Ignoring the sample graph and using only a small number of queries (i.e., up to 10% of the vertices) cannot lead to better accuracy in practice. The query complexity of sublinear algorithms (Eden et al. 2015) for triangle counting increases with the network size, while our algorithm uses a fixed number of queries. A more practical method is wedge sampling (Seshadhri, Pinar, and Kolda 2013), which estimates the number of triangles by querying wedges uniformly at random. In contrast to ours, wedge sampling requires that vertex degrees are known and are used to compute a uniform distribution over the wedges. As Figure 2(a) shows, even with this additional assumption, wedge-sampling requires more than $2 \cdot 10^5$ queries for an error rate similar to that of our estimator with only 200 queries.

Robustness to Sample Generation. We also evaluate the robustness of our approach to different sample generation models and depict the results for as-caida20071105 in Figure 2(b) and 2(c). To adjust the estimator, we change the weights used in lines 10 and 13 of Algorithm 2: in line 10, we divide with the probability that an edge is visible, and in line 13, with the probability that a triangle is visible in the sample. In fact, we let the estimator learn these probabilities from the revealed part of the network (i.e., use the empirical fractions of visible edges and triangles).

Different Probability p_v. Initially, each node $v \in V$ gets a probability p_v chosen uniformly from $[0, 0.7]$. Then, v hides independently, with probability p_v, each of its inci-

Graph	n	m	t	$\sum_u d_u^2/m^2$	$e_{\hat{m}(0)}$	$e_{\hat{m}(100)}$	$e_{\hat{t}(0)}$	$e_{\hat{t}(100)}$	$e_{\hat{t}(200)}$
roadNet-PA	1088K	1541K	67K	$4 \cdot 10^{-6}$	0.06	0.06	0.35	0.34	0.34
amazon0302	262K	899K	717K	$2 \cdot 10^{-5}$	0.13	0.13	0.36	0.31	0.29
higgs-social	456K	12508K	83023K	0.0003	0.47	0.22	1.71	1.06	0.96
lkml-reply	27K	165K	1893K	0.0039	2.08	1.29	7.24	2.93	1.50
powerlaw($\gamma = 2.1$)	100K	317K	1058K	0.0057	2.49	0.43	9.08	1.70	0.65
as-caida20071105	26K	53K	36K	0.0105	2.51	0.53	14.10	1.79	0.50
ego-gplus	23K	39K	18K	0.0192	3.63	0.82	15.23	2.04	0.35
oregon1_010526	11K	23K	19K	0.0227	3.9	0.61	17.63	1.76	0.38
web-frwikinews	25K	68K	23K	0.0628	6.27	0.17	18.1	0.72	0.29
wiki_talk_fr	1420K	2330K	5849K	0.2298	16.83	0.89	9.16	1.96	1.66
wiki_talk_ar	1095K	1549K	679K	0.3669	19.9	0.31	14.2	2.37	1.37

Table 1: Evaluation of one-sample edge and triangle estimators on real-world and synthetic networks. m is the number of edges and t is the number of triangles. The last columns depict the estimation %-error when 0, 100 (and 200 for triangles) queries are used. The last digits are uncertain due to the variance of the measurements. Network samples are generated with $p = 1/2$.



(a) Comparison to wedge sampling

(b) Sample with uniform p_v .

(c) Sample generated by random walk with jumps.

Figure 2: Comparison to existing methods and evaluation on different noise models.

dent edges. An edge is visible if none of its endpoints hides it. Figure 2(b) shows the performance of our adjusted one-sample triangle estimator. As can be seen there, using the weights estimated by the algorithm has no measurable impact on accuracy.

Random Walk with Jumps. Starting from a random node, in each step, either we continue at a random neighbor of the current node, or with probability 0.15, we restart from a uniform random node (as in (Leskovec and Faloutsos 2006)). We stop when 35% of the edges are visited and put them in the sample. Figure 2(c) shows that the estimation error decreases quickly for this sample generation model as well.

Acknowledgments

Dimitris Fotakis is supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "First Call for H.F.R.I. Research Projects to support Faculty members and Researchers' and the procurement of high-cost research equipment grant", project BALSAM, HFRI-FM17-1424. This research was carried out while Thanasis Pittas was an undergraduate student at National Technical University of Athens. He is currently supported by UW-Madison CS Departmental Research Fellowship. Stratis Skoulakis is supported by NRF 2018 Fellowship NRF-NRFF2018-07.

References

Ahmed, N. K.; Duffield, N.; Neville, J.; and Kompella, R. 2014. Graph sample and hold: A framework for big-graph analytics. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1446–1455.

Ahmed, N. K.; Willke, T. L.; and Rossi, R. A. 2016. Estimation of local subgraph counts. In *2016 IEEE International Conference on Big Data (Big Data)*, 586–595. IEEE.

Alon, N. 2002. Testing subgraphs in large graphs. *Random Struct. Algorithms* 21(3-4): 359–370.

Alon, N.; and Shapira, A. 2005. Linear Equations, Arithmetic Progressions and Hypergraph Property Testing. *Theory of Computing* 1(1): 177–216.

Bliss, C.; Danforth, C.; and Dodds, P. 2014. Estimation

of Global Network Statistics from Incomplete Data. *PLoS ONE* 9. URL https://doi.org/10.1371/journal.pone.010847.

Borgs, C.; Chayes, J. T.; Lovász, L.; Sós, V. T.; Szegedy, B.; and Vesztergombi, K. 2006. Graph limits and parameter testing. In *Proc. of the 38th ACM Symposium on Theory of Computing (STOC 2006)*, 261–270. ACM.

Chen, S.; Mira, A.; and Onnela, J. 2019. Flexible Model Selection for Mechanistic Network Models. *Journal of Complex Networks* .

Chierichetti, F.; Epasto, A.; Kumar, R.; Lattanzi, S.; and Mirrokni, V. 2015. Efficient Algorithms for Public-Private Social Networks. In *Proc. of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2015)*, 139–148. ACM.

Chierichetti, F.; Kleinberg, J. M.; and Oren, S. 2018. On discrete preferences and coordination. *Journal of Computer and System Sciences* 93: 11–29.

Clauset, A.; Moore, C.; and Newman, M. 2008. Hierarchical structure and the prediction of missing links in networks. *Nature* 453: 98–101.

Easley, D.; and Kleinberg, J. 2010. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge, MA, USA: Cambridge University Press.

Eden, T.; Levi, A.; Ron, D.; and Seshadhri, C. 2015. Approximately Counting Triangles in Sublinear Time. In *Proc. of the 56th IEEE Symposium on Foundations of Computer Science (FOCS 2015)*, 614–633.

Eden, T.; Ron, D.; and Seshadhri, C. 2018. On approximating the number of k-cliques in sublinear time. In *Proc. of the 50th ACM SIGACT Symposium on Theory of Computing (STOC 2018)*, 722–734.

Eden, T.; Ron, D.; and Seshadhri, C. 2020. Faster sublinear approximation of the number of k-cliques in low-arboricity graphs. In *Proc. of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2020)*, 1467–1478. SIAM.

Eliassi-Rad, T.; Caceres, R.; and LaRock, T. 2019. Incompleteness in Networks: Biases, Skewed Results, and Some Solutions. In *Proc. of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2019)*, 3217–3218. ACM.

Eliassi-Rad, T.; Soundarajan, S.; and Bhadra, S. 2018. Problems with Partially Observed (Incomplete) Networks: Biases, Skewed Results, and Solutions. In *Proc. of the 18th SIAM International Conference on Data Mining (SDM 2018)*. SIAM. URL http://eliassi.org/sdm18tut.html.

Epasto, A.; Esfandiari, H.; and Mirrokni, V. 2019. On-Device Algorithms for Public-Private Data with Absolute Privacy. In *Proc. of the 2019 World Wide Web Conference (WWW 2019)*, 405–416. ACM.

Goldberg, D.; and Roth, F. 2003. Assessing experimentally derived interactions in a small world. *Proceedings of the National Academy of Sciences* 100: 4372.

Goldreich, O. 2017. *Introduction to Property Testing*. Cambridge University Press.

Goldreich, O.; Goldwasser, S.; and Ron, D. 1998. Property Testing and its Connection to Learning and Approximation. *J. ACM* 45(4): 653–750.

Gonen, M.; Ron, D.; and Shavitt, Y. 2011. Counting stars and other small subgraphs in sublinear-time. *SIAM Journal on Discrete Mathematics* 25(3): 1365–1411.

Guptaa, S.; and Guptab, A. 2019. Dealing with Noise Problem in Machine Learning Data-sets: A Systematic Review. *Procedia Computer Science* 161: 466–474.

Hanneke, S.; and Xing, E. 2009. Network Completion and Survey Sampling. In *Proc. of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS 2009)*, volume 5 of *JMLR Proceedings*, 209–215.

Hoogendoorn, M.; and Funk, B. 2018. *Machine Learning for the Quantified Self: On the Art of Learning from Sensory Data*, volume 35 of *Cognitive Systems Monographs*. Spinger.

Jackson, M. 2008. *Social and Economic Networks*. Princeton, NJ, USA: Princeton University Press.

Jha, M.; Seshadhri, C.; and Pinar, A. 2015. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *Proceedings of the 24th International Conference on World Wide Web*, 495–505.

Kim, M.; and Leskovec, J. 2011. The Network Completion Problem: Inferring Missing Nodes and Edges in Networks. In *Proc. of the 11th SIAM International Conference on Data Mining, (SDM 2011)*, 47–58. SIAM / Omnipress.

Kolda, T. G.; Pinar, A.; Plantenga, T.; Seshadhri, C.; and Task, C. 2014. Counting triangles in massive graphs with MapReduce. *SIAM Journal on Scientific Computing* 36(5): S48–S77.

Koskinen, J.; Robins, G.; Wang, P.; and P.E.Pattison. 2013. Bayesian analysis for partially observed network data, missing ties, attributes and actors. *Social Networks* 35: 514–527.

Lakhotia, K.; Kannan, R.; Gaur, A.; Srivastava, A.; and Prasanna, V. 2019. Parallel edge-based sampling for static and dynamic graphs. In *Proceedings of the 16th ACM International Conference on Computing Frontiers*, 125–134.

LaRock, T.; Sakharov, T.; Bhadra, S.; and Eliassi-Rad, T. 2018. Reducing Network Incompleteness Through Online Learning: A Feasibility Study. In *Proc. of the 14th International Workshop on Mining and Learning with Graphs (MLG 2018)*. ACM. URL https://www.mlgworkshop.org/2018/papers/MLG2018_paper_40.pdf.

Leskovec, J.; and Faloutsos, C. 2006. Sampling from Large Graphs. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, 631–636. ACM.

Leskovec, J.; and Krevl, A. 2014. *SNAP Datasets: Stanford Large Network Dataset Collection*. Stanford University.

Morales, P.; Caceres, R.; and Eliassi-Rad, T. 2019. Deep Reinforcement Learning for Task-Driven Discovery of Incomplete Networks. In *Proc. of the 8th International Conference on Complex Networks and Their Applications - Volume*

1 (COMPLEX NETWORKS 2019), volume 881 of *Studies in Computational Intelligence*, 903–914. Springer.

Provost, F.; Melville, P.; and Saar-Tsechansky, M. 2007. Data acquisition and cost-effective predictive modeling: targeting offers for electronic commerce. In *Proc. of the 9th International Conference on Electronic Commerce (ICEC 2007)*, 389–398.

Rossi, R. A.; Zhou, R.; and Ahmed, N. K. 2018. Estimation of graphlet counts in massive networks. *IEEE transactions on neural networks and learning systems* 30(1): 44–57.

Saar-Tsechansky, M.; Melville, P.; and Provost, F. 2009. Active Feature-Value Acquisition. *Management Science* 55: 664–684.

Seshadhri, C.; Pinar, A.; and Kolda, T. G. 2013. Triadic measures on graphs: The power of wedge sampling. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, 10–18. SIAM.

Soundarajan, S.; Eliassi-Rad, T.; Gallagher, B.; and Pinar, A. 2015. MaxOutProbe: An Algorithm for Increasing the Size of Partially Observed Networks. In *Workshop on Networks in the Social and Information Sciences, 29th Conference on Neural Information Processing Systems (NIPS 2015)*. URL http://arxiv.org/abs/1511.06463.

Soundarajan, S.; Eliassi-Rad, T.; Gallagher, B.; and Pinar, A. 2016. MaxReach: Reducing network incompleteness through node probes. In *Proc. of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2016)*, 152–157. IEEE Computer Society.

Soundarajan, S.; Eliassi-Rad, T.; Gallagher, B.; and Pinar, A. 2017. ϵ-WGX: Adaptive Edge Probing for Enhancing Incomplete Networks. In *Proc. of the 2017 ACM on Web Science Conference (WebSci 2017)*, 161–170. ACM.

Tsourakakis, C.; Kolountzakis, M.; and Miller, G. 2011. Triangle Sparsifiers. *J. Graph Algorithms Appl.* 15(6): 703–726.