2015 IEEE 56th Annual Symposium on Foundations of Computer Science

Approximately Counting Triangles in Sublinear Time

Talya Eden
School of Computer Science
Tel Aviv University
Tel Aviv, Israel
talyaa01@gmail.com

Amit Levi
School of EE
Tel Aviv University
Tel Aviv, Israel
amitlev3@post.tau.ac.il

Dana Ron
School of EE
Tel Aviv University
Tel Aviv, Israel
danaron@tau.ac.il

C. Seshadhri
Department of Computer Science
University of California
Santa Cruz, USA
scomandu@ucsc.edu

Abstract

We consider the problem of estimating the number of triangles in a graph. This problem has been extensively studied in both theory and practice, but all existing algorithms read the entire graph. In this work we design a *sublinear-time* algorithm for approximating the number of triangles in a graph, where the algorithm is given query access to the graph. The allowed queries are degree queries, vertex-pair queries and neighbor queries.

We show that for any given approximation parameter 0<epsilon<1, the algorithm provides an estimate hat{t} such that with high constant probability, (1-epsilon) t<hat{t}<(1+epsilon)t, where t is the number of triangles in the graph G. The expected query complexity of the algorithm is O(n/tˆ{1/3} + min {m, mˆ{3/2}/t}) poly(log n, 1/epsilon), where n is the number of vertices in the graph and m is the number of edges, and the expected running time is (n/tˆ{1/3} + mˆ{3/2}/t) poly(log n, 1/epsilon). We also prove that Omega(n/tˆ{1/3} + min {m, mˆ{3/2}/t}) queries are necessary, thus establishing that the query complexity of this algorithm is optimal up to polylogarithmic factors in n (and the dependence on 1/epsilon).

I. Introduction

Counting the number of triangles in a graph is a fundamental algorithmic problem. In the study of complex networks and massive real-world graphs, triangle counting is a key operation in graph analysis for bioinformatics, social networks, community analysis, and graph modeling [1], [2], [3], [4], [5], [6], [7], [8], [9], [10]. In the theoretical computer science community, the primary tool for counting the number of triangles is fast matrix multiplication [11], [12], [13]. On the more applied side, there is a plethora of provable and practical algorithms that employ clever sampling methods for approximate triangle counting [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26]. Triangle counting has also been a popular problem in the streaming setting [27], [28], [29], [30], [31], [32], [33], [26], [34].

All these algorithms read the entire graph, which may be time consuming when the graph is very large. In this work, we focus on *sublinear* algorithms for triangle counting. We assume the following query access to the graph, which is standard for sublinear algorithms that approximate graph parameters. The algorithm can make: (1) Degree queries, in which the algorithm can query the degree d_v of any vertex v. (2) Neighbor queries, in which the algorithm can query what vertex is the i^{th} neighbor of a vertex v, for any $i \leq d_v$. (3) Vertex-pair queries, in which the algorithm can query for any pair of vertices v and u whether (u, v) is an edge.

Gonen et al. [35], who studied the problem of approximating the number of stars in a graph in sublinear time, also considered the problem of approximating the number of triangles in sublinear time. They proved that there is no sublinear approximation algorithm for the number of triangles when the algorithm is allowed to perform degree and neighbor queries (but not pair queries). [1]

They asked whether a sublinear algorithm exists when allowed vertex-pair queries in addition to degree and neighbor queries. We show that this is indeed the case.

A. Results

Let G be a graph with n vertices, m edges, and t triangles. We describe an algorithm that, given an approximation parameter $0 < \epsilon < 1$ and query access to G, outputs an estimate $\widehat{t}$, such that with high constant probability (over the randomness of the algorithm), $(1-\epsilon) \cdot t \leq \widehat{t} \leq (1+\epsilon) \cdot t$. The expected query complexity of the algorithm is

$$\left(\frac{n}{t^{1/3}} + \min\left\{m, \frac{m^{3/2}}{t}\right\}\right) \cdot \text{poly}(\log n, 1/\epsilon) \ ,$$

[1] To be precise, they showed that there exist two families of graphs over $m = \Theta(n)$ edges, such that all graphs in one family have $\Theta(n)$ triangles, all graphs in the other family have no triangles, but in order to distinguish between a random graph in the first family and random graph in the second family, it is necessary to perform $\Omega(n)$ degree and neighbor queries.

0272-5428/15 $31.00 © 2015 IEEE
DOI 10.1109/FOCS.2015.44

and its expected running time is $\left(\frac{n}{t^{1/3}} + \frac{m^{3/2}}{t}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon)$. We show that this result is almost optimal by proving that the number of queries performed by any multiplicative-approximation algorithm for the number of triangles in a graph is

$$\Omega\left(\frac{n}{t^{1/3}} + \min\left\{m, \frac{m^{3/2}}{t}\right\}\right) .$$

B. Overview of the algorithm

For the sake of clarity, we suppress any dependencies on the approximation parameter ϵ and on $\log n$ using the notation $O^*(\cdot)$.

1) A simple oracle-based procedure for a 1/3-estimate: First, let us assume access to an oracle that, given a vertex v, returns t_v, the number of triangles that v is incident to. Note that $t = \sum_v t_v/3$. An unbiased estimate is obtained by sampling, uniformly at random, a (multi-)set S of s vertices, and outputting $t_S = \frac{1}{3} \cdot \frac{n}{s} \cdot \sum_{v \in S} t_v$. Yet this estimate can have extremely large variance (consider the "wheel" graph where there is one vertex with $t_v = \Theta(n)$ and all other t_v's are constant). Inspired by work on estimating the average degree [36], [37], we can reduce the variance by simply "cutting off" the contribution of vertices v for which t_v is above a certain threshold. Call such vertices *heavy*, and denote the remaining *light*. If the threshold is set to $\Theta(t^{2/3}/\epsilon^{1/3})$, then the number of heavy vertices is $O((\epsilon t)^{1/3})$. This implies that the total number of triangles in which all three endpoints are heavy is $O(\epsilon t)$.

Hence, suppose we define $\widetilde{t}_v$ to be t_v if $t_v \leq t^{2/3}/\epsilon^{1/3}$ and 0 otherwise, and consider $\widetilde{t}_S = \frac{1}{3} \cdot \frac{n}{s} \cdot \sum_{v \in S} \widetilde{t}_v$. We can argue that $\mathbf{E}[\widetilde{t}_S] \in [(1/3 - \epsilon)t, t]$, since (roughly speaking) every triangle that contains at least one light vertex is counted at least once. Since $\widetilde{t}_v$ ranges between 0 and $t^{2/3}/\epsilon^{1/3}$, by applying the multiplicative Chernoff bound, a sample of size $s = O^*\left(\frac{n}{t^{1/3}}\right)$ is sufficient to ensure that with high constant probability $\widetilde{t}_S$ is in the range $\left[\left(\frac{1}{3} - 2\epsilon\right) \cdot t, (1 + \epsilon) \cdot t\right]$.

2) Assigning weights to triangles so as to improve the estimate: To improve the approximation, we assign weights to triangles inversely proportional to the number of their light endpoints (rather than assigning a uniform weight of $\frac{1}{3}$ as is done when defining $\widetilde{t}_S = \frac{n}{s} \cdot \sum_{v \in S} \frac{1}{3} \cdot \widetilde{t}_v$). If for each light vertex v we let $\mathrm{wt}(v)$ be the sum over the weights of all triangles that v participates in and for each heavy vertex v we let $\mathrm{wt}(v) = \widetilde{t}_v = 0$, then the expected value of $\frac{n}{s} \cdot \sum_{v \in S} \mathrm{wt}(v)$ is in $[(1 - O(\epsilon)) \cdot t, (1 + O(\epsilon)) \cdot t]$.

To get rid of the fictitious oracle, we must resolve two issues. The first issue is efficiently deciding whether a vertex is heavy or light, and the second is approximating $\frac{n}{s} \cdot \sum_{v \in S} \mathrm{wt}(v)$, assuming we have a procedure for deciding whether a vertex is heavy or light. We next discuss each of these two issues. For convenience, we will assume that the algorithm already has constant factor estimates for m and t. This can be removed by approximating m and performing a geometric search on t.

3) Deciding whether a vertex is heavy: Let v be a fixed vertex with degree d_v. Consider an edge e incident to v, and let u be the other endpoint of this edge. Let t_e denote the number of triangles that e belongs to. Consider the random variable Y defined by selecting, uniformly at random, a neighbor w of u, and setting $Y = d_u$ if (v, w) is an edge (so that (v, u, w) is a triangle) and $Y = 0$ otherwise. Since the number of neighbors of u that form a triangle with v is t_e, the expected value of Y is $\frac{t_e}{d_u} \cdot d_u = t_e$. Now consider selecting (uniformly at random) several edges incident to v, denoted $e_1, \ldots, e_r$, and for each edge e_j selected, defining the corresponding random variable Y_j. Then the expected value of $\frac{1}{r} \sum_{j=1}^{r} Y_j$ is $\frac{1}{d_v} \cdot \sum_{e=(v,u)} t_e = \frac{2}{d_v} \cdot t_v$. If we multiply by $d_v/2$, then we get an unbiased estimator for t_v, which in particular can indicate whether v is heavy or light.

However, once again the difficulty is with the variance of this estimator and the implication on the complexity of the resulting decision procedure. To reduce the variance we modify the procedure described above as follows. First, if d_v is above a certain threshold, then v is also considered heavy (where this threshold is of order $O\left(m/(\epsilon t)^{1/3}\right)$, so that the total number of triangles in which all three endpoints are heavy remains $O(\epsilon t)$). Second, observe that when trying to estimate the number of triangles that an edge $e = (x, y)$ participates in we can either select a random neighbor w of x and check whether $(y, w) \in E$, or we can select a random neighbor w of y and check whether $(x, w) \in E$. Since it is advantageous to consider the endpoint that has a smaller degree, each time we select an edge $e_j = (v, y_j)$ incident to v, we let u_j be the endpoint of e_j that has smaller degree. If d_{u_j} is relatively large (larger than $\sqrt{m}$), then we select $k = \lceil d_{u_j}/\sqrt{m} \rceil$ neighbors of u_j and let Y_j equal d_{u_j} times the fraction among these neighbors that close a triangle with e_j. Finally, we perform a standard median selection over $O(\log n)$ repetitions of the procedure.

Our analysis shows that it suffices to set r (the number of random edges incident to v that are selected) to be $O^*\left(\frac{m^{3/2}}{t}\right)$ so as to ensure the correctness of the procedure (with high probability). In the analysis of the expected query complexity and running time of the procedure we have to take into account the number of iterations $k = \lceil d_{u_j}/\sqrt{m} \rceil$ for each selected (lower degree endpoint) u_j.

4) Estimating $\sum_{v\in S} wt(v)$: Suppose we have a (multi-)set S of vertices such that $\frac{n}{s}\cdot\sum_{v\in S}\mathrm{wt}(v)$ is indeed in $[(1-O(\epsilon))\cdot t,(1+O(\epsilon))\cdot t]$ (which we know occurs with high probability if we select $s=O^*\left(\frac{n}{t^{1/3}}\right)$ vertices uniformly at random). Consider the set of edges incident to vertices in S, where we view edges as directed, so that if there is an edge between v and v' that both belong to S, then (v,v') and (v',v) are considered as two different edges. We denote this set of edges by E_S, and their number by d_S, where $d_S=\sum_{v\in S}d_v$. Suppose that for each edge e we assign a weight $\mathrm{wt}(e)$, which is the sum of the weights of all triangles that it participates in (where the weight of a triangle is as defined previously based on the number of light endpoints that it has). Then $\sum_{e\in E_S}\mathrm{wt}(e)=2\sum_{v\in S}\mathrm{wt}(v)$.

The next idea is to sample *edges* in E_S uniformly at random, and for each selected edge $e=(v,u)$ to estimate $\mathrm{wt}(e)$. An important observation is that since we can query the degrees of all vertices in S, we can efficiently select uniform random edges in E_S (as opposed to the more difficult task of selecting random edges from the entire graph). Similarly to what was described in the decision procedure for heavy vertices, given an edge $e\in E_S$ we let u be its endpoint that has smaller degree. We then select $\lceil\sqrt{m}/d_u\rceil$ random neighbors of u and for each check whether it closes a triangle with e. For each triangle found we check how many heavy endpoints it has (using the aforementioned procedure for detecting heavy vertices) so as to compute the weight of the triangle. In this manner we can obtain random variables whose expected value is $\frac{1}{d_S}\sum_{v\in S}\mathrm{wt}(v)$, and whose variance is not too large (upper bounded by $\sqrt{m}$ times this expected value). We can now take an average over sufficiently many ($O^*\left(\frac{m^{3/2}}{t}\right)$) such random variables and multiply by $d_S\cdot n$. By upper bounding the probability that d_S is much larger than its expected value we can prove that the output of the algorithm is as desired. The expected query complexity and running time of the algorithm are shown to be $O^*\left(\frac{n}{t^{1/3}}+\frac{m^{3/2}}{t}\right)$.

Finally we note that if $t<m^{1/2}$ so that $\frac{m^{3/2}}{t}>m$, then we can replace $\frac{m^{3/2}}{t}$ with m in the upper bound on the query complexity since we can store all queried edges so that no edge needs to be queried more than twice (once from each endpoint).

C. A high level discussion of the lower bound

Proving that every multiplicative-approximation algorithm must perform $\Omega\left(\frac{n}{t^{1/3}}\right)$ queries is fairly straightforward, and our main focus is on proving that $\Omega\left(\min\left\{m,\frac{m^{3/2}}{t}\right\}\right)$ queries are necessary as well. In order to prove this claim we define, for every n, every $1\leq m\leq\binom{n}{2}$ and every $1\leq t\leq\min\{\binom{n}{3},m^{3/2}\}$, a graph G_1 and a family of graphs $\mathcal{G}_2$ for which the following holds: (1) The graph G_1 and all the graphs in $\mathcal{G}_2$ have n vertices and m edges. (2) In G_1 there are no triangles, while the number of triangles in each graph $G\in\mathcal{G}_2$ is $\Theta(t)$. We prove that for values of t such that $t\geq\sqrt{m}$, at least $\Omega\left(\frac{m^{3/2}}{t}\right)$ queries are required in order to distinguish with high constant probability between G_1 and a random graph in $\mathcal{G}_2$. We then prove that for values of t such that $t<\sqrt{m}$, at least $\Omega(m)$ queries are required for this task. We give three different constructions for G_1 and $\mathcal{G}_2$ depending on the value of t as a function of m (where two of the constructions are for subcases of the case that $t\geq\sqrt{m}$). For further discussion of the lower bound, see Section IV. Due to space constraints, in this extended abstract we only described one of these constructions and provide part of the details for the corresponding lower bound proof. All missing details can be found in the full version of this paper [38].

D. Related Work

1) Approximating graph parameters in sublinear time: We build on previous work on approximating the average degree of a graph and the number of stars [36], [37], [35]. Feige [36] investigated the problem of estimating the average degree of a graph, denoted $\overline{d}$, when given query access to the degrees of the vertices. By performing a careful variance analysis, Feige proved that $O\left(\sqrt{n/\overline{d}}/\epsilon\right)$ queries are sufficient in order to obtain a $(\frac{1}{2}-\epsilon)$-approximation of $\overline{d}$. He also proved that a better approximation ratio cannot be achieved in sublinear time using only degree queries. The same problem was considered by Goldreich and Ron [37]. Goldreich and Ron proved that a $(1+\epsilon)$-approximation can be achieved with $O\left(\sqrt{n/\sqrt{\overline{d}}}\right)\cdot\mathrm{poly}(\log n,1/\epsilon)$ queries, if neighbor queries are also allowed.

Building on these ideas, Gonen et al. [35] considered the problem of approximating the number of s-stars in a graph. Their algorithm only used neighbor and degree queries. A major difference between stars and triangles is that the former are non-induced subgraphs, while the latter are. Additional work on sublinear algorithms for estimating other graph parameters include those for approximating the size of the minimum weight spanning tree [39], [40], [41], maximum matching [42], [43] and of the minimum vertex cover [44], [42], [45], [43], [46], [47].

2) Triangle counting: Triangle counting has a rich history. A classic result of Itai and Rodeh showed that triangles can be enumerated in $O(m^{3/2})$ time, and a more elegant algorithm was given by Chiba and Nishizeki [14]. The connections to matrix multiplication have been exploited for faster theoretical algorithms [11], [12], [13]. In practice, there is a diverse body on work on counting triangles using different techniques, for different models. There are serial algorithms based on eigenvalue methods [17], [19], graph sparsification [48], [20], [23], [49], and sampling paths [15], [25]. Triangle counters have been given for MapReduce [50], [22], [51]; external memory models [21]; distributed settings [24]; semi-streaming models [7], [20]; one-pass streaming [27], [28], [29], [30], [31], [32], [33], [26], [34]. It is worth noting that across the board, all these algorithms required reading the entire graph.

Most relevant to our work are various sampling algorithms, that set up a random variable whose expectation is directly related to the triangle count [15], [20], [28], [29], [25], [32], [33], [26], [34]. Typically, this involves sampling some set of vertices or edges to get a set of three vertices. The algorithm checks whether the sampled set induces a triangle, and uses the probability of success to estimate the triangle count. We follow the basic same philosophy. But it is significantly more challenging to set up the "right" random experiment, since we cannot read the entire graph.

II. Preliminaries

Let $G = (V, E)$ be a simple graph with $|V| = n$ vertices and $|E| = m$ edges. For a vertex $v \in V$, we denote by d_v the degree of the vertex, by Γ_v the set of v's neighbors, and by E_v the set of edges incident to v. We let t denote the number of triangles in the graph and t_e denote the number of triangles incident to an edge e. We set $t_v = \sum_{e \in E_v} t_e$. Note that the latter is twice the number of triangles incident to v. The set of triangles incident to an edge e is denoted by T_e, and the set of triangles in the graph G is denoted by T. We use $c, c_1, \ldots$ to denote sufficiently large constants.

We consider algorithms that can sample uniformly in V and perform three types of queries:

1) Degree queries, in which the algorithm may query for the degree d_v of any vertex v of its choice.
2) Neighbor queries, in which the algorithm may query for the i^{th} neighbor of any vertex v of its choice. If $i > d_v$, then a special symbol (e.g. †) is returned. No assumption is made on the order of the neighbors of any vertex.
3) Pair queries, in which the algorithm may ask if there is an edge $(u, v) \in E$ between any pair of vertices u and v.

We sometimes use set notations for operations on multisets. We use the notation $O^*(\cdot)$ to suppress dependencies on the approximation parameter ϵ or on $\log n$.

III. The Algorithm

We start by introducing the notions of heavy and light vertices and how they can be utilized in the context of estimating the number of triangles. We then give a procedure for deciding (approximately) whether a vertex is heavy or light. Using this procedure we give an algorithm for estimating the number of triangles based on the following assumption (which is later removed).

Assumption 1: Our initial algorithm takes as input estimates $\overline{t}$ and $\overline{m}$ on the number of edges and triangles in the graph respectively, such that

1) $t/4 \leq \overline{t} \leq t$.
2) $m/6 \leq \overline{m}$.

Assumption 1 can be easily removed by performing a geometric search on t and using the algorithm from [36] to approximate m, as explained precisely in the proof of Theorem 12.

A. Heavy and light vertices

Definition 1: We say that a vertex v is heavy if $d_v > \frac{2\overline{m}}{(\epsilon\overline{t})^{1/3}}$ or if $t_v > \frac{2\overline{t}^{2/3}}{\epsilon^{1/3}}$. If v is such that $d_v \leq \frac{2\overline{m}}{(\epsilon\overline{t})^{1/3}}$ and $t_v \leq \frac{\overline{t}^{2/3}}{2\epsilon^{1/3}}$, then we say that v is light.

We shall say that a partition (H, L) of V is appropriate (with respect to $\overline{m}$ and $\overline{t}$) if every heavy vertex belongs to H and every light vertex belongs to L.

Note that for an appropriate partition (H, L) both H and L may contain vertices that are neither heavy nor light (but no light vertex belongs to H and no heavy vertex belongs to L).

For a fixed partition (H, L) we associate with each triangle Δ a weight depending on the number of its endpoints that belong to L.

Definition 2: For a triangle Δ we define its weight $\mathrm{wt}_L(\Delta)$ to be

$$\mathrm{wt}_L(\Delta) = \begin{cases} 0 & \text{if no endpoints of } \Delta \text{ belong to } L \\ 1/2\ell & \text{if } \Delta \text{ has } \ell > 0 \text{ endpoints that belong to } L\ . \end{cases}$$

Whenever it is clear for the context, we drop the subscript L and use the notation $\mathrm{wt}(\cdot)$ instead of $\mathrm{wt}_L(\cdot)$.

Claim 1: If (H, L) is appropriate and Assumption 1 holds, then the number of triangles with weight 0 is at most $c_H \cdot \epsilon t$ for some constant c_H.

Proof: There are at most $6(\epsilon\overline{t})^{1/3} \leq 6(\epsilon t)^{1/3}$ vertices v such that $d_v > \frac{2\overline{m}}{(\epsilon\overline{t})^{1/3}}$, and at most $12(\epsilon t)^{1/3}$ vertices v such that $t_v > \frac{2\overline{t}^{2/3}}{\epsilon^{1/3}}$. Therefore, there are at most

$$\binom{18(\epsilon t)^{1/3}}{3} < 6000\epsilon t$$

triangles with all three endpoints in H. Setting $c_H = 6000$ completes the proof. ∎

Recall that T_e denotes the set of triangles that contain an edge e.

Definition 3: For any set T of triangles we define $\mathrm{wt}(T) = \sum_{\Delta \in T} \mathrm{wt}(\Delta)$. For a vertex $v \in L$ we define $\mathrm{wt}(v) = \sum_{e \in E_v} \mathrm{wt}(T_e)$, and $\mathrm{wt}(v) = 0$ for $v \in H$.

Lemma 2: For any partition (H, L), $\sum_{v \in L} \mathrm{wt}(v) \leq t$. If (H, L) is appropriate and Assumption 1 holds, then $\sum_{v \in L} \mathrm{wt}(v) \in [t(1 - c_H \cdot \epsilon), t]$.

Proof: Let $\chi(e, \Delta)$ be an indicator variable such that $\chi(e, \Delta) = 1$ if Δ contains the edge e, and $\chi(e, \Delta) = 0$ otherwise. Consider a triangle Δ that contains $\ell > 0$ light vertices. Then

$$\sum_{v \in L} \sum_{e \in E_v} \chi(e, \Delta) = 2\ell = 1/\mathrm{wt}(\Delta)\ .$$

If $\ell = \mathrm{wt}(\Delta) = 0$, then the above expression equals 0. By interchanging summations,

$$\sum_{v \in L} \mathrm{wt}(v) = \sum_{v \in L} \sum_{e \in E_v} \mathrm{wt}(T_e) = \sum_{\Delta \in T} \mathrm{wt}(\Delta) \sum_{v \in L} \sum_{e \in E_v} \chi(e, \Delta) = t - |\{\Delta \mid \mathrm{wt}(\Delta) = 0\}|.$$

Clearly for any partition (H, L) the above expression is at most t. On the other hand, If (H, L) is appropriate and Assumption 1 holds, then by Claim 1 we have that $|\{\Delta \mid \mathrm{wt}(\Delta) = 0\}| \leq c_H \cdot \epsilon t$, and the lemma follows. ∎

Theorem 3: Let $s = (c \log(n/\epsilon)/\epsilon^3)n/\overline{t}^{1/3}$ where c is a constant, and let S be a sample of s vertices $v_1, v_2, \ldots, v_s$ that are selected uniformly, independently at random. Then

$$\mathbf{E}\left[\frac{1}{s}\sum_{i=1}^{s} \mathrm{wt}(v_i)\right] \leq \frac{t}{n}\ .$$

Furthermore, if (H, L) is appropriate and Assumption 1 holds, then

$$\mathbf{E}\left[\frac{1}{s}\sum_{i=1}^{s} \mathrm{wt}(v_i)\right] \in [t(1 - c_H \cdot \epsilon)/n, t/n]$$

and for a sufficiently large constant c,

$$\Pr\left[\frac{1}{s}\sum_{i=1}^{s} \mathrm{wt}(v_i) < t(1 - 2c_H \cdot \epsilon)/n\right] < \epsilon^2/n\ .$$

Proof: Let Y denote the random variable $Y = \frac{1}{s}\sum_{i=1}^{s} \mathrm{wt}(v_i)$. By the first part of Lemma 2, $\mathbf{E}\left[\frac{1}{s}\sum_{i=1}^{s} \mathrm{wt}(v_i)\right] \leq t/n$. Now assume that (H, L) is appropriate and Assumption 1 holds. The claim regarding the expected value of Y follows from the second part of Lemma 2, so it remains to prove the claim regarding the deviation from the expected value. Note that $\mathrm{wt}(v) \leq t_v$ for every vertex v, which for $v \in L$ is at most $\frac{2\overline{t}^{2/3}}{\epsilon^{1/3}}$. By the multiplicative Chernoff bound and by Item 1 in Assumption 1,

$$\Pr\left[Y < (1 - \epsilon)\mathbf{E}[Y]\right] < \exp\left(-\frac{\epsilon^2 \mathbf{E}[Y]s}{4t^{2/3}/\epsilon^{1/3}}\right) < \exp\left(-\frac{\epsilon^2 \cdot c\log(n/\epsilon)(n/\epsilon\overline{t}^{1/3}) \cdot t/(2n)}{4\overline{t}^{2/3}/\epsilon^{1/3}}\right) < \frac{\epsilon^2}{n}\ ,$$

where the last inequality holds for a sufficiently large constant c. ∎

B. A procedure for deciding whether a vertex is heavy

In this subsection we provide a procedure for deciding (approximately) whether a given vertex v is heavy or light.

`Heavy`(v)

1) If $d_v > 2\overline{m}/(\epsilon\overline{t})^{1/3}$, output **heavy**.
2) For $i = 1, 2, \ldots, 10 \log n$:
   a) For $j = 1, 2, \ldots, s = 4\overline{m}^{3/2}/\epsilon^2\overline{t}$:
      i) Select an edge $e \in E_v$ uniformly, independently and at random, and let u be its endpoint with the smaller degree.
      ii) For $k = 1, 2, \ldots, r = \lceil d_u/\sqrt{\overline{m}} \rceil$:
         A) Pick a neighbor w of u uniformly at random.
         B) If e with w forms a triangle, set $Z_k = d_u$, else $Z_k = 0$.
      iii) Set $Y_j = \frac{1}{r}\sum_k Z_k$.
   b) Set $X_i = \frac{d_v}{s}\sum_j Y_j$.
3) If the median of the X_i variables is greater than $\overline{t}^{2/3}/\epsilon^{1/3}$, output **heavy**, else output **light**.

We have three nested loops, with loop variables i, j, k respectively. We refer to these as "iteration i", "iteration j", and "iteration k".

Lemma 4: For any iteration i, $\Pr[|X_i - t_v| > \epsilon \cdot \max(t_v, \overline{t}d_v/\overline{m})] < 1/4$.

Proof: Fix an iteration j and let e_j denote the edge chosen in the j^{th} iteration and u_j denote its smaller degree endpoint. We use $\mathcal{E}_j$ to denote the event of e_j being chosen. Conditioned on the event $\mathcal{E}_j$, the probability of finding a triangle in any iteration k is t_{e_j}/d_{u_j}. Hence,

$$\mathbf{E}[Z_k \mid \mathcal{E}_j] = \frac{t_{e_j}}{d_{u_j}} \cdot d_{u_j} = t_{e_j},$$

and

$$\mathbf{Var}[Z_k \mid \mathcal{E}_j] \leq \mathbf{E}[Z_k^2 \mid \mathcal{E}_j] \leq d_{u_j} \cdot \mathbf{E}[Z_k \mid \mathcal{E}_j].$$

By linearity of expectation,

$$\mathbf{E}[Y_j \mid \mathcal{E}_j] = \mathbf{E}\left[\frac{1}{r}\sum_{k=1}^{r} Z_k \mid \mathcal{E}_j\right] = \frac{1}{r}\sum_{k=1}^{r} \mathbf{E}\left[Z_k \mid \mathcal{E}_j\right] = t_{e_j}.$$

By the independence of the Z_k variables,

$$\begin{aligned}\mathbf{Var}[Y_j \mid \mathcal{E}_j] &= \mathbf{Var}\left[\frac{1}{r}\sum_{k=1}^{r} Z_k \mid \ \mathcal{E}_j\right] = \frac{1}{r^2}\sum_{k=1}^{r} \mathbf{Var}\left[Z_k \mid \mathcal{E}_j\right] \leq \frac{1}{r^2}\sum_{k=1}^{r} d_{u_j} \cdot \mathbf{E}\left[Z_k \mid \mathcal{E}_j\right] \\ &= \frac{d_{u_j}}{r^2} \cdot r \cdot t_{e_j} \leq \sqrt{\overline{m}} \cdot t_{e_j}.\end{aligned}$$

The conditioning can be removed to yield

$$\mathbf{E}[Y_j] = \sum_{e \in E_v} \frac{1}{d_v} \cdot \mathbf{E}[Y_j \mid \mathcal{E}_j] = \frac{1}{d_v} \cdot \sum_{e \in E_v} t_e = \frac{t_v}{d_v}.$$

By the law of total variance and the law of total expectation,

$$\begin{aligned}\mathbf{Var}[Y_j] &= \mathbf{E}_{e_j}\left[\mathbf{Var}\left[Y_j \mid \mathcal{E}_j\right]\right] + \mathbf{Var}_{e_j}\left[\mathbf{E}\left[Y_j \mid \mathcal{E}_j\right]\right] \\ &\leq \mathbf{E}_{e_j}\left[\sqrt{\overline{m}} \cdot \mathbf{E}\left[Y_j \mid \mathcal{E}_j\right]\right] + \mathbf{Var}_{e_j}[t_v/d_v] \\ &= \sqrt{\overline{m}} \cdot \mathbf{E}\left[Y_j\right] \ .\end{aligned}$$

Let $\overline{Y} = \frac{1}{s}\sum_j Y_j$. It holds that

$$\mathbf{Var}[\overline{Y}] = \mathbf{Var}\left[\frac{1}{s}\sum_{j=1}^{s} Y_j\right] = \frac{1}{s^2}\sum_{j=1}^{s}\mathbf{Var}[Y_j] \leq \frac{1}{s^2}\sum_{j=1}^{s}\sqrt{\overline{m}}\cdot\mathbf{E}\left[Y_j\right] = \frac{\sqrt{\overline{m}}}{s}\cdot\mathbf{E}\left[\frac{1}{s}\sum_{j=1}^{s} Y_j\right]$$
$$= \frac{\sqrt{\overline{m}}}{s}\mathbf{E}\left[\overline{Y}\right]. \tag{1}$$

By Chebyshev's inequality and Equation (1),

$$\Pr\left[|\overline{Y} - \frac{t_v}{d_v}| > \epsilon\max\left(\frac{t_v}{d_v}, \frac{\overline{t}}{\overline{m}}\right)\right] < \frac{\mathbf{Var}[\overline{Y}]}{\epsilon^2\max(t_v/d_v, \overline{t}/\overline{m})^2} \leq \frac{\sqrt{\overline{m}}(t_v/d_v)}{\epsilon^2(4/\epsilon^2)(\overline{m}^{3/2}/\overline{t})\cdot(t_v/d_v)\cdot(\overline{t}/\overline{m})} = 1/4.$$

Since $X = d_v\cdot\overline{Y}$, we have that $\Pr[|X_i - t_v| > \epsilon\max(t_v, \overline{t}d_v/\overline{m})] < 1/4$. ∎

Lemma 5: For every vertex v, if v is heavy, then a call to `Heavy`(v) returns **heavy** with probability at least $1-1/n^2$. If v is light, then a call to `Heavy`(v) returns **light** with probability at least $1-1/n^2$.

Proof: First consider a heavy vertex v. Clearly, if $d_v > 2\overline{m}/(\epsilon\overline{t})^{1/3}$, then v is declared heavy. Therefore, assume that $t_v > 2\overline{t}^{2/3}/\epsilon^{1/3}$ and $d_v \leq 2\overline{m}/(\epsilon\overline{t})^{1/3}$, so that $\overline{t}d_v/\overline{m} \leq 2\overline{t}^{2/3}/\epsilon^{1/3}$. By Lemma 4, for any iteration i, $\Pr\left[|X_i - t_v| > \epsilon t_v\right] < 1/4$. Hence, $\Pr[X_i < \overline{t}^{2/3}/\epsilon^{1/3}] < 1/4$, and by Chernoff, the probability that the median of the X_i variables (where $i = 1,\ldots,10\log n$) will be greater than $\overline{t}^{2/3}/\epsilon^{1/3}$ is at least $1-1/n^2$. Hence `Heavy`(v) outputs **heavy** with probability at least $1-1/n^2$.

Now consider a light vertex v. Since $d_v \leq 2\overline{m}/(\epsilon\overline{t})^{1/3}$ and $t_v \leq \overline{t}^{2/3}/2\epsilon^{1/3}$, it holds that $\overline{t}d_v/\overline{m} \leq 2\overline{t}^{2/3}/\epsilon^{1/3}$. Therefore, by Lemma 4, $\Pr[|X_i - t_v| > \epsilon(2\overline{t}^{2/3}/\epsilon^{1/3})] < 1/4$, and the probability that the median will be less than $\overline{t}^{2/3}/\epsilon^{1/3}$ is at least $1-1/n^2$. Hence v is declared **light** with probability at least $1-1/n^2$. ∎

The following is a corollary of Lemma 5.

Corollary 6: Consider running `Heavy` for all the vertices in the graph. Let H denote the set of vertices that are declared heavy and let L denote the set of vertices that are declared light. Then, with probability at least $1-1/n$, the partition (H, L) is appropriate (as defined in Definition 1).

We now turn to analyze the running time of `Heavy`. The proof will be similar to the complexity analysis of the exact triangle counter of Chiba and Nishizeki [14].

Lemma 7: If Item 2 in Assumption 1 holds, then for every vertex v the expected running time of `Heavy`(v) is $O^*(\overline{m}^{3/2}/\overline{t})$.

Proof: We first argue that the expected time to generate a single sample of Y_j is $O(1)$. Our query model allows for selecting an edge in E_v uniformly at random by a single query. If $d_v \leq \sqrt{\overline{m}}$, then the degree of the smaller endpoint for any $e \in E_v$ is at most $\sqrt{\overline{m}}$. Hence a sample is clearly generated in $O(1)$ time. Suppose that $d_v > \sqrt{\overline{m}}$. If an edge $e = (v,u)$ is sampled, then the runtime is $O(1+\min(d_v, d_u)/\sqrt{\overline{m}})$. Hence, the expected runtime to generate Y_j is, up to constant factors, at most:

$$\frac{1}{d_v}\sum_{u\in\Gamma_v}\left(1+\frac{\min\{d_v,d_u\}}{\sqrt{\overline{m}}}\right) \leq 1+\frac{1}{\sqrt{\overline{m}}\cdot d_v}\sum_{u\in\Gamma_v} d_u \leq 1+\frac{1}{\sqrt{\overline{m}}\cdot d_v}\sum_{u\in V} d_u \leq 1+\frac{2m}{\sqrt{\overline{m}}\cdot d_v} \leq 5,$$

where the last inequality follows from Item 2 in Assumption 1

By the above, each iteration of the 'for' loop in Step 2a takes $O(1)$ time in expectation. Therefore, together, all iterations of Step 2a take $O(\overline{m}^{3/2}/(\epsilon\overline{t}))$ time in expectation, and since it is repeated $O(\log n)$ times, the expected running time of the procedure is $(\overline{m}^{3/2}/\overline{t})\cdot\mathrm{poly}(\log n, 1/\epsilon)$. ∎

C. Estimating the number of triangles given $\overline{m}$ and $\overline{t}$

We are now ready to present an algorithm `Estimate-with-advice` that takes $\overline{m}, \overline{t}$ as input ("advice"), and outputs an estimate of t. Later, we employ the the average degree approximation algorithm of Feige [36] and a geometric search to get the bonafide algorithm that estimates t without any initial estimates $\overline{m}$ and $\overline{t}$. In what follows we rely on the following assumption.

Assumption 2: We will assume that the random coins used by `Heavy` are fixed in advance, and that the partition (H, L) as defined in Corollary 6 is indeed appropriate.

By Corollary 6 this assumption only adds $1/n$ to the error probability in all subsequent probability bounds. Recall that we use $c, c_1, \ldots$ to denote sufficiently large constants.

`Estimate-with-advice`$(\overline{m}, \overline{t}, \epsilon)$

1) Sample $s_1 = c_1\epsilon^{-3}\log(n/\epsilon)(n/\overline{t}^{1/3})$ vertices, uniformly, independently and at random. Denote the chosen multiset S.
2) Set up a data structure to enable sampling vertices in S proportional to their degree.
3) For $i = 1, 2, \ldots, s_2 = c_2\epsilon^{-4}(\log^2 n)(\overline{m}^{3/2}/\overline{t})$:
   a) Sample $v \in S$ proportional to d_v and sample $e \in E_v$ uniformly at random. Let u be lower degree endpoint.
   b) If $d_u \leq \sqrt{\overline{m}}$, set $r = 1$ with probability $d_u/\sqrt{\overline{m}}$ and set $r = 0$ otherwise. If $d_u > \sqrt{\overline{m}}$, set $r = \lceil d_u/\sqrt{\overline{m}} \rceil$.
   c) Repeat for $j = 1, 2, \ldots, r$:
      i) Pick a neighbor w of u uniformly at random.
      ii) If e and w do not form a triangle, then set $Z_j = 0$.
      iii) If e and w form a triangle Δ: call `Heavy` for all vertices in Δ, and let
      $$Z_j = \begin{cases} 0 & \text{if } \texttt{Heavy}(v) \text{ returned } \textbf{heavy} \\ \max(d_u, \sqrt{\overline{m}}) \cdot \mathrm{wt}(\Delta) & \text{otherwise} \end{cases}.$$
   d) Set $Y_i = \frac{1}{r}\sum_{j=1}^{r} Z_j$. (If $r = 0$, set $Y_i = 0$.)
4) Output $X = \frac{n}{s_1 s_2} \cdot \left(\sum_{v \in S} d_v\right) \cdot \left(\sum_{i=1}^{s_2} Y_i\right)$.

Recall that c_H is the constant defined in Claim 1.

Theorem 8: For X as defined in Step 4 of `Estimate-with-advice`, $E[X] \leq t$. Moreover, if (H, L) is appropriate and Assumption 1 holds, then $\mathbf{E}[X] \in [t(1 - c_H \cdot \epsilon), t]$ and $\Pr[X < t(1 - 3c_H \cdot \epsilon)] < 3\epsilon/\log n$.

There are three "levels" of randomness. First is the choice of S, second is the choice of e (Step 3a), and finally the Z_j's. For analyzing the randomness in any level, we condition on the previous levels. Before proving the theorem, we present the following definition and claim.

Definition 4: Let S be a multiset of s_1 vertices. We say that S is `good` if $\sum_{v \in S} \mathrm{wt}(v)/s_1 \geq t(1 - 2c_H \cdot \epsilon)/n$. We say that S is `great` if, in addition to being good, $d_S = \sum_{v \in S} d_v \leq s_1(2m/n)(\log n)/\epsilon$.

Claim 9: Fix the choice of the set S, and let $d_S = \sum_{v \in S} d_v$. For every i, $\mathbf{E}[Y_i \mid S] = d_S^{-1} \sum_{v \in S} \mathrm{wt}(v)$ and $\mathbf{Var}[Y_i \mid S] \leq \sqrt{\overline{m}} \cdot \mathbf{E}[Y_i \mid S]$.

Proof: This is similar to the argument in Lemma 4. Let v_i be the chosen vertex in the i^{th} iteration, and let e_i be the chosen edge. We refer to this event by $\mathcal{E}_i$, and condition over the set S being chosen and the event $\mathcal{E}_i$. Denote by u_i the lower degree endpoint of e_i.

If $\texttt{Heavy}(v_i)$=**heavy**, then $\mathbf{E}[Y_i \mid S, \mathcal{E}_i] = 0$ and $\mathbf{Var}[Y_i \mid S, \mathcal{E}_i] = 0$. If $\texttt{Heavy}(v_i)$=**light**, then there are two possibilities. If $d_{u_i} \leq \sqrt{\overline{m}}$ then,

$$\mathbf{E}[Y_i \mid S, \mathcal{E}_i] = \frac{d_{u_i}}{\sqrt{\overline{m}}} \sum_{\Delta \in T_{e_i}} \frac{1}{d_{u_i}} \cdot \sqrt{\overline{m}} \cdot \mathrm{wt}(\Delta) = \mathrm{wt}(T_{e_i}),$$

where the weight of the triangles is defined with respect to the (H, L) partition induced by `Heavy` (which we assume is appropriate). Since the maximum value of Y_i in this case is at most $\sqrt{\overline{m}}$,

$$\mathbf{Var}[Y_i \mid S, \mathcal{E}_i] \leq \mathbf{E}[Y_i^2 \mid S, \mathcal{E}_i] \leq \sqrt{\overline{m}} \cdot \mathbf{E}[Y_i \mid S, \mathcal{E}_i].$$

Now consider the case that $d_{u_i} > \sqrt{\overline{m}}$. In order to bound the variance of the Y_i variables we first analyze the expectation and variance of the Z_j variables. It holds that

$$\mathbf{E}[Z_j \mid S, \mathcal{E}_i] = \sum_{\Delta \in T_{e_i}} \frac{1}{d_{u_i}} \cdot d_{u_i} \cdot \mathrm{wt}(\Delta) = \mathrm{wt}(T_{e_i}),$$

and $\mathbf{Var}[Z_j \mid S, \mathcal{E}_i] \leq d_{u_i}\mathbf{E}[Z_j \mid S, \mathcal{E}_i]$. By linearity of expectation,

$$\mathbf{E}[Y_i \mid S, \mathcal{E}_i] = \mathrm{wt}(T_{e_i}).$$

By independence of the $(Z_j \mid S, \mathcal{E}_i)$ variables and linearity of expectation,

$$\mathbf{Var}[Y_i \mid S, \mathcal{E}_i] = \mathbf{Var}\left[\frac{1}{r}\sum_{j=1}^{r} Z_j \mid S, \mathcal{E}_i\right] = \frac{1}{r^2}\sum_{j=1}^{r}\mathbf{Var}\left[Z_j \mid S, \mathcal{E}_i\right] \leq \frac{1}{r^2}\sum_{j=1}^{r} d_{u_i}\mathbf{E}\left[Z_j \mid S, \mathcal{E}_i\right]$$
$$= \frac{d_{u_i}}{r}\cdot\mathbf{E}\left[\frac{1}{r}\sum_{j=1}^{r} Z_j \mid S, \mathcal{E}_i\right] \leq \sqrt{\overline{m}}\cdot\mathbf{E}[Y_i \mid S, \mathcal{E}_i].$$

We remove the conditioning on $\mathcal{E}_i$:

$$\mathbf{E}[Y_i \mid S] = \sum_{v\in S\cap L}\frac{d_v}{d_S}\times\frac{1}{d_v}\sum_{e\in E_v}\mathrm{wt}(T_e) = d_S^{-1}\sum_{v\in S\cap L}\sum_{e\in E_v}\mathrm{wt}(T_e) = d_S^{-1}\sum_{v\in S}\mathrm{wt}(v).$$

By the law of total variance and the law of total expectation,

$$\mathbf{Var}[Y_i \mid S] = \mathbf{E}_{e_i}\left[\mathbf{Var}\left[Y_i \mid S, \mathcal{E}_i\right]\right] + \mathbf{Var}_{e_i}\left[\mathbf{E}\left[Y_i \mid S, \mathcal{E}_i\right]\right] = \mathbf{E}_{e_i}\left[\mathbf{Var}\left[Y_i \mid S, \mathcal{E}_i\right]\right]$$
$$\leq \mathbf{E}_{e_i}\left[\sqrt{\overline{m}}\cdot\mathbf{E}\left[Y_i \mid S, \mathcal{E}_i\right]\right] = \sqrt{\overline{m}}\cdot\mathbf{E}[Y_i \mid S].$$

This completes the proof of Claim 9. ■

Proof of Theorem 8: For a fixed set S, let X_S denote the sum $X_S = \frac{n}{s_1 s_2}\left(\sum_{v\in S} d_v\right)\cdot\left(\sum_{i=1}^{s_2} Y_i\right)$ (as defined in Step 4 of `Estimate-with-advice`), given that the set S in chosen in Step 1. By the definition of X_S and by Claim 9,

$$\mathbf{E}[X_S] = \frac{n d_S}{s_1}\mathbf{E}[Y_i \mid S] = \frac{n}{s_1}\sum_{v\in S}\mathrm{wt}(v).$$

By Theorem 3, $\mathbf{E}_S\left[\frac{1}{s_1}\sum_{v\in S} wt(v)\right] \in [t(1-c_H\cdot\epsilon), t]$, implying that

$$\mathbf{E}[X_S] \in [t(1-c_H\cdot\epsilon), t].$$

By Theorem 3, Definition 4 and Assumption 2, S is good with probability at least $1-\epsilon^2/n$. The expected value, over S, of d_S is $\mathbf{E}_s[d_S] = s_1\cdot\frac{2m}{n}$. By Markov's inequality,

$$\Pr_S\left[d_S > s_1\cdot\frac{2m}{n}\cdot\frac{\log n}{\epsilon}\right] < \frac{\epsilon}{\log n}.$$

By taking a union bound, the probability that S is great is at least $1-2\epsilon/\log n$. For a fixed choice of S, let $Y_S = \frac{1}{s_2}\sum_{i=1}^{s_2} Y_i$. It holds that

$$\mathbf{Var}\left[Y_S\right] = \frac{1}{s_2^2}\sum_{i=1}^{s_2}\mathbf{Var}\left[Y_i \mid S\right] \leq \frac{1}{s_2^2}\sum_{i=1}^{s_2}\sqrt{\overline{m}}\cdot\mathbf{E}[Y_i \mid S] \leq \frac{\sqrt{\overline{m}}}{s_2}\cdot\mathbf{E}\left[Y_S\right].$$

Applying Chebyshev's inequality, we get that

$$\Pr\left[|Y_S - \mathbf{E}[Y_S]| > \epsilon\mathbf{E}[Y_S]\right] < \frac{\mathbf{Var}[Y_S]}{\epsilon^2\cdot\mathbf{E}[Y_S]^2} \leq \frac{\sqrt{\overline{m}}\cdot\mathbf{E}[Y_S]}{\epsilon^2(c_2\epsilon^{-4}\log^2 n)(\overline{m}^{3/2}/\overline{t})\cdot\mathbf{E}[Y_S]^2}$$
$$= \frac{\epsilon^2}{c_2(\log^2 n)(\overline{m}/\overline{t})\cdot\mathbf{E}[Y_S]}.$$

Note that $\mathbf{E}[Y_S] = d_S^{-1}\sum_{v\in S}\mathrm{wt}(v)$, which for a great S is at least $(t/4m)(\log n/\epsilon)$. Therefore, by Assumption 1, for a sufficiently large constant c_2,

$$\Pr\left[|Y_S - \mathbf{E}[Y_S]| > \epsilon\mathbf{E}[Y_S]\right] \leq \frac{\epsilon}{\log n}.$$

By the definition of X_S in Step 4 of the algorithm, X_S is just a scaling of Y_S. Therefore,

$$\Pr\left[|X_S - \mathbf{E}[X_S]| > \epsilon\mathbf{E}[X_S]\right] \leq \frac{\epsilon}{\log n}.$$

Note that $\mathbf{E}[X_S] = \frac{n}{s_1} \sum_{v \in S} \mathrm{wt}(v)$, which for a great S is at least $t(1 - 2c_H \cdot \epsilon)$. Hence, for a great S,

$$\Pr\left[X_S < (1 - 3c_H \cdot \epsilon) \cdot \frac{t}{n}\right] \leq \frac{\epsilon}{\log n}.$$

The probability of S not being great is at most $2\epsilon/\log n$. We apply the union bound to remove the conditioning, so we get

$$\Pr\left[X < (1 - 3c_H \cdot \epsilon) \cdot \frac{t}{n}\right] \leq \frac{3\epsilon}{\log n},$$

which completes the proof. □

Theorem 10: If Item 2 in Assumption 1 holds then the expected running time of `Estimate-with-advice` is $O^*(n/\overline{t}^{1/3} + \overline{m}^{3/2}/\overline{t})$.

Proof: The sampling of S is done in $O^*(n/\overline{t}^{1/3})$ time. Generating the Z_j variables, without the calls to `Heavy`, takes time $O^*(\overline{m}^{3/2}/\overline{t})$ in expectation, by an argument identical to that in the proof of Lemma 7. Therefore, it remains to bound the running time resulting from calls to `Heavy`.

Let us compute the expected number of triangles found during the run of the algorithm. In each iteration i, conditioned on choosing an edge e, the expected number of triangles found is at most $2(d_u/\sqrt{\overline{m}})(t_e/d_u) = 2t_e/\sqrt{\overline{m}}$. Averaging over the edges, the expected number of triangles found in a single iteration is at most $6t/(m \cdot \sqrt{\overline{m}})$ which by Item 2 in Assumption 1 is $O(\overline{t}/\overline{m}^{3/2})$. There are $O(\overline{m}^{3/2}/\overline{t}) \cdot \mathrm{poly}(\log n, 1/\epsilon)$ iterations, leading to a total of $O^*(1)$ expected triangles. Thus, there are $O^*(1)$ expected calls to `Heavy`, each taking $O^*(\overline{m}^{3/2}/\overline{t})$ time by Lemma 7. Together with the above, we get an expected running time of $O(n/\overline{t}^{1/3} + \overline{m}^{3/2}/\overline{t}) \cdot \mathrm{poly}(\log n, 1/\epsilon)$. ■

D. The final algorithm

We are now ready to present an algorithm that requires no prior knowledge regarding m and t.

`Estimate`(ϵ)

1) Let $\epsilon' = \epsilon/3c_H$, where c_H is the constant defined in Claim 1.
2) Invoke Feige's algorithm [36] for approximating the average degree of a graph $10 \log n$ times. Let $\overline{d}$ be the median value of all invocations.
3) Let $\overline{m} = n\overline{d}/2$.
4) Let $\widetilde{t} = n^3$.
5) While $\widetilde{t} \geq 1$
   a) For $\overline{t} = n^3, n^3/2, n^3/4, \ldots, \widetilde{t}$:
      i) For $i = 1, \ldots, c\epsilon^{-1} \log\log n$:
         A) Let $X_i =$`Estimate-with-advice`$(\epsilon', \overline{m}, \overline{t})$.
      ii) Let $X = \min_i\{X_i\}$.
      iii) If $X \geq \overline{t}$ **return** X.
   b) Let $\widetilde{t} = \widetilde{t}/2$.

Before analyzing the correctness and running time of the algorithm, we present the following simple proposition, whose proof we give for the sake of completeness.

Proposition 11: For every graph G, $t \leq \frac{4}{3}m^{3/2}$.

Proof:

$$t = \frac{1}{3}\sum_{v \in V}\frac{1}{2}t_v \leq \frac{1}{6}\left(\sum_{v:\, d_v > \sqrt{m}} t_v + \sum_{v:\, d_v \leq \sqrt{m}} 2d_v^2\right) \leq \frac{1}{6}\left(2\sqrt{m} \cdot 2m + 2\sqrt{m}\sum_{v:\, d_v \leq \sqrt{m}} d_v\right) \leq \frac{4}{3}m^{3/2}.$$

■

Theorem 12: Algorithm `Estimate`(ϵ) returns a value X, such that $(1-\epsilon)t \leq X \leq (1+\epsilon)t$, with probability at least 5/6. The expected query complexity of the algorithm is $O^*\left(n/t^{1/3} + \max\left\{m, m^{3/2}/t\right\}\right)$ and the expected running time of the algorithm is $O^*(n/t^{1/3} + m^{3/2}/t)$.

Proof: We first prove that the value of X is as stated in the theorem. Let d_{avg} denote the average degree of vertices in G. The algorithm from [36] returns a value $\overline{d}$ such that, with probability at least 2/3, $\overline{d} \in [d_{avg}/(2+\gamma), d_{avg}]$

for a constant γ. Since we take the median value of $10 \log n$ invocations, it follows from Chernoff's inequality that $\overline{m}$ is as stated in Item 2 of Assumption 1 with probability at least $1 - 1/\text{poly}(n)$. Assume that this is indeed the case.

Before analyzing the algorithm `Estimate` as described above, first consider executing Step 5a with $\widetilde{t} = 1$. That is, rather than running both an outer loop over decreasing values of $\widetilde{t}$ and an inner loop over decreasing values of $\overline{t}$, we only run a single loop over decreasing value of $\overline{t}$, starting with $\overline{t} = n^3$. By the first part of Theorem 8 and by Markov's inequality, for each value of $\overline{t}$ and for each i, $\Pr[X_i \leq (1+\epsilon)t] > \epsilon/2$, where X_i as defined in Step 5(a)iA. Therefore, for each value of $\overline{t}$, the minimum estimate X (as defined in Step 5(a)iii) is at most $(1+\epsilon)t$, with probability at least $1 - 1/\log^3 n$. It follows that for each $\overline{t}$ such that $\overline{t} > 2t$, we have that $X < \overline{t}$ with probability at least $1 - 1/\log^3 n$, and the algorithm will continue with $\overline{t} = \overline{t}/2$. Once we reach a value of $\overline{t}$ for which $t/4 \leq \overline{t} \leq t/2$, Item 1 in Assumption 1, regarding $\overline{t}$, holds. By the second part of Theorem 8, $X_i \in [(1-\epsilon)t, (1+\epsilon)t]$ for every i with probability at least $1 - c/\log n$. Hence, we have that

$$\overline{t} \leq \frac{1}{2}t \leq (1-\epsilon)t \leq X \leq (1+\epsilon)t,$$

with probability at least $1 - c/\log n$. Therefore, we halt and return correct X.

If however we do reach a value $\overline{t}$ such that $\overline{t} \leq t/4$, since Assumption 1 does not hold, we cannot lower bound X, implying that we can no longer bound the probability that $X < \overline{t}$. Therefore we might continue running with decreasing values of t, causing the running time to exceed the desired bound of $O^*(n/t^{1/3} + m^{3/2}/t)$. In order to avoid this scenario, we run both an outer loop over $\widetilde{t}$ and an inner loop over $\overline{t}$. Specifically, starting with $\widetilde{t} = n^3$, whenever we halve $\widetilde{t}$, we run over all values of $\overline{t} = n^3, n^3/2, \ldots$, until we reach $\widetilde{t}$. This implies that for every value of $\widetilde{t} > 2t$ the probability of returning an incorrect estimate, that is, outside the range of $(1-\epsilon)t \leq X \leq (1+\epsilon)t$, is at most $1 - 1/\log^2 n$. On the other hand, for values of $\widetilde{t}$ such that $\widetilde{t} \leq t/2$ the probability of returning a correct estimate (within $(1-\epsilon)t \leq X \leq (1+\epsilon)t$) is at least $1 - c/\log n$. A union bound over all failure probabilities gives a success probability of at least $5/6$.

We now turn to analyze the query complexity and running time of the algorithm. By [36], the expected running time of the average degree approximation algorithm is $O^*(n/\sqrt{m})$. By Theorem 10, conditioned on $\overline{m}$ satisfying Item 2 in Assumption 1, the expected running time of `Estimate-with-advice`$(\epsilon, \overline{m}, \overline{t})$ is $O^*(n/\overline{t}^{1/3} + \overline{m}^{3/2}/t)$. It follows from Proposition 11 that $n/\sqrt{m} = O(n/t^{1/3})$, implying that the running time is determined by the value of $\overline{m}$ and by the smallest value of $\overline{t}$ that `Estimate-with-advice`$(\epsilon, \overline{m}, \overline{t})$ is invoked with.

Recall that whenever we halve the value of $\tilde{t}$, we run with all values $\overline{t} = n^3, n^3/2, \ldots$. This, together with the fact that when running with $t/4 \leq \overline{t} \leq t/2$ we halt with probability at least $1 - c/\log n$, implies that the probability of reaching a value $\tilde{t} = t/2^k$ is at most $(c/\log n)^k$. Therefore, the expected running time, conditioned on $\overline{m}$ satisfying Item 2 in Assumption 1, is bounded by

$$\log^2 n \cdot O^*\left(\frac{n}{t^{1/3}} + \frac{\overline{m}^{3/2}}{t}\right) + \sum_{k=1}^{\log n} (c/\log n)^k \cdot 2^k \cdot O^*\left(\frac{n}{t^{1/3}} + \frac{\overline{m}^{3/2}}{t}\right) = O^*\left(\frac{n}{t^{1/3}} + \frac{\overline{m}^{3/2}}{t}\right).$$

Now consider the value of $\overline{m}$ computed in Step 3 of `Estimate`(ϵ). As stated previously, with probability at least $1 - 1/\text{poly}(n)$ (e.g., $1 - 1/n^4$), the estimate $\overline{m}$ is within a constant factor from m. Therefore the expected running time of the algorithm (without the conditioning on the value of $\overline{m}$) is bounded by

$$\left(1 - \frac{1}{n^4}\right) \cdot O^*\left(\frac{n}{t^{1/3}} + \frac{m^{3/2}}{t}\right) + \frac{1}{n^4} \cdot O(n^3) = O^*\left(\frac{n}{t^{1/3}} + \frac{m^{3/2}}{t}\right).$$

Observe that we can always assume that the algorithm does not perform queries it can answer by itself. That is, we can allow the algorithm to save all the information it obtained from past queries, and assume it does not query for information it can deduce from its past queries. Further observe that any pair query is preceded by a neighbor query. Therefore, if at any point the algorithm performs more than $2\overline{m}$ queries, it can abort. It follows that the expected query complexity is $O^*(n/t^{1/3} + \min\{m, m^{3/2}/t\})$. ■

IV. A Lower Bound

In this section we present a lower bound on the number of queries necessary for estimating the number of triangles in a graph. Due to space constraints, in this extended abstract we provide only partial details of the proof.

All details can be found in the full version of this paper [38]. Since we sometimes refer to the number of triangles in different graphs, we use the notation $t(G)$ for the number of triangles in a graph G. Our lower bound matches our upper bound in terms of the dependence on n, m and $t(G)$, up to polylogarithmic factors in n and the dependence

in $1/\epsilon$. In what follows, when we refer to approximation algorithms for the number of triangles in a graph, we mean multiplicative-approximation algorithms that output with high constant probability an estimation $\widehat{t}$ such that $t(G)/C \leq \widehat{t} \leq C \cdot t(G)$ for some predetermined approximation factor C.

We consider multiplicative-approximation algorithms that are allowed the following three types of queries: Degree queries, pair queries and random new-neighbor queries. Degree queries and pair queries are as defined in Section II. A random new-neighbor query q_i is a single vertex u and the corresponding answer is a vertex v such that $(u,v) \in E$ and the edge (u,v) is selected uniformly at random among the edges incident to u that have not yet been observed by the algorithm. It is not hard to verify (as we show in the full version of this paper [38]) that this implies a lower bound when the algorithm may perform (standard) neighbor queries instead of random new-neighbor queries.

We first give a simple lower bound that depends on n and $t(G)$.

Theorem 13: Any multiplicative-approximation algorithm for the number of triangles in a graph must perform $\Omega\left(\frac{n}{t(G)^{1/3}}\right)$ queries, where the allowed queries are degree queries, pair queries and random new-neighbor queries.

Proof: For every n and every $1 \leq t \leq \binom{n}{3}$ we next define a graph G_1 and a family of graphs $\mathcal{G}_2$ for which the following holds. The graph G_1 is the empty graph over n vertices. In $\mathcal{G}_2$, each graph consists of a clique of size $\left\lfloor t^{1/3} \right\rfloor$ and an independent set of size $n - \left\lfloor t^{1/3} \right\rfloor$. See Figure 1 for an illustration. Within $\mathcal{G}_2$ the graphs differ only in the labeling of the vertices. By construction, G_1 contains no triangles and each graph in $\mathcal{G}_2$ contains $\Theta(t)$ triangles. Clearly, unless the algorithm "hits" a vertex in the clique it cannot distinguish between the two cases. The probability of hitting such a vertex in a graph selected uniformly at random from $\mathcal{G}_2$ is $\left\lfloor t^{1/3} \right\rfloor / n$. Thus, in order for this event to occur with high constant probability, $\Omega\left(\frac{n}{t^{1/3}}\right)$ queries are necessary. ∎

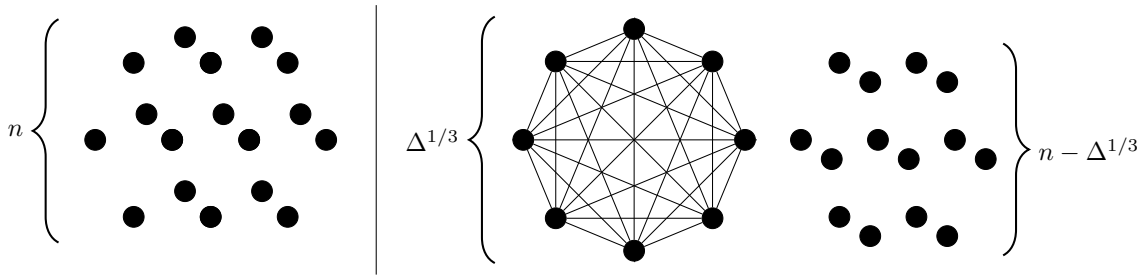


Figure 1. An illustration of the two families.

We next state our main theorem.

Theorem 14: Any multiplicative-approximation algorithm for the number of triangles in a graph must perform at least $\Omega\left(\min\left\{\frac{m^{3/2}}{t(G)}, m\right\}\right)$ queries, where the allowed queries are degree queries, pair queries and random new-neighbor queries.

For every n, every $1 \leq m \leq \binom{n}{2}$ and every $1 \leq t \leq \min\left\{\binom{n}{3}, m^{3/2}\right\}$ we define a graph G_1 and a family of graphs $\mathcal{G}_2$ for which the following holds. The graph G_1 and all the graphs in $\mathcal{G}_2$ have n vertices and m edges. For the graph G_1, $t(G_1) = 0$, and for every graph $G \in \mathcal{G}_2$, $t(G) = \Theta(t)$. We prove it is necessary to perform $\Omega\left(\min\left\{\frac{m^{3/2}}{t}, m\right\}\right)$ queries in order to distinguish with high constant probability between G_1 and a random graph in $\mathcal{G}_2$. For the sake of simplicity, in everything that follows we assume that $\sqrt{m}$ is even.

We prove that for values of t such that $t < \frac{1}{4}\sqrt{m}$, at least $\Omega(m)$ queries are required, and for values of t such that $t \geq \sqrt{m}$ at least $\Omega\left(\frac{m^{3/2}}{t}\right)$ queries are required. For the former case we refer the reader to the full version of this paper [38], and turn to the case that $t \geq \sqrt{m}$. Our construction of $\mathcal{G}_2$ depends on the value of t as a function of m where we deal separately with the following two ranges of t:

1) $t \in [\Omega(m), O(m^{3/2})]$.
2) $t \in [\Omega(\sqrt{m}, O(m)]$.

We prove that for every t as above, $\Omega(m^{3/2}/t)$ queries are needed in order to distinguish between the graph G_1 and a random graph in $\mathcal{G}_2$. Observe that by Proposition 11, for every graph G, it holds that $t(G) = O\left(m^{3/2}\right)$. Hence, the above ranges indeed cover all the possible values of t as a function of m.

A high level discussion of the lower bound: The constructions for the different ranges of $t \geq \sqrt{m}$ are all based on the same basic idea, and have the following in common. In all construction for t as above, G_1 consists of a complete

bipartite graph $(L \cup R, E)$ with $|L| = |R| = \sqrt{m}$ and an independent set of $n - 2\sqrt{m}$ vertices. The basic structure of the graphs in the family $\mathcal{G}_2$ is the same as that of G_1 with the following modifications:

- For every value of t, we add $t/\sqrt{m}$ edges between vertices in L (and similarly in R). Since each edge contributes (roughly) $\sqrt{m}$ triangles, this gives the desired total number of triangles in the graph. In the case that $t = m$ this is done by adding a perfect matching within L and a perfect matching within R. In the case that $t > m$ we add several such perfect matchings, and in the case that $\sqrt{m} \leq t \leq m/4$ we add a (non-perfect) matching of size $t/\sqrt{m}$.
- In order to maintain the degrees of all the vertices in the bipartite component, we remove edges between vertices in L and R.

For an illustration of the case $t = m$, see Figure 2. In what follows we assume that the algorithm knows in advance which vertices are in L and which are in R, and consider only the bipartite component of the graphs. In order to give the intuition for the $m^{3/2}/t$ lower bound we consider each type of query separately, starting with degree queries.

Since both in the graph G_1 and in all the graphs in $\mathcal{G}_2$, all the vertices in $L \cup R$ have the same degree (of $\sqrt{m}$), degree queries do not reveal any information that is useful for distinguishing between the two.

As for pair queries, unless the algorithm queries a pair in $L \times L$ (or $R \times R$) and receives a positive answer, or queries a pair in $L \times R$ and receives a negative answer, the algorithm cannot distinguish between the bipartite component of the graph G_1 and those of the graphs in $\mathcal{G}_2$. We refer to these pairs as *witness pairs*. Roughly speaking, since there are $\Theta(t/\sqrt{m})$ such pairs, and m pairs in total, it takes $\Omega(m^{3/2}/t)$ queries in order to "catch a witness pair".

We are left to deal with neighbor queries. Here too, distinguishing between the graph G_1 and the graphs in $\mathcal{G}_2$ can be done by "catching a witness". That is, if the algorithm queries for a neighbor of a vertex in L and the answer is another vertex in L (analogously for a vertex in R). As before, the probability for hitting such a witness pair is small. However, there is another source of difference resulting from neighbor queries. When the algorithm queries a vertex $v \in L$ there is a difference in the conditional distribution on answers $v \in R$ when the answer is according to the graph G_1 or according to a graph in the family $\mathcal{G}_2$. The reason for the difference, is that in the graph G_1 every vertex has exactly $\sqrt{m}$ neighbors in the opposite side, while for graphs in $\mathcal{G}_2$, each vertex has $\Theta(\sqrt{m} - t/m)$ neighbors in the opposite side (for the range $\Omega(\sqrt{m}) \leq t \leq O(m)$ this is true on average). We prove that this difference in sufficiently small so as to ensure the $\Omega(m^{3/2}/t)$ lower bound.

Our formal analysis is based on defining two processes that interact with an algorithm for approximating the number of triangles, denoted ALG. The first process answer queries according to G_1, and the second process answers queries while constructing a uniformly selected graph in $\mathcal{G}_2$. An interaction between ALG and each of these processes induces a distribution over sequences of queries and answers. We prove that if the number of queries performed by ALG is smaller than $m^{3/2}/(ct)$ for a sufficiently large constant c, then the statistical distance between the two distributions is a small constant.

In this extended abstract we focus on the case that $t = m$ which is a special case of the construction for the range $t \in [\Omega(m), O(m^{3/2})]$. Before doing so we introduce the notion of a *knowledge graph* (as defined previously in e.g., [52]), which will be used in all lower bound proofs. Let ALG be an algorithm for approximating he number of triangles, which performs Q queries. Let q_t denote its t^{th} query and let a_t denote the corresponding answer. Then ALG is a (possibly probabilistic) mapping from *query-answer histories* $\pi \triangleq \langle (q_1, a_1), \ldots, (q_t, a_t) \rangle$ to q_{t+1}, for every $t < Q$, and to $\mathbb{N}$ for $t = Q$.

We assume that the mapping determined by the algorithm is determined only on histories that are consistent with the graph G_1 or one of the graphs in $\mathcal{G}_2$. Any query-answer history π of length t can be used to define a knowledge graph G^{kn}_π at time t. Namely, the vertex set of G^{kn}_π consists of n vertices. For every new-neighbor query u_i answered by v_i for $i \leq t$, the knowledge graph contains the edge (u_i, v_i), and similarly for every pair query (u_j, v_j) that was answered by 1. In addition, for every pair query (u_i, v_i) that is answered by 0, the knowledge graph maintains the information that (u_i, v_i) is a non-edge. The above definition of the knowledge graph is a slight abuse of the notation of a graph since G^{kn}_π is a subgraph of the graph tested by the algorithm, but it also contains additional information regarding queried pairs that are not edges. For a vertex u, we denote its set of neighbors in the knowledge graph by $\Gamma^{kn}_\pi(u)$, and let $d^{kn}_\pi(u) = \left|\Gamma^{kn}_\pi(u)\right|$. We denote by $N^{kn}_\pi(u)$ the set of vertices v such that (u, v) is either an edge or a non-edge in G^{kn}_π.

A. The lower-bound construction

The graph G_1 has two components. The first component is a complete bipartite graph with $\sqrt{m}$ vertices on each side, i.e, $K_{\sqrt{m},\sqrt{m}}$, and the second component is an independent set of size $n - 2\sqrt{m}$. We denote by L the set of

vertices $\ell_1, \ldots, \ell_{\sqrt{m}}$ on the left-hand side of the bipartite component and by R the set of vertices $r_1, \ldots, r_{\sqrt{m}}$ on its right-hand side. The graphs in the family $\mathcal{G}_2$ have the same basic structure with a few modifications. We first choose for each graph a perfect matching M^C between the two sides R and L and remove the edges in M^C from the graph. We refer to the removed matching as the "red matching" and its pairs as "crossing non-edges" or "red pairs". Now, we add two perfect matching from L to L and from R to R, denoted M^L and M^R respectively. We refer to these matchings as the blue matchings and their edges as "non-crossing edges" or "blue pairs". Thus for each choice of three perfect matchings M^C, M^L and M^R as defined above, we have a corresponding graph in $\mathcal{G}_2$.

Consider a graph $G \in \mathcal{G}_2$. Clearly, every blue edge participate in $\sqrt{m} - 2$ triangles. Since, every triangle in the graph contains exactly one blue edge, there are $2\sqrt{m} \cdot (\sqrt{m} - 2) = \Theta(m)$ triangles in G.

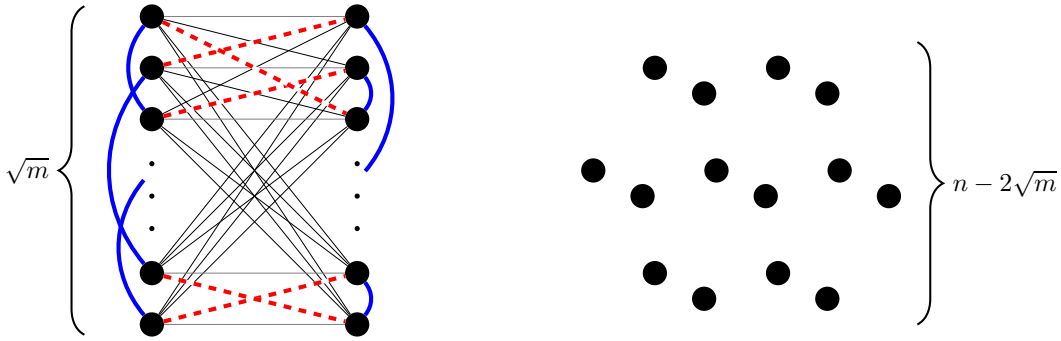


Figure 2. An illustration of the family $\mathcal{G}_2$ for $t = m$.

B. Definition of the processes P_1 and P_2

In what follows we describe two random processes, P_1 and P_2, which interact with an arbitrary algorithm ALG. The process P_1 answers ALG's queries consistently with G_1. The process P_2 answers ALG's queries while constructing a uniformly selected random graph from $\mathcal{G}_2$. We assume without loss of generality that ALG does not ask queries whose answers can be derived from its knowledge graph, since such queries give it no new information. For example, ALG does not ask a pair query about a pair of vertices that are already known to be connected by an edge due to a neighbor query. Also, we assume ALG knows in advance which vertices belong to L and which to to R, so that ALG need not query vertices in the independent set. Since the graphs in $\mathcal{G}_2$ differ from G_1 only in the edges of the subgraph induced by $L \cup R$, we think of G_1 and graphs in $\mathcal{G}_2$ as consisting only of this subgraph. Finally, since in our constructions all the vertices in $L \cup R$ have the same degree of $\sqrt{m}$, we assume that no degree queries are performed.

For every, Q, every $t \leq Q$ and every query-answer history π of length $t-1$ the process P_1 answers the t^{th} query of the algorithm consistently with G_1. Namely:

- For a pair query $q_t = (u, v)$ if the pair (u, v) is a crossing pair in G_1, then the process replies 1, and otherwise it replies 0.
- For a random new-neighbor query $q_t = u$ the process answers with a random neighbor of u that has yet been observed by the algorithm. That is, for every vertex v such that $v \in \Gamma(u) \setminus \Gamma_\pi^{kn}(u)$ the process replies $a_t = v$ with probability $1/(\sqrt{m} - d_\pi^{kn}(u))$.

The process P_2 is defined as follows:

- For a query-answer history π we denote by $\mathcal{G}_2(\pi) \subset \mathcal{G}_2$ the subset of graphs in $\mathcal{G}_2$ that are consistent with π.
- For every $t \leq Q$ and every query-answer history π of length $t-1$, the process P_2 selects a graph in $\mathcal{G}_2$ uniformly at random and answers the t^{th} query as follows.
  1) If the t^{th} query is a pair query $q_t = (u, v)$, then P_2 answers the query q_t according to the selected graph.
  2) If the t^{th} query is a random new-neighbor query $q_t = u_t$, then P_2's answer is a uniform new neighbor of u_t in the selected graph.
- After all queries are answered (i.e., after Q queries), uniformly choose a random graph G from $\mathcal{G}_2(\pi)$.

For a query-answer history π of length Q we denote by $\pi^{\leq t}$ the length t prefix of π and by $\pi^{\geq t}$ the $Q - t + 1$ suffix of π.

We note that the selected graph is only used to answer the t^{th} query and is then "discarded back to" the remaining graphs that are consistent with that answer (and all previous answers in π).

Claim 15: Let π be a query-answer history of length $t-1$. We use $\circ$ to denote concatenation.

- If the t^{th} query is a pair query, then $a_t = 1$ with probability
$$\frac{|\mathcal{G}_2(\pi \circ (q_t, 1))|}{|\mathcal{G}_2(\pi)|},$$
and $a_t = 0$ with probability
$$\frac{|\mathcal{G}_2(\pi \circ (q_t, 0))|}{|\mathcal{G}_2(\pi)|}.$$
- If the t^{th} query is a random new-neighbor query $q_t = u_t$, then for every $v \in V \setminus \Gamma^{kn}_{\pi}(u)$ the probability that the process P_2 answers $a_t = v$ is
$$\frac{|\mathcal{G}_2(\pi \circ (q_t, v))|}{|\mathcal{G}_2(\pi)|} \cdot \frac{1}{\sqrt{m} - d^{kn}_{\pi}(u_t)}.$$
If $v \in \Gamma^{kn}_{\pi}(u)$ then the probability that P_2 answers $a_t = v$ is 0.

Proof: First consider a pair query $q_t = (u_t, v_t)$. The probability that (u_t, v_t) is an edge in the graph chosen by the process P_2 is the fraction of graphs in $\mathcal{G}_2(\pi)$ in which (u_t, v_t) is an edge. This is exactly $\frac{|\mathcal{G}_2(\pi \circ (q_t,1))|}{|\mathcal{G}_2(\pi)|}$. Similarly, the probability of choosing a graph in which (u_t, v_t) is not an edge is $\frac{|\mathcal{G}_2(\pi \circ (q_t,0))|}{|\mathcal{G}_2(\pi)|}$.

Now consider a random new-neighbor query $q_t = u_t$. We start with the case that $v \in V \setminus \Gamma^{kn}_{\pi}$. The probability that v is chosen by P_2 is the probability that a graph G in which v is a neighbor of u_t is chosen in the first step, and that v is the chosen new neighbor among all of u's neighbors in the second step. Since there are $|\mathcal{G}_2(\pi \circ (u, v))|$ graphs in which v is a neighbor of u_t, and u_t has $\sqrt{m} - d^{kn}_{\pi}(u_t)$ neighbors, this happens with probability
$$\frac{|\mathcal{G}_2(\pi \circ (u, v))|}{|\mathcal{G}_2|} \cdot \frac{1}{\sqrt{m} - d^{kn}_{\pi}(u_t)}.$$
For a vertex v such that $v \notin V \setminus \Gamma^{kn}_{\pi}$, in every graph $G \in \mathcal{G}_2$, v is not a neighbor of u_t, implying that the probability that the process replies $a_t = v$ is 0. ∎

Lemma 16: For every algorithm ALG, the process P_2, when interacting with ALG, answers ALG's queries according to a uniformly generated graph G in $\mathcal{G}_2$.

Proof: Consider a specific graph $G \in \mathcal{G}_2$. Let π be the query-answer history generated by the interaction between ALG and P_2. Let Q be the number of queries performed during the interaction. The probability that G is the resulting graph from that interaction is

$$\Pr[G \in \mathcal{G}_2(\pi^{\leq 1})] \cdot \Pr[G \in \mathcal{G}_2(\pi^{\leq 2}) \mid G \in \mathcal{G}_2(\pi^{\leq 1})] \cdot \ \ldots \ \cdot \Pr[G \in \mathcal{G}_2(\pi^{\leq Q}) | G \in \mathcal{G}_2(\pi^{\leq Q-1})] \cdot \frac{1}{|\mathcal{G}(\pi^{\leq Q})|}$$
$$= \frac{|\mathcal{G}_2(\pi^{\leq 1})|}{|\mathcal{G}_2|} \cdot \frac{|\mathcal{G}_2(\pi^{\leq 2})|}{|\mathcal{G}_2(\pi^{\leq 1})|} \cdot \ \ldots \ \cdot \frac{|\mathcal{G}_2(\pi^{\leq Q})|}{|\mathcal{G}_2(\pi^{\leq Q-1})|} \cdot \frac{1}{|\mathcal{G}_2(\pi^{Q})|} = \frac{1}{|\mathcal{G}_2|},$$
and the lemma follows. ∎

For a fixed algorithm ALG that performs Q queries, and for $b \in \{1, 2\}$, let $\mathcal{D}^b_{\text{ALG}}$ denote the distribution on query-answers histories of length Q induced by the interaction between ALG and P_b. We shall show that for every algorithm ALG that performs at most $Q = \frac{m^{3/2}}{100t}$ queries, the statistical distance between $\mathcal{D}^{\text{ALG}}_1$ and $\mathcal{D}^{\text{ALG}}_2$, denoted $d\left(\mathcal{D}^{\text{ALG}}_1, \mathcal{D}^{\text{ALG}}_2\right)$, is at most $\frac{1}{3}$. This will imply that the lower bound stated in Theorem 14 holds for the case that $t(G) = m$. In order to obtain this bound we introduce the notion of a query-answer witness pair, defined next.

Definition 5: We say that ALG has detected a query-answer witness pair in three cases:

1) If q_t is a pair query for a crossing pair $(u_t, v_t) \in L \times R$ and $a_t = 0$.
2) If q_t is a pair query for a non-crossing pair $(u_t, v_t) \in (L \times L) \cup (R \times R)$ and $a_t = 1$.
3) If $q_t = u_t$ is a random new-neighbor query and $a_t = v$ for some v such that (u_t, v) is a non-crossing pair.

We note that the source of the difference between $\mathcal{D}^{\text{ALG}}_1$ and $\mathcal{D}^{\text{ALG}}_2$ is not only due to the probability that the query-answer history contains a witness pair (which is 0 under $\mathcal{D}^{\text{ALG}}_1$ and non-0 under $\mathcal{D}^{\text{ALG}}_2$). There is also a difference in the distribution over answers to random new neighbor queries when the answers do not result in witness pairs (in particular when we condition on the query-answer history prior to the t^{th} query). However, the analysis of witness pairs serves us also in bounding the contribution to the distance due to random new neighbor queries that do not result in a witness pairs.

Let w be a "witness function", such that for a pair query q_t on a crossing pair, $w(q_t) = 0$, and for a non-crossing pair, $w(q_t) = 1$. The probability that ALG detects a witness pair when q_t is a pair query (u_t, v_t) and π is a query-answer history of length $t-1$, is

$$\Pr_{P_2}[w(q_t) \mid \pi] = \frac{|\mathcal{G}_2(\pi \circ (q_t, w(q_t)))|}{|\mathcal{G}_2(\pi)|} \leq \frac{|\mathcal{G}_2(\pi \circ (q_t, w(q_t)))|}{|\mathcal{G}_2(\pi \circ (q_t, \overline{w(q_t)}))|} .$$

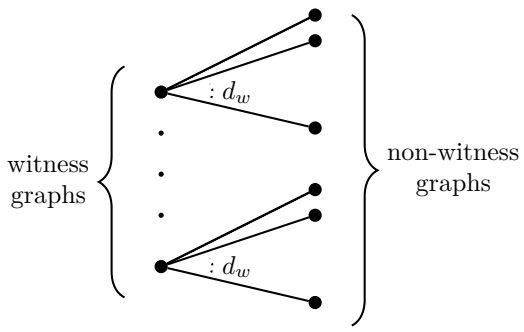
Therefore, to bound the probability that the algorithm observes a witness pair it is sufficient to bound the ratio between the number of graphs in $\mathcal{G}_2(\pi \circ (q, w(q_t)))$ and the number of graphs in $\mathcal{G}_2(\pi \circ (q, \overline{w(q_t)}))$. We do this by introducing an auxiliary graph, which is defined next.

C. The auxiliary graph

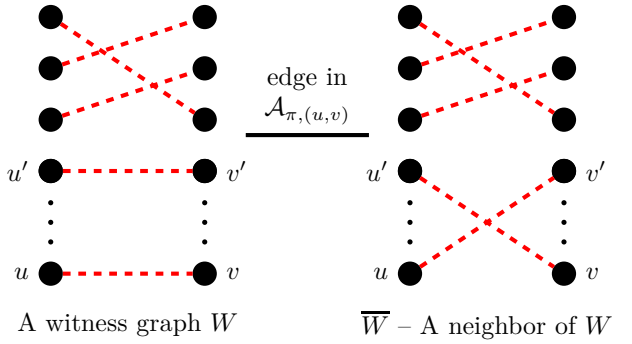
For every $t \leq Q$, every query-answer history π of length $t-1$ for which π is consistent with G_1 (that is, no witness pair has yet been detected), and every pair (u, v), we consider a bipartite auxiliary graph $\mathcal{A}_{\pi,(u,v)}$. On one side of $\mathcal{A}_{\pi,(u,v)}$ we have a node for every graph in $\mathcal{G}_2(\pi)$ for which the pair (u, v) is a witness pair. We refer to these nodes as witness graphs. On the other side of the auxiliary graph, we place a node for every graph in $\mathcal{G}_2(\pi)$ for which the pair is not a witness. We refer to these nodes as non-witness graphs. We put an edge in the auxiliary graph between a witness graph W and a non-witness graph $\overline{W}$ if the pair (u, v) is a crossing (non-crossing) pair and the two graphs are identical except that their red (blue) matchings differ on exactly two pairs – (u, v) and one additional pair. In other words, $\overline{W}$ can be obtained from W by performing a *switch* operation, as defined next.

Definition 6: We define a switch between pairs in a matching in the following manner. Let (u, v) and (u', v') be two matched pairs in a matching M. A switch between (u, v) and (u', v') means removing the edges (u, v) and (u', v') from M and adding to it the edges (u, v') and (u', v).

Note that the switch process maintains the cardinality of the matching. We denote by $d_w(\mathcal{A}_{\pi,(u,v)})$ the minimal degree of any witness graph in $\mathcal{A}_{\pi,(u,v)}$, and by $d_{nw}(\mathcal{A}_{\pi,(u,v)})$ the maximal degree of the non-witness graphs. See Figure 3 for an illustration.



(a) The auxiliary graph with witness nodes on the left and non-witness nodes on the right.



(b) An illustration of two neighbors in the auxiliary graph for $t = m$.

Figure 3.

Lemma 17: Let $t = m$ and $Q = \frac{m^{3/2}}{100t}$. For every $t \leq Q$, every query-answer history π of length $t-1$ such that π is consistent with G_1 and every pair (u, v),

$$\frac{d_{nw}(\mathcal{A}_{\pi,(u,v)})}{d_w(\mathcal{A}_{\pi,(u,v)})} \leq \frac{2}{\sqrt{m}} = \frac{2t}{m^{3/2}}.$$

Proof: Recall that the graphs in $\mathcal{G}_2$ are as defined in Subsection IV-A and illustrated in Figure 2. In the following we consider crossing pairs, as the proof for non-crossing pairs is almost identical. Recall that a crossing pair is a pair (u, v) such that $u \in L$ and $v \in R$ or vise versa. A witness graph W with respect to the pair (u, v) is a graph in which (u, v) is a red pair, i.e., $(u, v) \in M^C$. There is an edge from W to every non-witness graph $\overline{W} \in \mathcal{G}_2(\pi)$ such that $M^C(W)$ and $M^C(\overline{W})$ differ exactly on (u, v) and one additional edge.

Every red pair $(u', v') \in M^C(W)$ creates a potential non-witness graph $\overline{W}_{(u',v')}$ when switched with (u, v) (as defined in Definition 6). However, not all of the these non-witness graphs are in $\mathcal{G}_2(\pi)$. If u' is a neighbor of v in

the knowledge graph G^{kn}_{π}, i.e., $u' \in \Gamma^{kn}_{\pi}(v)$, then $\overline{W}_{(u',v')}$ is not consistent with the knowledge graph, and therefore $\overline{W}_{(u',v')} \notin \mathcal{G}_2(\pi)$. This is also the case for a pair (u', v') such that $v' \in \Gamma^{kn}_{\pi}(u)$. Therefore, only pairs $(u', v') \in M^C$ such that $u' \notin \Gamma^{kn}_{\pi}(v)$ and $v' \notin \Gamma^{kn}_{\pi}(u)$ produce a non-witness graph $\overline{W}_{(u',v')} \in \mathcal{G}_2(\pi)$ when switched with (u, v). We refer to these pairs as consistent pairs. Since $t \leq \frac{\sqrt{m}}{100}$, both u and v each have at most $\frac{m}{100}$ neighbors in the knowledge graph, implying that out of the $\sqrt{m} - 1$ potential pairs, the number of consistent pairs is at least

$$\sqrt{m} - 1 - d^{kn}_{\pi}(u) - d^{kn}_{\pi}(v) \geq \sqrt{m} - 1 - 2 \cdot \frac{\sqrt{m}}{100} \geq \frac{1}{2}\sqrt{m}.$$

Therefore, the degree of every witness graph $W \in \mathcal{A}_{\pi,(u,v)}$ is at least $\frac{1}{2}\sqrt{m}$, implying that $d_w(\mathcal{A}_{\pi,(u,v)}) \geq \frac{1}{2}\sqrt{m}$.

In order to prove that $d_{nw}(\mathcal{A}_{\pi,(u,v)}) = 1$, consider a non-witness graph $\overline{W}$. Since $\overline{W}$ is a non-witness graph, the pair (u, v) is not a red pair. This implies that u is matched to some vertex $v' \in R$, and v is matched to some vertex $u' \in L$. That is, $(u, v'), (v, u') \in M^C$. By the construction of the edges in the auxiliary graph, every neighbor W of $\overline{W}$ can be obtained by a single switch between two red pairs in the red matching. The only possibility to switch two pairs in $M^C(\overline{W})$ and obtain a matching in which (u, v) is a red pair is to switch the pairs (u, v') and (v, u'). Hence, every non-witness graph $\overline{W}$ has at most one neighbor.

We showed that $d_w(\mathcal{A}_{\pi,(u,v)}) \geq \frac{1}{2}\sqrt{m}$ and that $d_{nw}(\mathcal{A}_{\pi,(u,v)}) \leq 1$, implying

$$\frac{d_{nw}(\mathcal{A}_{\pi,(u,v)})}{d_w(\mathcal{A}_{\pi,(u,v)})} \leq \frac{2}{\sqrt{m}} = \frac{2t}{m^{3/2}},$$

and the proof is complete. ■

D. Statistical distance

For a query-answer history π of length $t - 1$ and a query q_t, let $Ans(\pi, q_t)$ denote the set of possible answers to the query q_t that are consistent with π. Namely, if q_t is a pair query (for a pair that does not belong to the knowledge graph G^{kn}_{π}), then $Ans(\pi, q_t) = \{0, 1\}$, and if q_t is a random new-neighbor query, then $Ans(\pi, q_t)$ consists of all vertices except those in N^{kn}_{π}. For the proofs of the next lemma we refer the reader to the full version of this paper [38].

Lemma 18: Let $t = m$ and $Q = \frac{m^{3/2}}{100t}$. For every $t \leq Q$, every query-answer history π of length $t - 1$ such that π is consistent with G_1 and for every query q_t:

$$\sum_{a \in Ans(\pi,q_t)} \left| \Pr_{P_1}[a \mid \pi, q_t] - \Pr_{P_2}[a \mid \pi, q_t] \right| \leq \frac{12}{\sqrt{m}} = \frac{12t}{m^{3/2}}.$$

Recall that $\mathcal{D}^{\mathrm{ALG}}_b$, $b \in \{1, 2\}$, denotes the distribution on query-answer histories of length Q, induced by the interaction of ALG and P_b. We show that the two distributions are indistinguishable for Q that is sufficiently small.

Lemma 19: Let $t = m$. For every algorithm ALG that asks at most $Q = \frac{m^{3/2}}{100t}$ queries, the statistical distance between $\mathcal{D}^{\mathrm{ALG}}_1$ and $\mathcal{D}^{\mathrm{ALG}}_2$ is at most $\frac{1}{3}$.

Proof: Consider the following hybrid distribution. Let $\mathcal{D}^{\mathrm{ALG}}_{1,t}$ be the distribution over query-answer histories of length Q, where in the length t prefix ALG is answered by the process P_1 and in the length $Q - t$ suffix ALG is answered by the process P_2. Observe that $\mathcal{D}^{\mathrm{ALG}}_{1,Q} = \mathcal{D}^{\mathrm{ALG}}_1$ and that $\mathcal{D}^{\mathrm{ALG}}_{1,0} = \mathcal{D}^{\mathrm{ALG}}_2$. Let $\pi = (\pi_1, \pi_2, \ldots, \pi_\ell)$ denote a query-answer history of length ℓ. By the triangle inequality

It thus remains to bound $d(\mathcal{D}^{\mathrm{ALG}}_{1,t+1}, \mathcal{D}^{\mathrm{ALG}}_{1,t}) = \frac{1}{2}\sum_{\pi} \left| \Pr_{\mathcal{D}^{\mathrm{ALG}}_{1,t+1}}[\pi] - \Pr_{\mathcal{D}^{\mathrm{ALG}}_{1,t}}[\pi] \right|$ for every t such that $0 \leq t \leq Q - 1$. Let $\mathcal{Q}$ denote the set of all possible queries.

$$\begin{aligned}
\sum_{\pi} \left| \Pr_{\mathcal{D}^{\mathrm{ALG}}_{1,t+1}}[\pi] - \Pr_{\mathcal{D}^{\mathrm{ALG}}_{1,t}}[\pi] \right| = & \sum_{\pi_1,\ldots,\pi_{t-1}} \Pr_{P_1,\mathrm{ALG}}[\pi_1, \ldots, \pi_{t-1}] \cdot \sum_{q \in \mathcal{Q}} \Pr_{\mathrm{ALG}}[q \mid \pi_1, \ldots, \pi_{t-1}] \\
& \cdot \sum_{a \in Ans((\pi_1,\ldots,\pi_{t-1}),q)} \left| \Pr_{P_1}[a \mid \pi_1, \ldots, \pi_{t-1}, q] - \Pr_{P_2}[a \mid \pi_1, \ldots, \pi_{t-1}, q] \right| \\
& \cdot \sum_{\pi_{t+1},\ldots,\pi_Q} \Pr_{P_2,\mathrm{ALG}}[\pi_{t+1}, \ldots, \pi_Q \mid \pi_1, \ldots, \pi_{t-1}, (q, a)] \,.
\end{aligned}$$

By Lemma 18, for every $1 \leq t \leq Q-1$, and every $\pi_1, \ldots, \pi_{t-1}$ and q,

$$\sum_{a \in Ans((\pi_1,\ldots,\pi_{t-1}),q)} \left| \Pr_{P_1}[a \mid \pi_1, \ldots, \pi_{t-1}, q] - \Pr_{P_2}[a \mid \pi_1, \ldots, \pi_{t-1}, q] \right| \leq \frac{12t}{m^{3/2}} .$$

We also have that for every pair (q, a),

$$\sum_{\pi_{t+1},\ldots,\pi_Q} \Pr_{P_2,\mathrm{ALG}}[\pi_{t+1}, \ldots, \pi_Q \mid \pi_1, \ldots, \pi_{t-1}, (q, a)] = 1 .$$

Therefore,

$$\sum_{\pi} \left| \Pr_{\mathcal{D}^{\mathrm{ALG}}_{1,t+1}}[\pi] - \Pr_{\mathcal{D}^{\mathrm{ALG}}_{1,t}}[\pi] \right| \leq \sum_{\pi_1,\ldots,\pi_{t-1}} \Pr_{P_1,\mathrm{ALG}}[\pi_1, \ldots, \pi_{t-1}] \sum_{q \in \mathcal{Q}} \Pr_{\mathrm{ALG}}[q \mid \pi_1, \ldots, \pi_{t-1}] \cdot \frac{12t}{m^{3/2}} = \frac{12t}{m^{3/2}} .$$

Hence, for $Q = \frac{\sqrt{m}}{100}$,

$$d(\mathcal{D}^{\mathrm{ALG}}_1, \mathcal{D}^{\mathrm{ALG}}_2) = \frac{1}{2} \sum_{\pi} \sum_{t=1}^{Q-1} \left| \Pr_{\mathcal{D}^{\mathrm{ALG}}_{1,t+1}}[\pi] - \Pr_{\mathcal{D}^{\mathrm{ALG}}_{1,t}}[\pi] \right| \leq \frac{1}{2} \cdot Q \cdot \frac{12t}{m^{3/2}} \leq \frac{1}{3},$$

and the proof is complete. ■

The case of $t = m$ in Theorem 14 follows from Lemma 19.

Acknowledgments

D.R. acknowledges the support of the Israel Science Foundation grant No. 671/13 and the Blavatnik fund.

References

[1] P. W. Holland and S. Leinhardt, "A method for detecting structure in sociometric data," *American Journal of Sociology*, vol. 76, pp. 492–513, 1970.

[2] J. S. Coleman, "Social capital in the creation of human capital," *American Journal of Sociology*, vol. 94, pp. S95–S120, 1988. [Online]. Available: http://www.jstor.org/stable/2780243

[3] A. Portes, "Social capital: Its origins and applications in modern sociology," *LESSER, Eric L. Knowledge and Social Capital. Boston: Butterworth-Heinemann*, pp. 43–67, 2000.

[4] J.-P. Eckmann and E. Moses, "Curvature of co-links uncovers hidden thematic layers in the World Wide Web," *Proceedings of the National Academy of Sciences (PNAS)*, vol. 99, no. 9, pp. 5825–5829, 2002.

[5] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[6] R. S. Burt, "Structural holes and good ideas," *American Journal of Sociology*, vol. 110, no. 2, pp. 349–399, 2004. [Online]. Available: http://www.jstor.org/stable/10.1086/421787

[7] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis, "Efficient semi-streaming algorithms for local triangle counting in massive graphs," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 16–24.

[8] B. Foucault Welles, A. Van Devender, and N. Contractor, "Is a friend a friend?: Investigating the structure of friendship networks in virtual worlds," in *CHI'10 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2010, pp. 4027–4032.

[9] J. W. Berry, B. Hendrickson, R. A. LaViolette, and C. A. Phillips, "Tolerating the Community Detection Resolution Limit with Edge Weighting," *Physical Review E*, vol. 83, no. 5, May 2011.

[10] C. Seshadhri, T. G. Kolda, and A. Pinar, "Community structure and scale-free collections of Erdös-Rényi graphs," *Physical Review E*, vol. 85, no. 5, p. 056109, May 2012.

[11] A. Itai and M. Rodeh, "Finding a minimum circuit in a graph," *SIAM Journal on Computing*, vol. 7, no. 4, pp. 413–423, 1978.

[12] N. Alon, R. Yuster, and U. Zwick, "Finding and counting given length cycles," *Algorithmica*, vol. 17, no. 3, pp. 209–223, 1997.

[13] A. Björklund, R. Pagh, V. V. Williams, and U. Zwick, "Listing triangles," in *Proceedings of International Colloquium on Automata, Languages, and Programming (ICALP)*, 2014, pp. 223–234.

[14] N. Chiba and T. Nishizeki, "Arboricity and subgraph listing algorithms," *SIAM Journal on Computing*, vol. 14, no. 1, pp. 210–223, 1985.

[15] T. Schank and D. Wagner, "Finding, counting and listing all triangles in large graphs, an experimental study," in *Experimental and Efficient Algorithms*. Springer Berlin / Heidelberg, 2005, pp. 606–609.

[16] ——, "Approximating clustering coefficient and transitivity," *Journal of Graph Algorithms and Applications*, vol. 9, pp. 265–275, 2005.

[17] C. E. Tsourakakis, "Fast counting of triangles in large real networks without counting: Algorithms and laws," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 608–617.

[18] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos, "Doulion: counting triangles in massive graphs with a coin," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 837–846.

[19] H. Avron, "Counting triangles in large graphs using randomized matrix trace estimation," in *Workshop on Large-scale Data Mining: Theory and Applications*, vol. 1, 2010.

[20] M. N. Kolountzakis, G. L. Miller, R. Peng, and C. E. Tsourakakis, "Efficient triangle counting in large graphs via degree-based vertex partitioning," *Internet Mathematics*, vol. 8, no. 1-2, pp. 161–185, 2012.

[21] S. Chu and J. Cheng, "Triangle listing in massive networks and its applications," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 672–680.

[22] S. Suri and S. Vassilvitskii, "Counting triangles and the curse of the last reducer," in *World Wide Web (WWW)*, 2011, pp. 607–614.

[23] C. E. Tsourakakis, M. N. Kolountzakis, and G. L. Miller, "Triangle sparsifiers." *J. Graph Algorithms Appl.*, vol. 15, no. 6, pp. 703–726, 2011.

[24] S. Arifuzzaman, M. Khan, and M. Marathe, "Patric: A parallel algorithm for counting triangles in massive networks," in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 2013, pp. 529–538.

[25] C. Seshadhri, A. Pinar, and T. G. Kolda, "Fast triangle counting through wedge sampling," in *Proceedings of the SIAM Conference on Data Mining*, 2013. [Online]. Available: http://arxiv.org/abs/1202.5230

[26] K. Tangwongsan, A. Pavan, and S. Tirthapura, "Parallel triangle counting in massive streaming graphs," in *ACM Conference on Information & Knowledge Management (CIKM)*, 2013.

[27] Z. Bar-Yossef, R. Kumar, and D. Sivakumar, "Reductions in streaming algorithms, with an application to counting triangles in graphs," in *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, 2002, pp. 623–632.

[28] H. Jowhari and M. Ghodsi, "New streaming algorithms for counting triangles in graphs," in *Computing and Combinatorics*. Springer, 2005, pp. 710–716.

[29] L. S. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, and C. Sohler, "Counting triangles in data streams," in *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2006, pp. 253–262.

[30] K. J. Ahn, S. Guha, and A. McGregor, "Graph sketches: sparsification, spanners, and subgraphs," in *Principles of Database Systems*, 2012, pp. 5–14.

[31] D. M. Kane, K. Mehlhorn, T. Sauerwald, and H. Sun, "Counting arbitrary subgraphs in data streams," in *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2012, pp. 598–609.

[32] M. Jha, C. Seshadhri, and A. Pinar, "A space efficient streaming algorithm for triangle counting using the birthday paradox," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '13. New York, NY, USA: ACM, 2013, pp. 589–597. [Online]. Available: http://doi.acm.org/10.1145/2487575.2487678

[33] A. Pavan, K. Tangwongsan, S. Tirthapura, and K.-L. Wu, "Counting and sampling triangles from a graph stream," in *International Conference on Very Large Databases (VLDB)*, 2013.

[34] N. K. Ahmed, N. Duffield, J. Neville, and R. Kompella, "Graph sample and hold: A framework for big graph analytics," in *Conference on Knowledge Discovery and Data Mining (KDD)*, 2014.

[35] M. Gonen, D. Ron, and Y. Shavitt, "Counting stars and other small subgraphs in sublinear-time," *SIAM Journal on Discrete Math*, vol. 25, no. 3, pp. 1365–1411, 2011.

[36] U. Feige, "On sums of independent random variables with unbounded variance and estimating the average degree in a graph," *SIAM Journal on Computing*, vol. 35, no. 4, pp. 964–984, 2006.

[37] O. Goldreich and D. Ron, "Approximating average parameters of graphs," *Random Structures and Algorithms*, vol. 32, no. 4, pp. 473–493, 2008.

[38] T. Eden, A. Levi, D. Ron, and C. Seshadhri, "Approximately counting triangles in sublinear time," *CoRR*, vol. abs/1504.00954v2, 2015. [Online]. Available: http://arxiv.org/abs/1504.00954

[39] B. Chazelle, R. Rubinfeld, and L. Trevisan, "Approximating the minimum spanning tree weight in sublinear time," *SIAM Journal on Computing*, vol. 34, no. 6, pp. 1370–1379, 2005.

[40] A. Czumaj and C. Sohler, "Estimating the weight of metric minimum spanning trees in sublinear time," *SIAM Journal on Computing*, vol. 39, no. 3, pp. 904–922, 2009.

[41] A. Czumaj, F. Ergün, L. Fortnow, A. Magen, I. Newman, R. Rubinfeld, and C. Sohler, "Approximating the weight of the euclidean minimum spanning tree in sublinear time," *SIAM Journal on Computing*, vol. 35, no. 1, pp. 91–109, 2005.

[42] H. N. Nguyen and K. Onak, "Constant-time approximation algorithms via local improvements," in *Proceedings of the Forty-Ninth Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2008, pp. 327–336.

[43] Y. Yoshida, M. Yamamoto, and H. Ito, "An improved constant-time approximation algorithm for maximum," in *Proceedings of the Fourty-First Annual ACM Symposium on the Theory of Computing*. ACM, 2009, pp. 225–234.

[44] M. Parnas and D. Ron, "Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms," *Theoretical Computer Science*, vol. 381, no. 1-3, pp. 183–196, 2007.

[45] S. Marko and D. Ron, "Approximating the distance to properties in bounded-degree and general sparse graphs," *ACM Transactions on Algorithms*, vol. 5, no. 2, 2009.

[46] A. Hassidim, J. A. Kelner, H. N. Nguyen, and K. Onak, "Local graph partitions for approximation and testing," in *Proceedings of the Fiftieth Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2009, pp. 22–31.

[47] K. Onak, D. Ron, M. Rosen, and R. Rubinfeld, "A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size," in *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2012, pp. 1123–1131.

[48] C. E. Tsourakakis, P. Drineas, E. Michelakis, I. Koutis, and C. Faloutsos, "Spectral counting of triangles in power-law networks via element-wise sparsification," in *Social Network Analysis and Mining, 2009. ASONAM'09. International Conference on Advances in*. IEEE, 2009, pp. 66–71.

[49] R. Pagh and C. Tsourakakis, "Colorful triangle counting and a mapreduce implementation," *Information Processing Letters*, vol. 112, pp. 277–281, 2012.

[50] J. Cohen, "Graph twiddling in a MapReduce world," *Computing in Science & Engineering*, vol. 11, pp. 29–41, 2009.

[51] T. G. Kolda, A. Pinar, T. Plantenga, C. Seshadhri, and C. Task, "Counting triangles in massive graphs with MapReduce," arXiv:1301.5887, January 2013.

[52] O. Goldreich and D. Ron, "Property testing in bounded degree graphs," *Algorithmica*, pp. 302–343, 2002.