Counting Stars and Other Small Subgraphs in Sublinear Time

Mira Gonen* Dana Ron† Yuval Shavitt‡

Abstract

Detecting and counting the number of copies of certain subgraphs (also known as *network motifs* or *graphlets*), is motivated by applications in a variety of areas ranging from Biology to the study of the World-Wide-Web. Several polynomial-time algorithms have been suggested for counting or detecting the number of occurrences of certain network motifs. However, a need for more efficient algorithms arises when the input graph is very large, as is indeed the case in many applications of motif counting.

In this paper we design *sublinear-time* algorithms for approximating the number of copies of certain constant-size subgraphs in a graph G. That is, our algorithms do not read the whole graph, but rather query parts of the graph. Specifically, we consider algorithms that may query the degree of any vertex of their choice and may ask for any neighbor of any vertex of their choice. The main focus of this work is on the basic problem of counting the number of length-2 paths and more generally on counting the number of stars of a certain size. Specifically, we design an algorithm that, given an approximation parameter $0 < \epsilon < 1$ and query access to a graph G, outputs an estimate $\hat{\nu}_s$ such that with high constant probability, $(1-\epsilon)\nu_s(G) \leq \hat{\nu}_s \leq (1+\epsilon)\nu_s(G)$, where $\nu_s(G)$ denotes the number of stars of size $s+1$ in the graph. The expected query complexity and running time of the algorithm are $O\left(\frac{n}{(\nu_s(G))^{\frac{1}{s+1}}} + \min\left\{n^{1-\frac{1}{s}}, \frac{n^{s-\frac{1}{s}}}{(\nu_s(G))^{1-\frac{1}{s}}}\right\}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon)$. We also prove lower bounds showing that this algorithm is tight up to polylogarithmic factors in n and the dependence on ϵ.

Our work extends the work of Feige (*SIAM Journal on Computing, 2006*) and Goldreich and Ron (*Random Structures and Algorithms, 2008*) on approximating the number of edges (or average degree) in a graph. Combined with these results, our result can be used to obtain an estimate on the variance of the degrees in the graph and corresponding higher moments.

In addition, we give some (negative) results on approximating the number of triangles and on approximating the number of length-3-paths in sublinear time.

1 Introduction

This work is concerned with approximating the number of copies of certain constant-size subgraphs in a graph G. Detecting and counting subgraphs (also known as *network motifs* [28] or *graphlets* [31]), is motivated by applications in a variety of areas ranging from Biology to the study of the World-Wide-Web (see e.g., [28, 24, 32, 31, 35, 33, 20, 13, 22, 1, 21, 19]), as well as by the basic quest to understand simple structural properties of graphs. Our work differs from previous works on counting subgraphs (with the exception of counting the number of edges [15, 17]) in that we design *sublinear* algorithms. That is, our algorithms do not read the whole graph, but rather query parts of the graph (where we shall specify the type of queries we allow when we state our precise results). The need for such algorithms arises when the input graph is very large (as is indeed the case in many of the application of motif counting).

The main focus of this work is on the problem of counting the number of length-2 paths and more generally on counting the number of stars of a certain size. We emphasize that we count *non-induced* subgraphs. We shall use the term *s-star* for a subgraph over $s+1$ vertices in which one single vertex (the star *center*) is adjacent to all other vertices (and there are no edges between the other vertices). Observe that a length-2 path is a 2-star. We also give some (negative) results on approximating the number of triangles and on approximating the number of length-3-paths.

As we show in detail below, we obtain almost matching upper and lower bounds on the query complexity and running time of approximating the number of s-stars. These bounds are a function of the number, n, of graph vertices and the actual number of s-stars in the graph, and have a non-trivial form. Our results extend the works [15] and [17] on sublinear-time approximation of the average degree in a graph, or equivalently, approximating the number of edges (where an edge is the simplest (non-empty) subgraph). Note that if we have an estimate for the number of length-2 paths and for the average degree, then we can obtain an estimate for the variance of the degrees in the graph, and the number of larger stars corresponds to higher moments. Thus, the study of the frequencies of these particular subgraphs in a graph sheds light on basic structural properties of graphs.

Our Results. We assume graphs are represented by the incidence lists of the vertices (or, more precisely, incidence arrays), where each list is accompanied by its length. Thus, the algorithm can query the degree, $d(v)$, of any vertex v of its choice and for any vertex v and

*School of Electrical Engineering, Tel-Aviv University, gonenmir@post.tau.ac.il

†School of Electrical Engineering, Tel-Aviv University, danar@eng.tau.ac.il. Research supported by the Israel Science Foundation (grant No. 246/08)

‡School of Electrical Engineering, Tel-Aviv University, shavitt@eng.tau.ac.il

index $1 \leq i \leq d(v)$ it can query who is the i^{th} neighbor of v.

Let $\nu_s(G)$ denote the number of s-stars in a graph G. Our main positive result is an algorithm that, given an approximation parameter $0 < \epsilon < 1$ and query access to a graph G, outputs an estimate $\hat{\nu}_s$ such that with high constant probability (over the coin flips of the algorithm), $(1-\epsilon)\nu_s(G) \leq \hat{\nu}_s \leq (1+\epsilon)\nu_s(G)$. The expected query complexity and running time of the algorithm are:

$$O\left(\frac{n}{(\nu_s(G))^{\frac{1}{s+1}}} + \min\left\{n^{1-\frac{1}{s}}, \frac{n^{s-\frac{1}{s}}}{(\nu_s(G))^{1-\frac{1}{s}}}\right\}\right) \cdot \text{poly}(\log n, 1/\epsilon)\,. \tag{1.1}$$

The dependence on s is exponential, and is not stated explicitly as we assume s is a constant. The complexity of our algorithm as stated in Equation (1.1) is best understood by viewing Table 1, in which we see that there are three regions when considering $\nu_s(G)$ as a function of n, and in each the complexity is governed by a different term. Observe that in the first range

$\boldsymbol{\nu_s(G)}$	**Complexity**
$\nu_s(G) \leq n^{s-\frac{1}{s}}$	$O\left(\frac{n}{(\nu_s(G))^{\frac{1}{s+1}}}\right) \cdot \text{poly}\left(\frac{\log n}{\epsilon}\right)$
$n^{s-\frac{1}{s}} < \nu_s(G) \leq n^s$	$O\left(n^{1-\frac{1}{s}}\right) \cdot \text{poly}\left(\frac{\log n}{\epsilon}\right)$
$\nu_s(G) > n^s$	$O\left(\frac{n^{s-\frac{1}{s}}}{(\nu_s(G))^{1-\frac{1}{s}}}\right) \cdot \text{poly}\left(\frac{\log n}{\epsilon}\right)$

Table 1: The query complexity and running time of our algorithm for approximating the number of s-stars.

($\nu_s(G) \leq n^{s-\frac{1}{s}}$) the complexity of the algorithm (which is at its maximum when $\nu_s(G)$ is very small) decreases as $\nu_s(G)$ increases; in the second range ($n^{s-\frac{1}{s}} < \nu_s(G) \leq n^s$) the complexity does not depend on $\nu_s(G)$; and in the last range ($\nu_s(G) > n^s$) it again decreases as $\nu_s(G)$ increases (where in the extreme case, when $\nu_s(G) = \Omega(n^{s+1})$ the complexity is just $\text{poly}(\log n, 1/\epsilon)$). We note that if one is willing to allow an additive error, that is, the requirement is that the estimate $\hat{\nu}_s$ satisfy (with high probability) $\nu_s(G) - \alpha \leq \hat{\nu}_s \leq \nu_s(G) + \alpha$ for α that is larger than $\epsilon\nu_s(G)$ (for $0 < \epsilon < 1$), then this can be obtained by a slight adaptation of our algorithm. The complexity of the resulting algorithm is as in the expression in Equation (1.1) where $\nu_s(G)$ is replaced by α (and there is no dependence on $1/\epsilon$).

The expression in Equation (1.1) might seem unnatural and hence merely an artifact of our algorithm. However, we prove that it is tight up to polylogarithmic factors in n and the dependence on ϵ. Namely, we show that:

- Any multiplicative approximation algorithm for the number of s-stars must perform $\Omega\left(\frac{n}{(\nu_s(G))^{\frac{1}{s+1}}}\right)$ queries.
- Any constant-factor approximation algorithm for the number of s-stars must perform $\Omega(n^{1-\frac{1}{s}})$ queries when the number of s-stars is $O(n^s)$.
- Any constant-factor approximation algorithm for the number of s-stars must perform $\Omega\left(\frac{n^{s-\frac{1}{s}}}{(\nu_s(G))^{1-\frac{1}{s}}}\right)$ queries when the number of s-stars is $\Omega(n^s)$.

We mention that another type of queries, which are natural in the context of dense graphs, are vertex-pair queries. That is, the algorithm may query about the existence of an edge between any pair of vertices. We note that our lower bounds imply that allowing such queries cannot reduce the complexity for counting the number of stars (except possibly by polylogarithmic factors in n).

We also consider other small graphs that extend length-2 paths: triangles, and length-3 paths. We show that if an algorithm uses a number of queries that is sublinear in the number of edges, then for triangles it is hard to distinguish between the case that a graph contains $\Theta(n)$ triangles and the case that it contains *no* triangles, and for length-3 paths it is hard to distinguish between the case that there are $\Theta(n^2)$ length-3 paths and the case that there are no such paths. These lower bounds hold when the number of edges is $\Theta(n)$.[1]

Techniques. Our starting point is similar to the one of [17]. Consider a partition of the graph vertices into $O(\log n/\epsilon)$ *buckets* where in each bucket all vertices have the same degree (with respect to the entire graph) up to a multiplicative factor of $(1 \pm O(\epsilon))$. If we could get a good estimate of the size of each bucket by sampling, then we would have a good estimate of the number of s-stars (since the vertices in each bucket are the centers of approximately the same number of stars). The difficulty is that some buckets may be very small and we might not even hit them when sampling vertices. The approach taken in [17] to get a multiplicative estimate of $(1 \pm \epsilon)$ is to estimate the number of edges between large buckets and small buckets, and incorporate this estimate in the final approximation.[2]

[1] We mention that we are currently studying these problems in an extended query model that also allows vertex-pair queries. For length-3 paths, even if we allow such queries then there is a lower bound that is linear in the number of edges when the number of edges is $\Theta(n)$, and for triangles there is a lower bound that is linear in n when the number of edges is $\Theta(n^2)$.

[2] We note that in the case of the average degree (number of edges), if we ignore the small buckets (for an appropriate definition of "small") then we can already get (roughly) a factor-

Here we first observe that we need a more refined procedure. In particular, we need a separate estimate for the number of edges between each large bucket and each small bucket. Note that if we have an estimate $\hat{e}$ on the number of edges incident to vertices in a certain bucket, and all vertices in that bucket have degree roughly d, then the number of s-stars whose center belongs to this bucket is approximately[3] $\frac{1}{s}\hat{e} \cdot \binom{d-1}{s-1}$. As a first attempt for obtaining such an estimate on the number of edges incident to vertices in a bucket, consider uniformly sampling edges incident to vertices that belong to large buckets. We can then estimate the number of edges between the large buckets and each small bucket by querying the degree of the other end point of each sampled edge. It is possible to show that for a sufficiently large sample of edges we can indeed obtain a good estimate for the number of s-stars using this procedure. However, the complexity of the resulting procedure, which is dominated by the number of edges that need to be sampled, is far from optimal. The reason for this has to do with the variance between the number of edges that different vertices in the same large bucket have to the various small buckets. To overcome this and get an (almost) optimal algorithm, we further refine the sampling process.

Specifically, we first define the notion of *significant* small buckets. Such buckets have a non-negligble contribution to the total number of s-stars (where each vertex accounts for the number of stars that it is a center of). Now, for each large bucket B_i and (significant) small bucket B_j we further consider partitioning the vertices in B_i according to the number of neighbors they have in B_j. The difficulty is that in order to determine *exactly* to which sub-bucket a vertex in B_i belongs to, we would need to query all its neighbors, which may be too costly. Moreover, even if an estimate on this number suffices, if a vertex in B_i has relatively few neighbors in B_j then we would need a relatively large sample of its neighbors in order to obtain such an estimate. Fortunately, we encounter a tradeoff between the number of vertices in B_i that need to be sampled in order to get sufficiently many vertices that belong to a particular sub-bucket and the number of neighbors that should be sampled so as to detect (approximately) to which sub-bucket a vertex belongs to. We exemplify this by an extreme case: consider the sub-bucket of vertices for which at least half of their neighbors belong to B_j. This sub-bucket may be relatively small (and still contribute significantly to the total number of edges between B_i and B_j) but if we sample a vertex from this sub-bucket then we can easily detect this by taking only constant sample of its neighbors. For more details see Subsection 2.4.

Related Work. As noted previously, our work extends the works [15, 17] on approximating the average degree of a graph in sublinear time. In particular, our work is most closely related to [17] where it is shown how to get an estimate of the average degree of a graph G that is within $(1 \pm \epsilon)$ of the correct value $\bar{d}(G)$. The expected running time and query complexity of the algorithm in [17] are $O((n/\bar{d}(G))^{1/2}) \cdot \mathrm{poly}(\log n, 1/\epsilon)$.

There are quite a few works that deal with finding subgraphs of a certain kind and of counting their number in polynomial time. One of the most elegant techniques devised is *color-coding*, introduced in [4], and further applied in [5, 7, 2, 1, 3]. In particular, in [7] the authors use color-coding and a technique from [23] to design a randomized algorithm for approximately counting the number of subgraphs in a given graph G which are isomorphic to a bounded treewidth graph H. The running time of the algorithm is $k^{O(k)} \cdot n^{b+O(1)}$, where n and k are the number of vertices in G and H, respectively, and b is the treewidth of H. In [2] the authors use color-coding and balanced families of perfect hash functions to obtain a deterministic algorithm for counting simple paths or cycles of size k in time $2^{O(k \log\log k)} n^{O(1)}$. In [1] these results are improved in terms of the dependence on k. We note that sampling is also applied in [24, 35], where the authors are interested in uniformly sampling induced subgraphs of a given size k. Other related work in this category include [14, 20, 8, 26, 36, 19, 9, 6, 27, 34].

Another related line of work deals with approximating other graph measures (such as the weight of a minimum spanning tree) in sublinear time and includes [10, 12, 11, 30, 29].

Organization. For the sake of the exposition we first describe the algorithm and the analysis, as well as the lower bounds, for the case $s = 2$, that is, length-2 paths. This is done in Sections 2 and 3, respectively. In Section 4 we explain how to adapt the algorithm for length-2 paths in order to get an algorithm for s-stars, and in Section 5 we explain how to adapt the

2 approximation in $O(\sqrt{n})$ time [15, 17]. However, this is not the case for s-stars (even when $s = 2$). To verify this consider the case that the graph G is a star. There are two buckets: one containing only the star center, and another containing all other vertices. If we ignore the (very) small bucket that contains the star center then we get an estimate of 0 while the graph contains $\Theta(n^2)$ length-2 paths (2-stars).

[3] To see why this is true, consider an edge (u, v) that is incident to a vertex u that has degree (roughly) d. Then the number of stars that include this edge and are centered at u is (roughly) $\binom{d-1}{s-1}$. If we sum this expression over all $\hat{e}$ edges that are incident to vertices in the bucket of u, then each star (that is centered at a vertex in the bucket) is counted s time, and hence we divide the expression $\hat{e} \cdot \binom{d-1}{s-1}$ by s.

lower bounds. Finally, in Section 6 we shortly discuss triangles and length-3 paths. All missing details can be found in the full version of this paper [18].

2 An Algorithm for Approximating the Number of Length-2 Paths

In this section we describe and analyze an algorithm for estimating the number of length-2 paths (2-stars) in a graph G, where we denote this number by $\ell(G)$. In all that follows we consider undirected simple graphs. We denote the set of neighbors of a vertex v by $\Gamma(v)$ and its degree by $d(v)$. For two (not necessarily disjoint) sets of vertices $V_1, V_2 \subseteq V$ we let $E(V_1, V_2) \stackrel{\text{def}}{=} \{(v_1, v_2) \in E : v_1 \in V_1, v_2 \in V_2\}$. For $\gamma \in (0,1)$ we use the notation $a = (1 \pm \gamma)b$ to mean that $(1-\gamma)b \leq a \leq (1+\gamma)b$. We start by giving the high-level idea behind the algorithm.

2.1 A High-Level Description of the Algorithm.

Let $\beta = \epsilon/c$ where $c > 1$ is a constant that will be set subsequently, and let $t = \left\lceil \log_{(1+\beta)} n \right\rceil$ (so that $t = O(\log n/\epsilon)$). For $i = 0, \ldots, t$, let $B_i \stackrel{\text{def}}{=} \{v : d(v) \in \left((1+\beta)^{i-1}, (1+\beta)^i\right]\}$. We refer to the B_i's as (degree) *buckets*. Note that since degrees are integers, the interval of degrees in each bucket is actually $\left(\left\lfloor (1+\beta)^{i-1} \right\rfloor, \left\lfloor (1+\beta)^i \right\rfloor\right]$, and some buckets are empty. For simplicity we do not use floors unless it has an influence on our analysis, and when we write $\binom{a}{b}$ for a that is not necessarily an integer (e.g., $\binom{(1+\beta)^i}{2}$) then we interpret it as $\binom{\lfloor a \rfloor}{b}$. We also have that $\binom{a}{b} = 0$ for $a < b$ (and in particular when $a \leq 0 < b$).

Suppose that for each bucket B_i we could obtain an estimate, $\hat{b}_i$, such that $(1-\beta)|B_i| \leq \hat{b}_i \leq (1+\beta)|B_i|$. If we let $\hat{\ell} = \sum_{i=0}^{t} \hat{b}_i \cdot \binom{(1+\beta)^i}{2}$ then $(1-\beta) \cdot \ell(G) \leq \hat{\ell} \leq (1+\beta)^4 \ell(G)$. If we set $\beta \leq \epsilon/8$, then we get an estimate that is within $(1 \pm \epsilon)$ of the correct value $\ell(G)$. The difficulty is that in order to obtain such an estimate $\hat{b}_i$ of $|B_i|$ in sublinear time, that is, by sampling, the size of the sample needs to grow with $n/|B_i|$. Our algorithm indeed takes a sample of vertices, but it uses it only to estimate the size of the "large" buckets, for an appropriate threshold of "largeness". Using the estimated sizes of the large buckets it can obtain an estimate on the number of length-2 paths whose mid-point belongs to the large buckets.

As noted in the introduction, it is possible that only a small (or even zero) fraction of the length-2 paths have a mid-point that belongs to a large bucket. This implies that we must find a way to estimate the number of length-2 paths whose mid-point is in small buckets (for those small buckets that have a non-negligble contribution to the total number of length-2 paths).

To this end we do the following. Let $E_{i,j} \stackrel{\text{def}}{=} E(B_i, B_j)$. For each large bucket B_i and small bucket B_j such that the number of length-2 paths whose mid-point is in B_j is non-negligble, we obtain an estimate $\hat{e}_{i,j}$ to the number, $|E_{i,j}|$, of edges between the two buckets. The estimate is such that if $|E_{i,j}|$ is above some threshold, then $\hat{e}_{i,j} = (1 \pm \beta)|E_{i,j}|$, and otherwise, $\hat{e}_{i,j}$ is small. Our estimate for the number of length-2 paths whose midpoint is in a small bucket is $\frac{1}{2} \sum_{i \in L} \sum_{j \notin L} \hat{e}_{i,j} \cdot \left((1+\beta)^j - 1\right)$, where L denotes the set of indices of the large buckets. This estimate does not take into account length-2 paths in which all vertices on the path do not belong to L. However, we shall set our threshold of "largeness" so that the number of such paths is negligble.

One way to estimate $\hat{e}_{i,j}$ (for $i \in L$ and $j \notin L$) is to uniformly select random neighbors of vertices sampled in B_i and check what bucket they belong to. This will indeed give us a good estimate with high probability for a sufficiently large sample. However, the variance in the number of neighbors in B_j that different vertices in B_i have implies that the sample size used by this scheme is significantly larger than necessary. In order to obtain an estimate with a smaller sample, we do the following. For each $i \in L$ and $j \notin L$ we consider partitioning the vertices in B_i that have neighbors in B_j into *sub-buckets*. Namely, for $r = 0, \ldots, i$, $B_{i,j,r} \stackrel{\text{def}}{=} \{v \in B_i : (1+\beta)^{r-1} < |\Gamma(v) \cap B_j| \leq (1+\beta)^r\}$. By the definition of $B_{i,j,r}$ we have that $|B_{i,j,r}| \cdot (1+\beta)^r = (1 \pm \beta) \cdot |E_{i,j}|$. Now, if we can obtain good estimates of the sizes of the subsets $|B_{i,j,r}|$, then we get a good estimate for $|E_{i,j}|$. The difficulty is that while in order to determine for a vertex v to which bucket B_i it belongs, we only need to perform a single degree query, in order to determine to which sub-bucket $B_{i,j,r}$ it belongs we need to estimate the number of neighbors that it has in B_j. In particular, if $v \in B_{i,j,0}$, that is, v has a single neighbor in B_j, we must query all the neighbors of v in order to determine that it belongs to $B_{i,j,0}$. What works in our favor is the following tradeoff. When r is large then $|B_{i,j,r}|$ may be relatively small (even if $|E(B_{i,j,r}, B_j)|$ is non-negligble) so that we need to take a relatively large sample of vertices in order to "hit" $B_{i,j,r}$. However, in order to determine whether a vertex (in B_i) belongs to $B_{i,j,r}$ for large r, it suffices to take a small sample of its neighbors. On the other hand, when r is relatively small then $B_{i,j,r}$ must be relatively big (if $|E(B_{i,j,r}, B_j)|$ is non-negligble). Therefore, it suffices to take a relatively small sample so as to "hit" $B_{i,j,r}$ and then we can afford performing many neighbor queries from the selected vertices.

We next present our algorithm in detail and then analyze it.

2.2 The Algorithm. In what follows we assume that we have a rough estimate $\tilde{\ell}$ such that $\frac{1}{2}\ell(G) \leq \tilde{\ell} \leq 2\ell(G)$. We later remove this assumption. Recall that for any two buckets B_i and B_j we use the shorthand $E_{i,j}$ for $E(B_i, B_j)$.

ALGORITHM 1. (Estimating the number of length-2 paths for $G = (V, E)$)
Input: ϵ *and* $\tilde{\ell}$.

1. *Let* $\beta \stackrel{\text{def}}{=} \frac{\epsilon}{32}$, $t \stackrel{\text{def}}{=} \left\lceil \log_{(1+\beta)} n \right\rceil$, *and* $\theta_1 \stackrel{\text{def}}{=} \frac{\epsilon^{2/3}\tilde{\ell}^{1/3}}{32t^{4/3}}$.
2. *Uniformly and independently select* $\Theta\left(\frac{n}{\theta_1} \cdot \frac{\log t}{\epsilon^2}\right)$ *vertices from* V, *and let* S *denote the multiset of selected vertices (that is, we allow repetitions).*
3. *For* $i = 0, \ldots, t$ *determine* $S_i = S \cap B_i$ *by performing a degree query on every vertex in* S.
4. *Let* $L = \left\{ i \,:\, \frac{|S_i|}{|S|} \geq 2\frac{\theta_1}{n} \right\}$.
   If $\max_{i \in L} \left\{ \binom{(1+\beta)^{i-1}}{2} \cdot \theta_1 \right\} > 4\tilde{\ell}$ *then terminate.*
5. *For each* $i \in L$ *run Algorithm 2 to get estimates* $\{\hat{e}_{i,j}\}_{j \notin L}$ *for* $\{|E_{i,j}|\}_{j \notin L}$.
6. *Output*

$$\hat{\ell} \;=\; \sum_{i \in L} n \cdot \frac{|S_i|}{|S|} \cdot \binom{(1+\beta)^i}{2} + \sum_{j \notin L} \frac{1}{2} \sum_{i \in L} \hat{e}_{i,j} \cdot \left((1+\beta)^j - 1\right) .$$

ALGORITHM 2. (Estimating $\{|E_{i,j}|\}$ for a given $i \in L$ and all $j \notin L$)
Input: L, $i \in L$, ϵ *and* $\tilde{\ell}$.

1. *For each* $0 \leq p \leq i$ *let* $\theta_2(p) \stackrel{\text{def}}{=} \frac{\epsilon^{3/2}\tilde{\ell}^{1/2}}{c_2 t^{5/2}(1+\beta)^{p/2}}$, *where* c_2 *is a constant that will be set in the analysis, (and where* $t = \left\lceil \log_{(1+\beta)} n \right\rceil$ *for* $\beta = \epsilon/32$*). Let* p_0 *be the smallest value of* p *satisfying* $\frac{1}{4}\theta_2(p+1) \leq n$.
2. *For* $p = i$ *down to* p_0 *initialize* $\hat{S}^{(p)}_{i,j,p} = \emptyset$.
3. *For* $p = i$ *down to* p_0 *do:*
   (a) *Let* $s^{(p)} = \Theta\left(\frac{n}{\theta_2(p)} \cdot \left(\frac{t}{\beta}\right)^2 \log t\right)$, *and let* $g^{(p)} = \Theta\left(\frac{(1+\beta)^{i-p}\log(tn)}{\beta^2}\right)$.
   (b) *Uniformly, independently at random select* $s^{(p)}$ *vertices from* $S^{(p+1)}$ *(where* $S^{(i+1)} = V$*) and let* $S^{(p)}$ *be the multiset of vertices selected.*
   (c) *Determine* $S^{(p)}_i = S^{(p)} \cap B_i$ *by performing a degree query on every vertex in* $S^{(p)}$. *If* $|S^{(p)}_i| < \frac{s^{(p)}}{n} \cdot \frac{1}{4(1+\beta)}\theta_2(p)$, *then go to Step 4. Else, if* $|S^{(p)}_i| > \frac{s^{(p)}}{n} \cdot \frac{4\tilde{\ell}}{\binom{(1+\beta)^{i-1}}{2}}$ *then terminate.*
   (d) *For each* $v \in S^{(p)}_i$ *select (uniformly, independently at random)* $g^{(p)}$ *neighbors of* v, *and for each* $j \notin L$ *let* $\gamma^{(p)}_j(v)$ *be the number of these neighbors that belong to* B_j. *(If* $g^{(p)} \geq d(v)$ *then consider all neighbors of* v.*)*
   (e) *For each* $j \notin L$ *and for each* $v \in S^{(p)}_i \setminus \bigcup_{p'>p} \hat{S}^{(p')}_{i,j,p'}$, *if* $\frac{(1+\beta)^{p-1}}{d(v)} < \frac{\gamma^{(p)}_j(v)}{g^{(p)}(v)} \leq \frac{(1+\beta)^p}{d(v)}$ *then add* v *to* $\hat{S}^{(p)}_{i,j,p}$.
4. *For each* $j \notin L$ *let* $\hat{e}_{i,j} = \sum_{p=p_0}^{i} \frac{n}{s^{(p)}} \cdot |\hat{S}^{(p)}_{i,j,p}| \cdot (1+\beta)^p$.
5. *Return* $\{\hat{e}_{i,j}\}_{j \notin L}$.

THEOREM 1. *If* $\frac{1}{2}\ell(G) \leq \tilde{\ell} \leq 2\ell(G)$ *then with probability at least* $2/3$, *the output,* $\hat{\ell}$, *of Algorithm 1 satisfies* $\hat{\ell} = (1 \pm \epsilon) \cdot \ell(G)$. *The query complexity and running time of the algorithm are* $O\left(\frac{n}{\ell^{1/3}} + \min\left\{n^{1/2}, \frac{n^{3/2}}{\ell^{1/2}}\right\}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon)$.

We first prove the second part of Theorem 1, concerning the complexity of the algorithm and then turn to proving the first part, concerning the quality of the output of the algorithm. We later show how to remove the assumption that the algorithm has an estimate $\tilde{\ell}$ for $\ell(G)$.

2.3 Proof of the second part of Theorem 1. The running time of Algorithm 1 is linear in its query complexity, and hence it suffices to bound the latter. The query complexity of Algorithm 1 is the sum of $\Theta\left(\frac{n}{\theta_1} \cdot \frac{\log t}{\epsilon^2}\right) = O\left(\frac{n}{\ell^{1/3}} \cdot \frac{\log^2 n}{\epsilon^4 \log(1/\epsilon)}\right)$ (the size of the sample selected in Step 2 of the algorithm) and the number of queries performed in the executions of Algorithm 2. In order to bound the latter we first observe that if Algorithm 1 did not terminate in Step 4, then

$$\forall\, i \in L : \quad (1+\beta)^i = O\left(\frac{\tilde{\ell}^{1/3} \cdot t^{1/2}}{\epsilon^{1/3}}\right) . \tag{2.2}$$

Similarly, if Algorithm 2 did not terminate in any of its executions in Step 3c, then

$$\forall\, i \in L \;\text{ and }\; p_0 \leq p \leq i \;:$$

$$|S_i^{(p)}| = O\left(\frac{s^{(p)}}{n}\cdot\frac{\tilde{\ell}}{(1+\beta)^{2i}}\right)\ . \tag{2.3}$$

In addition, it trivially always holds that $|S_i^{(p)}| \le s^{(p)}$. Recall that p runs from i down to p_0 where p_0 is the smallest value of p satisfying $\frac{1}{4}\theta_2(p+1) \le n$ where $\theta_2(p) \stackrel{\text{def}}{=} \frac{\epsilon^{3/2}\tilde{\ell}^{1/2}}{c_2 t^{5/2}(1+\beta)^{p/2}}$. That is, $p_0 = \left\lfloor \log_{1+\beta} \frac{\epsilon^3\tilde{\ell}}{c_2' \cdot t^5 n^2}\right\rfloor$ for a certain constant c_2'. This implies that if if $\tilde{\ell} \le \frac{c_2' \cdot t^5}{\epsilon^3}\cdot n^2$, then $p_0 = 0$, and otherwise it may be larger. Therefore, the total number of queries performed in the executions of Algorithm 2 is upper bounded by:

$$\begin{aligned}
&\sum_{i\in L}\sum_{p=p_0}^{i}\left(s^{(p)} + \min\left\{s^{(p)}, \frac{s^{(p)}}{n}\cdot\frac{4\tilde{\ell}}{\binom{(1+\beta)^{i-1}}{2}}\right\}\cdot g^{(p)}\right) \\
&\quad\le \sum_{i\in L} i\cdot s^{(i)} \\
&\quad + \sum_{i\in L}\sum_{p=p_0}^{i}\min\left\{s^{(p)}, \frac{s^{(p)}}{n}\cdot\frac{4\tilde{\ell}}{\binom{(1+\beta)^{i-1}}{2}}\right\}\cdot g^{(p)} \qquad (2.4)
\end{aligned}$$

For the first summand we apply Equation (2.2) and get:

$$\begin{aligned}
\sum_{i\in L} i\cdot s^{(i)} &\le \sum_{i\in L} t\cdot O\left(\frac{n\cdot t^{9/2}\log t\cdot(1+\beta)^{i/2}}{\epsilon^{7/2}\tilde{\ell}^{1/2}}\right) \\
&= O\left(\frac{n\cdot t^{13/2}\log t\cdot\left(\frac{\tilde{\ell}^{1/3}\cdot t^{1/2}}{\epsilon^{1/3}}\right)^{1/2}}{\epsilon^{7/2}\tilde{\ell}^{1/2}}\right) \\
&= O\left(\frac{n}{\tilde{\ell}^{1/3}}\cdot\frac{t^7\log t}{\epsilon^4}\right)\ . \qquad (2.5)
\end{aligned}$$

Turning to the second summand,

$$\begin{aligned}
&\sum_{i\in L}\sum_{p=p_0}^{i}\min\left\{s^{(p)}, \frac{s^{(p)}}{n}\cdot\frac{4\tilde{\ell}}{\binom{(1+\beta)^{i-1}}{2}}\right\}\cdot g^{(p)} \\
&= \sum_{i\in L}\sum_{p=p_0}^{i} O\Big(\min\Big\{\frac{n\cdot(1+\beta)^{p/2}}{\tilde{\ell}^{1/2}}\cdot\frac{t^{13/2}\log t}{\epsilon^{7/2}}, \\
&\qquad \frac{\tilde{\ell}^{1/2}}{(1+\beta)^{2i-p/2}}\cdot\frac{t^{13/2}\log t}{\epsilon^{7/2}}\Big\} \\
&\qquad \cdot\frac{(1+\beta)^{i-p}\log(tn)}{\beta^2}\Big) \\
&= \sum_{i\in L} O\Big(\min\Big\{\frac{n\cdot(1+\beta)^{i}}{\tilde{\ell}^{1/2}}, \frac{\tilde{\ell}^{1/2}}{(1+\beta)^{i}}\Big\} \\
&\qquad \cdot\frac{t^{13/2}\log t\log(tn)}{\epsilon^{11/2}}\Big) \\
&\qquad \cdot(1+\beta)^{-p_0/2}\cdot\sum_{k=0}^{i-p_0}(1+\beta)^{-k/2}\ . \qquad (2.6)
\end{aligned}$$

In order to bound the expression in Equation (2.6) we first note that if $(1+\beta)^i \le \frac{\tilde{\ell}^{1/2}}{n^{1/2}}$ then $\frac{n\cdot(1+\beta)^i}{\tilde{\ell}^{1/2}} \le n^{1/2}$, while if $(1+\beta)^i \ge \frac{\tilde{\ell}^{1/2}}{n^{1/2}}$, then $\frac{\tilde{\ell}^{1/2}}{(1+\beta)^i} \le n^{1/2}$ as well. Therefore, if $p_0 = 0$ then (since $(1+\beta)^{-p_0/2} = 1$ and $\sum_{k=0}^{i-p_0}(1+\beta)^{-k/2} = O(1/\beta)$), the right-hand-side of Equation (2.6) is upper bounded by

$$O\left(n^{1/2}\cdot\frac{t^{15/2}\log t\log(tn)}{\epsilon^{13/2}}\right)\ . \tag{2.7}$$

If $p_0 > 0$ then the bound in Equation (2.7) should be multiplied by $(1+\beta)^{-p_0/2}$. By definition of p_0 we have that $(1+\beta)^{-p_0/2} = O\left(\frac{t^{5/2}n}{\epsilon^{3/2}\tilde{\ell}^{1/2}}\right)$, and so we get the (tighter) bound:

$$\begin{aligned}
&O\left(n^{1/2}\cdot\frac{t^{15/2}\log t\log(tn)}{\epsilon^{13/2}}\right)\cdot(1+\beta)^{-p_0/2} \\
&\quad = O\left(\frac{n^{3/2}}{\tilde{\ell}^{1/2}}\cdot\frac{t^{10}\log t\log(tn)}{\epsilon^{10}}\right)\ . \qquad (2.8)
\end{aligned}$$

The total number of queries performed in the executions of Algorithm 2 is hence upper bounded by

$$O\left(\frac{n}{\tilde{\ell}^{1/3}} + \min\left\{n^{1/2}, \frac{n^{3/2}}{\tilde{\ell}^{1/2}}\right\}\right)\cdot\mathrm{poly}(\log n, 1/\epsilon)\ .$$

2.4 Proof of the first part of Theorem 1. In what follows we claim that certain events occur with high constant probability, and in some cases we claim that they hold with larger probability (e.g., $1-\frac{1}{\mathrm{poly}(t)}$). In all cases the statement holds for sufficiently large constants in the $\Theta(\cdot)$ notations for the sample sizes used by the algorithm. The next lemma follows by applying the multiplicative Chernoff bound and the union bound and recalling that the size of the sample S is $\Theta\left(\frac{n}{\theta_1}\cdot\frac{\log t}{\epsilon^2}\right)$, where θ_1 is as defined in Step 1 of Algorithm 1.

LEMMA 1. *With high constant probability, for every i such that $|B_i| \ge \theta_1$ it holds that $\frac{|S_i|}{|S|} = \left(1\pm\frac{\epsilon}{8}\right)\frac{|B_i|}{n}$, and for every i such that $|B_i| < \theta_1$ it holds that $\frac{|S_i|}{|S|} < 2\frac{\theta_1}{n}$.*

As a direct corollary of the Lemma 1 and the definition of L in Algorithm 1 we get:

COROLLARY 2. *With high constant probability, for every $i \in L$ we have that $\frac{|S_i|}{|S|} = \left(1\pm\frac{\epsilon}{8}\right)\frac{|B_i|}{n}$, and for every $i \notin L$ we have that $|B_i| < 4\theta_1$.*

The first part of Corollary 2 implies that (with high constant probability) the estimate $\sum_{i\in L} n\cdot\frac{|S_i|}{|S|}\cdot\binom{(1+\beta)^i}{2}$ is close to the actual number of length-2 paths whose

mid-point belongs to a bucket B_i such that $i \in L$. It is not hard to verify that it also implies that Algorithm 1 does not terminate in Step 4 (with high constant probability).

The remainder of the analysis deals with the quality of the estimate for the number of length-2 paths in G whose mid-point is not in L.

We introduce the following notations. For $j \notin L$ and $\sigma \in \{1,2,3\}$, let $\ell_j^{(\sigma)}(G,\overline{L})$ denote the number of length-2 paths in G whose mid-point belongs to B_j and such that the number of vertices on the path that belong to B_k for $k \notin L$ (including j) is σ. For $\sigma \in \{1,2,3\}$ let $\ell^{(\sigma)}(G,\overline{L}) = \sum_{j \notin L} \ell_j^{(\sigma)}(G,\overline{L})$ and for every $j \notin L$ let $\ell_j(G,\overline{L}) = \sum_{\sigma=1}^{3} \ell_j^{(\sigma)}(G,\overline{L})$. We first observe that with high constant probability both $\ell^{(3)}(G,\overline{L})$ and $\ell^{(2)}(G,\overline{L})$ are relatively small.

LEMMA 3. *With high constant probability,* $\ell^{(3)}(G,\overline{L}) \leq \frac{\epsilon}{4}\ell(G)$ *and* $\ell^{(2)}(G,\overline{L}) \leq \frac{\epsilon}{4}\ell(G)$.

Proof. First observe that by the second part of Corollary 2 and the definition of θ_1 we have that with high constant probability,

$$\sum_{j \notin L} |B_j| \;<\; \frac{1}{8t^{1/3}} \epsilon^{2/3} \tilde{\ell}^{1/3} \;. \tag{2.9}$$

By our assumption that $\tilde{\ell} \leq 2\ell(G)$,

$$\begin{aligned} \ell^{(3)}(G,\overline{L}) &\leq \binom{\sum_{j \notin L} |B_j|}{3} \\ &\leq \binom{\epsilon^{2/3}\tilde{\ell}^{1/3}/(8t^{1/3})}{3} \\ &< \frac{\epsilon}{8}\tilde{\ell} \;\leq\; \frac{\epsilon}{4}\ell(G) \;. \end{aligned} \tag{2.10}$$

In order to bound $\ell^{(2)}(G,\overline{L})$ we observe that since the total number of length-2 paths is $\ell(G)$, for every bucket B_j we have that $\binom{(1+\beta)^{j-1}+1}{2} \leq \ell(G)/|B_j|$, and so

$$(1+\beta)^j \;\leq\; 2\frac{\ell^{1/2}(G)}{|B_j|^{1/2}} \;. \tag{2.11}$$

Therefore,

$$\begin{aligned} \ell^{(2)}(G,\overline{L}) &\leq \sum_{j \notin L} |B_j| \cdot (1+\beta)^j \cdot \sum_{k \notin L} |B_k| \\ &\leq \frac{\epsilon^{2/3}\tilde{\ell}^{1/3}}{4t^{1/3}} \cdot \sum_{j \notin L} \left(\ell^{1/2}(G) \cdot |B_j|^{1/2} \right) \\ &\leq \frac{\epsilon^{2/3}\tilde{\ell}^{1/3}}{4t^{1/3}} \cdot \ell^{1/2}(G) \cdot t \cdot \frac{\epsilon^{1/3}\tilde{\ell}^{1/6}}{2\sqrt{2}t^{2/3}} \\ &< \frac{\epsilon}{4}\ell(G) \;, \end{aligned} \tag{2.12}$$

and the proof is completed.

Lemma 3 implies that in order to obtain a good estimate on the number of length-2 paths whose mid-point belongs to small buckets it suffices to get a good estimate on the number of such paths that have at least one end-point in a large bucket.[4] We next define the notion of significant buckets for buckets B_j such that $j \notin L$. Roughly speaking, non-significant small buckets are buckets that we can ignore, or, more precisely, we can undercount the number of edges between vertices in them and vertices in large buckets.

DEFINITION 1. (SIGNIFICANT SMALL BUCKETS) *For every $j \notin L$ we say that j is* significant *if $|B_j| \cdot \binom{(1+\beta)^j}{2} \geq \frac{\epsilon}{c_3 t}\tilde{\ell}$, where c_3 is a constant that will be set in the analysis. We denote the set of indices of significant buckets B_j (where $j \notin L$) by SIG.*

Note that by the definition of SIG,

$$\sum_{j \notin L, j \notin SIG} \ell_j(G,\overline{L}) \;<\; \frac{\epsilon}{c_3}\tilde{\ell} \;\leq\; \frac{2\epsilon}{c_3}\ell(G) \;. \tag{2.13}$$

Let $E_j \stackrel{\text{def}}{=} \bigcup_{k=0}^{t} E_{j,k}$, and recall that $\theta_2(r) \stackrel{\text{def}}{=} \frac{\epsilon^{3/2}\tilde{\ell}^{1/2}}{c_2 t^{5/2}(1+\beta)^{r/2}}$. We have the following lemma concerning significant buckets.

LEMMA 4. *If $j \in SIG$, then for every r such that $|B_{i,j,r}| > 0$ for some i we have that*

$$|E_j| \geq \frac{(c_2/c_3^{1/2})t^2}{\epsilon}\theta_2(r) \cdot (1+\beta)^r \;.$$

The implication of Lemma 4 is roughly the following. Consider any $j \in SIG$ and a non-empty sub-bucket $B_{i,j,r}$. Recall that by the definition of $B_{i,j,r}$ the number of edges between $B_{i,j,r}$ and B_j is approximately $|B_{i,j,r}| \cdot (1+\beta)^r$. Suppose that $B_{i,j,r}$ is small, and in particular, that it is smaller than $\theta_2(r)$. Then the number of edges between $B_{i,j,r}$ and B_j as a fraction of all the edges incident to B_j, that is, E_j, is $O(\epsilon/t^2)$, which is negligble. This means that we may underestimate the size of such small sub-buckets without incurring a large error.

Proof. Since j is significant,

$$(1+\beta)^j > \sqrt{\frac{2\epsilon\tilde{\ell}}{c_3 t|B_j|}} \;. \tag{2.14}$$

Since the graph contains no multiple edges, $|B_j| \geq (1+\beta)^r$ for every non-empty $B_{i,j,r}$. Therefore,

$$|E_j| \;\geq\; |B_j| \cdot (1+\beta)^{j-1}$$

[4] The assertion follows from the first part of Lemma 3, which bounds $\ell^{(3)}(G,\overline{L})$. The reason we also need a bound on $\ell^{(2)}(G,\overline{L})$ will be made clear subsequently.

$$\geq \frac{1}{1+\beta}\sqrt{\frac{2\epsilon\tilde{\ell}}{c_3 t|B_j|}} \tag{2.15}$$

$$\geq \frac{1}{c_3^{1/2}t^{1/2}} \cdot \epsilon^{1/2}\tilde{\ell}^{1/2}(1+\beta)^{r/2} \tag{2.16}$$

$$\geq \frac{(c_2/c_3^{1/2})t^2}{\epsilon}\theta_2(r)\cdot(1+\beta)^r\ , \tag{2.17}$$

and the proof is completed.

Armed with Lemmas 3 and 4 we now turn to analyzing Algorithm 2. We start with a high-level discussion and then turn to the precise details.

The high level structure of the analysis of Algorithm 2. Recall that the algorithm works iteratively as follows. It first takes (uniformly, independently, at random) a sample S^i from V, and in further iterations $0 \leq p < i$ the sample $S^{(p)}$ is selected (uniformly, independently, at random) from $S^{(p+1)}$. Since the same vertex may be selected more than once, the $S^{(p)}$'s may actually be multi-sets. For each p the algorithm tries to estimate $|B_{i,j,p}|$ by deciding for each vertex $v \in S^{(p)} \cap B_i$ whether it belongs to $B_{i,j,p}$. This is done by sampling from the neighbors of the vertex and checking what fraction of its neighbors belong to B_j. If the fraction is within some interval, then the vertex is assumed to belong to $B_{i,j,p}$ and is put in a corresponding subset $\hat{S}^{(p)}_{i,j,p}$.

The difficulty is that this estimate of the fraction of neighbors in B_j may deviate somewhat from its expected value. As a result, vertices that belong to $B_{i,j,p}$ may not be deemed so because the number of neighbors they have in B_j is close to the lower bound of $(1+\beta)^{p-1}$ or the upper bound $(1+\beta)^p$, and in the sample they fall outside of the required interval. Similarly, vertices that do not belong to $B_{i,j,p}$ (but have a number of neighbors in B_j that is close to $(1+\beta)^{p-1}$ or $(1+\beta)^p$, that is, that belong to $B_{i,j,p-1}$ or $B_{i,j,p+1}$) may fall inside the required interval and are then added to $\hat{S}^{(p)}_{i,j,p}$.

If the size of the sample $S^{(p)}$ was the same for all p then the above wouldn't really be a difficulty: we could take a single sample $S = S^i$ and work iteratively from $p = i$ down to $p = 0$. For each p we would consider only those vertices v that were not yet added to $\hat{S}^{(p')}_{i,j,p'}$ for $p' > p$ and decide whether to add v to $\hat{S}^{(p)}_{i,j,p}$. By the above discussion, for every r and every $v \in B_{i,j,r}$ the vertex v would be put either in $\hat{S}^{(r+1)}_{i,j,r+1}$ or in $\hat{S}^{(r)}_{i,j,r}$ or in $\hat{S}^{(r-1)}_{i,j,r-1}$. The algorithm would then output, as an estimate for $|E_{i,j}|$, the sum over all $0 \leq p \leq i$, of $\frac{n}{|S|} \cdot |\hat{S}^{(p)}_{i,j,p}|(1+\beta)^r$. If $S \cap B_{i,j,r}$ is close to its expected size for each r then the deviation of the final estimate from $|E_{i,j}|$ can be easily bounded.

However, as p decreases from i to 0 we need to use a smaller sample $S^{(p)}$. Recall that a smaller sample suffices since $\theta_2(p)$ increases when p decreases, and it is necessary to use a smaller sample because the cost of estimating the number of neighbors in B_j increases as p decreases. Thus, in each iteration p, the new, smaller sample, $S^{(p)}$, is selected from the sample $S^{(p+1)}$ of the previous iteration. What we would like to ensure is that: (1) The size of each subset $S^{(p)}_{i,j,r} \stackrel{\text{def}}{=} S^{(p)} \cap B_{i,j,r}$ is close to its expectation; (2) If some fraction of $S^{(p+1)}_{i,j,r}$ was added to $\hat{S}^{(p+1)}_{i,j,p+1}$ for $r = p+1$ or $r = p$, then in the new sample $S^{(p)}$, the size of $S^{(p)} \cap (S^{(p+1)}_{i,j,r} \setminus \hat{S}^{(p+1)}_{i,j,p+1})$ is close to its expectation. Here, when we say "close to its expectation" we mean up to a multiplicative factor of $(1 \pm O(\epsilon))$. This should be the case unless the expected value is below some threshold (which is determined by $\theta_2(r)$). If the expected value is below the threshold then it suffices that we don't get a significant overestimate. To get the idea for why this suffices, see the discussion following Lemma 4. Further details follow.

Recall that $s^{(p)}$ denotes the size of the sample $S^{(p)}$, where $s^{(p)} = \Theta\left(\frac{n}{\theta_2(p)} \cdot \left(\frac{t}{\beta}\right)^2 \log t\right)$. The next lemma established that by our choice of $s^{(p)}$, if a fixed subset of $S^{(p+1)}$ is sufficiently large, then the number of its vertices that are selected in $S^{(p)}$ is close to the expected value, and if it is small then few of its vertices will appear in $S^{(p)}$. Lemma 5 follows directly be applying a multiplicative Chernoff bound (and will be applied to various subsets of the samples $S^{(p)}$).

LEMMA 5. *For any fixed choice of $\tilde{S}^{(p+1)} \subseteq S^{(p+1)}$, if $\frac{|\tilde{S}^{(p+1)}|}{s^{(p+1)}} \geq \frac{\theta_2(p)}{8n}$ then, with probability at least $1 - \frac{1}{32t^4}$,*

$$\frac{|S^{(p)} \cap \tilde{S}^{(p+1)}|}{s^{(p+1)}} \leq \left(1 + \frac{\beta}{2(i+1)}\right) \cdot \frac{|\tilde{S}^{(p+1)}|}{s^{(p+1)}}\ ,$$

and

$$\frac{|S^{(p)} \cap \tilde{S}^{(p+1)}|}{s^{(p+1)}} \geq \frac{1}{1+\frac{\beta}{2(i+1)}} \cdot \frac{|\tilde{S}^{(p+1)}|}{s^{(p+1)}}\ ,$$

and if $\frac{|\tilde{S}^{(p+1)}|}{s^{(p+1)}} < \frac{\theta_2(p)}{8n}$ then with probability at least $1 - \frac{1}{32t^4}$,

$$\frac{|S^{(p)} \cap \tilde{S}^{(p+1)}|}{s^{(p)}} < \left(1 + \frac{\beta}{2(i+1)}\right) \cdot \frac{\theta_2(p)}{8n}\ .$$

Let $S^{(p)}_i \stackrel{\text{def}}{=} S^{(p)} \cap B_i$ and let $S^{(p)}_{i,j,r} \stackrel{\text{def}}{=} S^{(p)} \cap B_{i,j,r}$. (Note that $S^{i+1}_i = B_i$ and $S^{i+1}_{i,j,r} = B_{i,j,r}$). Since $\theta_2(p)$ is monotonically decreasing with p (so that $s^{(p)}$ is monotonically increasing with r), and because $(1 + \frac{\beta}{2(i+1)})^{i+1} \leq 1 + \beta$, Lemma 5 implies the next corollary.

COROLLARY 6. *With high constant probability, for every $i \in L$ and $j \notin L$, and for every r such that $|B_{i,j,r}| \geq \frac{1}{4}\theta_2(r)$, we have that for every $r-1 \leq p \leq i$,*

$$\frac{|S^{(p)}_{i,j,r}|}{s^{(p)}} \leq \left(1+\frac{\beta}{2(i+1)}\right)^{i-p+1} \cdot \frac{|B_{i,j,r}|}{n}.$$

and

$$\frac{|S^{(p)}_{i,j,r}|}{s^{(p)}} \geq \left(\frac{1}{1+\frac{\beta}{2(i+1)}}\right)^{i-p+1} \cdot \frac{|B_{i,j,r}|}{n}$$

On the other hand, if $|B_{i,j,r}| < \frac{1}{4}\theta_2(r)$ then

$$\frac{|S^{(p)}_{i,j,r}|}{s^{(p)}} < (1+\beta)\cdot\frac{\theta_2(r)}{4n}$$

for every p.

Lemma 5 also implies that with high constant probability, Algorithm 2 does not terminate in Step 3c. Recall that the algorithms terminates in Step 3c if $n\cdot\frac{|S^{(p)}_i|}{s^{(p)}} \geq \frac{1}{4(1+\beta)}\theta_2(p)$ and $n\cdot\frac{|S^{(p)}_i|}{s^{(p)}}\cdot\binom{(1+\beta)^{i-1}}{2} > 4\tilde{\ell}$. By Lemma 5, with probability at least $1-\frac{1}{32t^2}$, for every i and p, if $|B_i| < \frac{1}{6}\theta_2(p)$, then $n\cdot\frac{|S^{(p)}_i|}{s^{(p)}} \leq (1+\beta)\frac{1}{6}\theta_2(p)$, and if $|B_i| \geq \frac{1}{6}\theta_2(p)$, then $n\cdot\frac{|S^{(p)}_i|}{s^{(p)}} \leq (1+\beta)|B_i|$. Assuming this is in fact the case, if $|B_i| < \frac{1}{6}\theta_2(p)$ then $n\cdot\frac{|S^{(p)}_i|}{s^{(p)}} < \frac{1}{4(1+\beta)}\theta_2(p)$, so that the algorithm will not terminate. On the other hand, if $|B_i| \geq \frac{1}{6}\theta_2(p)$, then

$$\begin{aligned} n\cdot\frac{|S^{(p)}_i|}{s^{(p)}}\cdot\binom{(1+\beta)^{i-1}}{2} &\leq (1+\beta)|B_i|\cdot\binom{(1+\beta)^{i-1}}{2} \\ &\leq (1+\beta)\ell(G) < 4\tilde{\ell}\,, \end{aligned} \tag{2.18}$$

so that the algorithm will not terminate in this case as well.

The next Lemma deals with the estimates we get for the number of neighbors that a vertex in B_i has in B_j, and it too follows from the multiplicative Chernoff bound. In the lemma and what follows we shall use the notations $\Gamma_j(v) \stackrel{\text{def}}{=} \Gamma(v)\cap B_j$ and $d_j(v) \stackrel{\text{def}}{=} |\Gamma_j(v)|$.

LEMMA 7. *Let $i \in L$, $j \notin L$ and for each $0 \leq p \leq i$, let $g^{(p)} = \Theta\left(\frac{(1+\beta)^{i-p}\cdot\log(t\cdot n)}{\beta^2}\right)$. For any $r \geq p-1$ and for any fixed choice of a vertex $v \in S^{(p)}_{i,j,r}$, if we take a sample of size $g^{(p)}$ of neighbors of v and let $\gamma^{(p)}_j(v)$ be the number of neighbors in the sample that belong to $\Gamma_j(v)$, then with probability at least $1-\frac{1}{16\cdot t\cdot n}$,*

$$\frac{1}{1+\beta}\cdot\frac{d_j(v)}{d(v)} \leq \frac{\gamma^{(p)}_j(v)}{g^{(p)}} \leq (1+\beta)\cdot\frac{d_j(v)}{d(v)}\,.$$

In addition, for each $r \leq p-2$ and $v \in S^{(p)}_{i,j,r}$, with probability at least $1-\frac{1}{16\cdot t\cdot n}$,

$$\frac{\gamma^{(p)}_j(v)}{g^{(p)}} < \frac{(1+\beta)^{p-1}}{d(v)}\,.$$

The next lemma is central to our analysis. Ideally we would have liked each vertex in the sample to be added to its "correct" subset. That is, if $v \in S^{(r)}_{i,j,r}$ $(= S^{(r)}\cap B_{i,j,r})$ then ideally it should be added to $\hat{S}^{(r)}_{i,j,r}$. However, since the decision concerning whether to add a vertex to a particular subset depends on sampling its neighbors and estimating the number of neighbors that it has in B_j, we can not ensure that it will be added to precisely the right subset. However, we can ensure (with high probability) that it won't be added to a subset $\hat{S}^{(p)}_{i,j,p}$ for p that differ significantly from r.

LEMMA 8. *With high constant probability, for every $i \in L$, $j \notin L$, $0 \leq r \leq i$ and $v \in B_{i,j,r}$ such that v is selected in the initial sample S^i, the vertex v may belong to either $\hat{S}^{(r+1)}_{i,j,r+1}$ or to $\hat{S}^{(r)}_{i,j,r}$ or to $\hat{S}^{(r-1)}_{i,j,r-1}$, but not to any other $\hat{S}^{(r')}_{i,j,r'}$. In other words, $\hat{S}^{(r)}_{i,j,r} \subseteq B_{i,j,r+1}\cup B_{i,j,r}\cup B_{i,j,r-1}$.*

Proof. By the definition of $B_{i,j,r}$, if $v \in B_{i,j,r}$ then $(1+\beta)^{r-1} < d_j(v) \leq (1+\beta)^r$. By Lemma 7, for each $p \leq r+1$ with probability at least $1-\frac{1}{16\cdot t\cdot n}$,

$$\begin{aligned} \frac{1}{1+\beta}\cdot\frac{(1+\beta)^{r-1}}{d(v)} &< \frac{\gamma^{(p)}_j(v)}{g^{(p)}} \\ &\leq (1+\beta)\cdot\frac{(1+\beta)^{r+1}}{d(v)}\,. \end{aligned} \tag{2.19}$$

That is, for each $p \leq r+1$ and in particular for $r-1 \leq p \leq r+1$,

$$\frac{(1+\beta)^{r-2}}{d(v)} < \frac{\gamma^{(p)}_j(v)}{g^{(p)}} \leq \frac{(1+\beta)^{r+2}}{d(v)}. \tag{2.20}$$

On the other hand, for $p \geq r+2$, with probability at least $1-\frac{1}{16\cdot t\cdot n}$,

$$\frac{\gamma^{(p)}_j(v)}{g^{(p)}} < \frac{(1+\beta)^{p-1}}{d(v)}\,. \tag{2.21}$$

By taking a union bound over all vertices v, and for each $v \in B_{i,j,r}$ over all $0 \leq p \leq i$, this implies that: (1) for $r+2 \leq p \leq i$, no vertex in $S^{(p)}_{i,j,r}$ is added to $\hat{S}^{(p)}_{i,j,r}$; (2) for $r-1 \leq p \leq r+1$ the following holds: If a vertex v belongs to $S^{(r+1)}_{i,j,r}$, then it may be added to $\hat{S}^{(r+1)}_{i,j,r+1}$, and if not, then it may be added to $\hat{S}^{(r)}_{i,j,r}$ (assuming $v \in S^{(r)}$). If it was added to neither of the two subsets and it is selected in $S^{(r-1)}$, them it is added to $\hat{S}^{(r-1)}_{i,j,r-1}$ (and it won't be added to $\hat{S}^{(p)}_{i,j,r}$ for any $p < r-1$).

In the next lemma we establish that the estimates $\hat{e}_{i,j}$ computed by Algorithm 2 are essentially close to the corresponding values of $|E_{i,j}|$. Recall that SIG denotes the set of all significant indices (as defined in Definition 1) and that $E_j \stackrel{\text{def}}{=} \bigcup_{k=0}^{t} E_{j,k}$.

LEMMA 9. *For an appropriate choice of c_2 (in the definition of $\theta_2(\cdot)$ in Step 1 of Algorithm 2) and of c_3 (in Definition 1), with high constant probability, for all $j \notin L$, if $j \in SIG$, then*

$$\left(1-\frac{\epsilon}{8}\right)\sum_{i\in L}|E_{i,j}| - \frac{\epsilon}{16}|E_j| \;\le\; \sum_{i\in L}\hat{e}_{i,j} \;\le\; \left(1+\frac{\epsilon}{4}\right)|E_j|\,,$$

and if $j \notin SIG$ then

$$\sum_{i\in L}\frac{1}{2}\hat{e}_{i,j}\cdot\left((1+\beta)^j-1\right) \;\le\; \frac{\epsilon}{4t}\ell(G)\,.$$

The proof of Lemma 9, which is quite technical (and long), can be found in the full version of this paper [18].

Putting it all together: proving the first part of Theorem 1. Recall that

$$\begin{aligned}\hat{\ell} \;&=\; \sum_{i\in L} n\cdot\frac{|S_i|}{|S|}\cdot\binom{(1+\beta)^i}{2} \\ &\quad + \sum_{j\notin L}\frac{1}{2}\sum_{i\in L}\hat{e}_{i,j}\cdot\left((1+\beta)^j-1\right)\,. \qquad (2.22)\end{aligned}$$

Let $\ell(G,L)$ denote the number of length-2 paths in G whose mid-point belongs to a bucket B_i such that $i \in L$, and let $\ell(G,\overline{L})$ denote the number of length-2 paths whose mid-point belongs to a bucket B_j such that $j \notin L$ (so that $\ell(G,L)+\ell(G,\overline{L}) = \ell(G)$). By the first part of Corollary 2 (and the setting of β) we have that with high constant probability:

$$\begin{aligned}&\sum_{i\in L} n\cdot\frac{|S_i|}{|S|}\cdot\binom{(1+\beta)^i}{2} \\ &\qquad = \left(1\pm\frac{\epsilon}{4}\right)\ell(G,L)\,. \qquad (2.23)\end{aligned}$$

Turning to the second summand in Equation (2.22), by Lemma 9,

$$\begin{aligned}&\sum_{j\notin L}\frac{1}{2}\sum_{i\in L}\hat{e}_{i,j}\cdot\left((1+\beta)^j-1\right) \\ &= \sum_{j\notin L, j\in SIG}\frac{1}{2}\sum_{i\in L}\hat{e}_{i,j}\cdot\left((1+\beta)^j-1\right) \\ &\qquad + \sum_{j\notin L, j\notin SIG}\frac{1}{2}\sum_{i\in L}\hat{e}_{i,j}\cdot\left((1+\beta)^j-1\right) \\ &\le \left(1+\frac{\epsilon}{4}\right)\cdot\sum_{j\notin L}\frac{1}{2}|E_j|\cdot\left((1+\beta)^j-1\right) \;+\; \frac{\epsilon}{4}\ell(G) \\ &\le \left(1+\frac{\epsilon}{2}\right)\ell(G,\overline{L}) \;+\; \frac{\epsilon}{4}\ell(G)\,. \qquad (2.24)\end{aligned}$$

In the other direction, recall that $\ell^{(\sigma)}(G,\overline{L}) = \sum_{j\notin L}\ell_j^{(\sigma)}(G,\overline{L})$ where for $j \notin L$ and $\sigma \in \{1,2,3\}$, we let $\ell_j^{(\sigma)}(G,\overline{L})$ denote the number of length-2 paths whose mid-point belongs to B_j and such that the number of vertices on the path that belong to B_k for $k \notin L$ (including j) is σ,

$$\begin{aligned}&\sum_{j\notin L}\frac{1}{2}\sum_{i\in L}\hat{e}_{i,j}\cdot(1+\beta)^j \\ &\ge \sum_{j\notin L, j\in SIG}\frac{1}{2}\left(\sum_{i\in L}\left(1-\frac{\epsilon}{8}\right)|E_{i,j}| - \frac{\epsilon}{16}|E_j|\right) \\ &\qquad ; \quad \cdot\left((1+\beta)^j-1\right) \\ &\ge \sum_{j\notin L}\frac{1}{2}\sum_{i\in L}\left(1-\frac{\epsilon}{8}\right)|E_{i,j}|\cdot\left((1+\beta)^j-1\right) \\ &\qquad - \frac{(1+\beta)\epsilon}{16}\ell(G) \\ &\qquad - \sum_{j\notin L, j\notin SIG}\frac{1}{2}\sum_{i\in L}\left(1-\frac{\epsilon}{8}\right)|E_{i,j}|\cdot\left((1+\beta)^j-1\right) \\ &\ge \left(1-\frac{\epsilon}{8}\right)\left(\ell^{(1)}(G,\overline{L})+\frac{1}{2}\ell^{(2)}(G,\overline{L})\right)-\frac{\epsilon}{4}\ell(G) \\ &\ge \left(1-\frac{\epsilon}{8}\right)\left(\ell^{(1)}(G,\overline{L})+\ell^{(2)}(G,\overline{L})+\ell^{(3)}(G,\overline{L})\right) \\ &\qquad - \frac{1}{2}\ell^{(2)}(G,\overline{L}) - \ell^{(3)}(G,\overline{L}) - \frac{\epsilon}{4}\ell(G) \\ &\ge \left(1-\frac{\epsilon}{8}\right)\ell(G,\overline{L}) - \frac{3\epsilon}{4}\ell(G)\,, \\ &\qquad (2.25)\end{aligned}$$

where we used Equation (2.13) (based on the definition of SIG and taking $c_3 \ge 32$ as it was set previously), and we applied Lemma 3. By combining Equations (2.23), (2.24) and (2.25) we get that $\hat{\ell} = (1\pm\epsilon)\ell(G)$ with high constant probability.

2.5 Removing the assumption on $\tilde{\ell}$. Our analysis builds on the assumption that $\frac{1}{2}\ell(G) \le \tilde{\ell} \le 2\ell(G)$. In order to get rid of this assumption we observe that if we run Algorithm 1 with $\tilde{\ell} > 2\ell(G)$ then our analysis implies that with high constant probability $\hat{\ell} \le \left(1+\frac{\epsilon}{2}\right)\ell(G) + \frac{\epsilon}{8}\tilde{\ell}$. This is true because: (1) the algorithm still obtains (with high probability) an estimate of the number of length-2 paths whose mid-point is in a bucket B_i where $i \in L$ that does not overestimate the actual number of such paths by more than a factor of $\left(1+\frac{\epsilon}{4}\right)$; (2) For the number of length-2 paths whose mid-point is in a bucket B_j where $j \notin L$ and

$j \in SIG$ the approximation factor is at most $\left(1+\frac{\epsilon}{2}\right)$; (3) The additional error caused by overestimating the number of length-2 paths whose mid-point is in a bucket B_j where $j \notin L$ and $j \notin SIG$ is at most $\frac{\epsilon}{8}\tilde{\ell}$.

Suppose we run Algorithm 1 with $\tilde{\ell} > 2\ell(G)$. Then with high constant probability $\hat{\ell} < \left(\frac{1}{2}+\frac{\epsilon}{2}\right)\tilde{\ell}$. On the other hand, if we run Algorithm 1 with $\frac{1}{2}\ell(G) \leq \tilde{\ell} < \ell(G)$, then with high constant probability, $\hat{\ell} \geq (1-\epsilon)\ell(G) > (1-\epsilon)\tilde{\ell}$, which is greater than $\left(\frac{1}{2}+\frac{\epsilon}{2}\right)\tilde{\ell}$ for every $\epsilon < 1/3$.

Therefore, we do the following. Starting from $\tilde{\ell} = n \cdot \binom{n}{2}$, we repeatedly call a slight variant of Algorithm 1 with our current estimate $\tilde{\ell}$. The variant is that we increase all sample sizes by a factor of $\Theta(\log\log n)$ so as to reduce the failure probability of each execution to $O(1/\log n)$ and we run the algorithm with $\epsilon = \min\{\epsilon, 1/4\}$. In each execution we reduce the previous value of $\tilde{\ell}$ by a factor of 2, and stop once $\hat{\ell} > (1-\epsilon)\tilde{\ell}$, at which point we output $\hat{\ell}$. By the above discussion, with high constant probability we do not stop before $\tilde{\ell}$ goes below $2\ell(G)$, and conditioned on this, with high probability $(1 - O(1/\log n))$ we do stop once $\frac{1}{2}\ell(G) \leq \tilde{\ell} < \ell(G)$ (or possibly, one iteration earlier, when $\ell(G) \leq \tilde{\ell} < 2\ell(G)$) with $\hat{\ell} = (1 \pm \epsilon)\ell(G)$.

Since there is a non-zero probability that the algorithm does not stop when $\frac{1}{2}\ell(G) \leq \tilde{\ell} < \ell(G)$, we next bound the expected running time of the algorithm. The total running time of all executions until $\frac{1}{2}\ell(G) \leq \tilde{\ell} < \ell(G)$ is $O\left(\frac{n}{\ell(G)^{1/3}} + \min\left\{n^{1/2}, \frac{n^{3/2}}{\ell(G)^{1/2}}\right\}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon)$. Once $\tilde{\ell} < \frac{1}{2}\ell(G)$, the algorithm may terminate in Step 4 of Algorithm 1 or in Step 3c of Algorithm 2, but if it does not, then the probability that $\hat{\ell} \leq (1-\epsilon)\tilde{\ell}$ in any particular execution is upper bounded by $O(1/\log n)$. Since the executions are independent, the expected running time is $O\left(\frac{n}{\ell(G)^{1/3}} + \min\left\{n^{1/2}, \frac{n^{3/2}}{\ell(G)^{1/2}}\right\}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon)$.

We thus have the following theorem.

THEOREM 2. *With probability at least 2/3, the aforementioned algorithm, which uses Algorithm 1 as a subroutine, returns an estimates $\hat{\ell}$ that satisfies $\hat{\ell} = (1 \pm \epsilon) \cdot \ell(G)$. The expected query complexity and running time of the algorithm are $O\left(\frac{n}{(\ell(G))^{1/3}} + \min\left\{n^{1/2}, \frac{n^{3/2}}{(\ell(G))^{1/2}}\right\}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon)$.*

3 Lower Bounds for Approximating the Number of Length-2 Paths

In the next theorem we state three lower bounds that together match our upper bound in terms of the dependence on n and $\ell(G)$. In what follows, when we refer to a multiplicative approximation algorithm for the number of length-2 paths, we mean an algorithm that outputs an estimate $\hat{\ell}$ that with high probability satisfies $\ell(G)/C \leq \hat{\ell} \leq C\ell(G)$ for some (predetermined) approximation factor C (where C may depend on the size of the graph). If C is a constant then the algorithm is a constant factor approximation algorithm.

THEOREM 3. *1. Any multiplicative approximation algorithm for the number of length-2 paths must perform $\Omega\left(\frac{n}{\ell^{1/3}(G)}\right)$ queries.*

2. Any constant-factor approximation algorithm for the number of length-2 paths must perform $\Omega(\sqrt{n})$ queries when the number of length-2 paths is $O(n^2)$.

3. Any constant-factor approximation algorithm for the number of length-2 paths must perform $\Omega\left(\frac{n^{3/2}}{\ell^{1/2}}\right)$ queries when the number of length-2 paths is $\Omega(n^2)$.

3.1 Proof of Item 1 in Theorem 3. To establish the first item in Theorem 3 we show that every n and for every value of ℓ, there exists a family of n-vertex graphs for which the following holds. For each graph G in the family we have that $\ell(G) = \Theta(\ell)$, but it is not possible to distinguish with high constant probability between a random graph in the family and the empty graph (for which $\ell(G) = 0$). Each graph in the family simply consists of a clique of size $b = \lceil \ell^{1/3} \rceil$ and an independent set of size $n - b$. The number of length-2 paths in the graph is $b \cdot \binom{b-1}{2} = \Theta(\ell)$. However, in order to distinguish between a random graph in the family and the empty graph it is necessary to perform a query on a vertex in the clique. The probability of hitting such a vertex in $o\left(\frac{n}{\ell^{1/3}(G)}\right)$ queries is $o(1)$.

3.2 Proof of Item 2 in Theorem 3. Since we have already established in Item 1 in Theorem 3 that there is a lower bound of $\Omega\left(\frac{n}{\ell^{1/3}(G)}\right)$, and since for $\ell \leq n^{3/2}$ we have that $\frac{n}{\ell^{1/3}(G)} \geq n^{1/2}$, we may consider the case that $\ell(G) > n^{3/2} > n$. To establish Item 2 in Theorem 3 we show that every n, every constant c and every $n < \ell < (n/2c)^2$ there exist two families of n-vertex graphs for which the following holds. In both families the number of length-2 paths is $\Theta(\ell)$, but in one family this number is a factor c larger than in the other family. However, it is not possible to distinguish with high constant probability between a graph selected randomly in one family and a graph selected randomly in the other family using $o(\sqrt{n})$ queries. We first present two families that include some graphs with multiple edges and self-loops, and then modify the construction

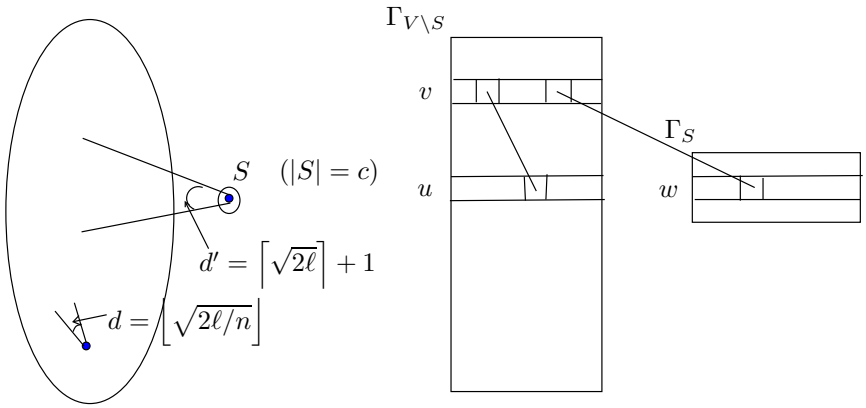

Figure 1: An illustration for the proof of Item 2 in Theorem 3. On the left-hand-side is a graph in $\mathcal{G}_2$, and on the right-hand-side are the corresponding neighborhood tables, $\Gamma_{V\setminus S}$ and Γ_S. Each row in $\Gamma_{V\setminus S}$ corresponds to a vertex in $V \setminus S$ and each row in Γ_S corresponds to a vertex in S. A connecting line between a pair of entries in the two tables indicates that there is an edge between the two corresponding vertices.

to obtain simple graphs.

The graph families. In the first family, denoted $\mathcal{G}_1$, each graph is determined by $d = \left\lfloor \sqrt{2\ell/n} \right\rfloor$ matchings. Thus, each vertex has degree $d = \left\lfloor \sqrt{2\ell/n} \right\rfloor$ and $\ell(G) = n \cdot \binom{d}{2} < \ell$. A random graph in $\mathcal{G}_1$ is determined by simply selecting d random matchings. In the second family, denoted $\mathcal{G}_2$, each graph is determined as follows. There is a small subset, S, of c vertices, where each vertex in S has degree $d' = \left\lceil \sqrt{2\ell} \right\rceil + 1$, and each vertex in $V \setminus S$ has degree $d = \left\lfloor \sqrt{2\ell/n} \right\rfloor$ (like all vertices in the graph in $\mathcal{G}_1$). If we view each vertex in S as having d' *ports* (one for each incident edge) and each vertex in $V \setminus S$ as having d ports, then a graph is the family $\mathcal{G}_2$ is defined by a perfect matching between the $(n-c)\cdot d + c \cdot d'$ ports (we assume this number is even, otherwise, d and d' can be slightly modified). For an illustration, see the left-hand-side of Figure 1. Thus, $\ell(G) > c \cdot \binom{d'}{2} = c \cdot \binom{\lceil \sqrt{2\ell} \rceil + 1}{2} > c\ell$.

Processes that construct graphs in the families. In order to show that it is hard to distinguish between graphs selected randomly from the two families in $o(\sqrt{n})$ queries, we follow [16, 25] and define two processes, $\mathcal{P}_1$ and $\mathcal{P}_2$, that interact with an approximation algorithm $\mathcal{A}$. The process $\mathcal{P}_1$ answers the queries of $\mathcal{A}$ while constructing a random graph in $\mathcal{G}_1$, and the process $\mathcal{P}_2$ answers the queries of $\mathcal{A}$ while constructing a random graph in $\mathcal{G}_2$. We consider the distributions over the respective *query-answer histories*, $\langle (q_1, a_1), \ldots, (q_t, a_t) \rangle$, and show that for histories of length $o(\sqrt{n})$ the distributions are very close, implying that $\mathcal{A}$ must have a high failure probability if it performs only $o(\sqrt{n})$ queries. Details follow.

For simplicity we assume that for every vertex that appears in either a neighbor query or an answer to such a query, both processed give the degree of the vertex "for free", so there is no need for degree queries. We also assume that an answer u to a neighbor query (v, i) comes with the label i' of the edge from u's side of the edge. Clearly any lower bound under these assumptions gives a lower bound without the assumptions.

The Process $\mathcal{P}_1$. The process $\mathcal{P}_1$ maintains an $n \times d$ table Γ. A graph in $\mathcal{G}_1$ corresponds to a perfect matching between the table entries. That is, if there is an edge (v, u) in the graph, and the edge is labeled i from v's side and i' from u's side, then $\Gamma(v, i) = (u, i')$ and $\Gamma(u, i') = (v, i)$. Thus, a random selection of a graph in $\mathcal{G}_1$ corresponds to a random selection of a perfect matching between the entries of Γ. Such a matching can be constructed iteratively, where in each iteration an unmatched entry in the table is selected *arbitrarily* and matched to a *uniformly selected* yet-unmatched entry. The process $\mathcal{P}_1$ fills the entries of Γ in the course of answering the queries of the algorithm $\mathcal{A}$: Given a query $q_{t+1} = (v, i)$, the process $\mathcal{P}_1$ answers with a uniformly selected yet-unmatched entry, (u, i')

The Process $\mathcal{P}_2$. The process $\mathcal{P}_2$ maintains two tables: one $(n - c) \times d$ table, $\Gamma_{V\setminus S}$, and one $c \times d'$ table, Γ_S. The rows of $\Gamma_{V\setminus S}$ correspond to vertices in $V \setminus S$, and the rows of Γ_S correspond to vertices in S. A random graph in $\mathcal{G}_2$ can be determined in the following iterative manner. In each step a pair (v, i) is selected arbitrarily among all pairs such that: (1) either there is already a row labeled by v in one of the two tables

but the entry (v,i) is yet-unmatched, or (2) there is no row labeled by v. In the latter case, we first select, uniformly at random, a yet-unlabeled row in one of the two tables, and label it by v. We then select, uniformly at random, a yet-unmatched entry in one of the two tables. If the row of the selected entry is yet-unlabeled, then we give it a random label (among all yet-unused labels in $\{1,\ldots,n\}$).

The process $\mathcal{P}_2$ fills the entries of $\Gamma_{V\setminus S}$ and Γ_S in the course of answering the queries of the algorithm $\mathcal{A}$ in the following manner. First note that once a vertex v that appears in either a query or an answer is determined to belong to S, then we may assume that $\mathcal{A}$ terminates, since it has evidence to distinguish between the two families (recall that the degree of a vertex is revealed when it appears in a query or an answer). Now, given a query $q_{t+1}=(v,i)$, if v is a vertex that was not yet observed in the query-answer history (that is, it does not label any row), then $\mathcal{P}_2$ first determines whether it belongs to S or to $V\setminus S$, that is, if it labels a row in Γ_S or in $\Gamma_{V\setminus S}$. Let the number of vertices already determined to be in $V\setminus S$ be denoted by b (so that $b\leq 2t$). With probability $\frac{c}{n-b}$ the vertex v is determined to belong to S (at which point $\mathcal{A}$ can terminate) and with probability $1-\frac{c}{n-b}$ it is determined to belong to $V\setminus S$, so that it labels a yet-unlabeled row in $\Gamma_{V\setminus S}$. Next, a yet-unmatched entry is selected uniformly among all such entries in $\Gamma_{V\setminus S}$ and Γ_S. If the selected entry is in Γ_S then $\mathcal{A}$ can terminate. Otherwise, let i' be the column to which the entry belongs (in $\Gamma_{V\setminus S}$). If the entry belongs to a row that is already labeled by some $u\in\{1,\ldots,n\}$, then $\mathcal{P}_2$ answers (u,i'), and if the row is unlabeled then $\mathcal{P}_2$ uniformly selects a label $u\in\{1,\ldots,n\}$ among all yet-unused row labels, and answers (u,i').

Analyzing the distributions on query-answer histories. Consider $\mathcal{P}_2$, and observe that if the number of queries performed is $o(\sqrt{n})$ then the probability that a vertex v in a query (v,i) is determined to belong to S is $o(\sqrt{n})\cdot\frac{c}{n-c-o(\sqrt{n})}=o\left(\frac{1}{\sqrt{n}}\right)$. The second observation about $\mathcal{P}_2$ is that for every $t=o(\sqrt{n})$, the probability that the answer to a query (v,i) will be (u,i') where $u\in S$ is upper-bounded by $\frac{c\cdot d'}{(n-c)\cdot d-2t}=O\left(\frac{1}{\sqrt{n}}\right)$, and so the probability that such an event occurs in a sequence of $o(\sqrt{n})$ queries is $o(1)$.

Finally, for both processes, the probability that an an answer to a query $q_{t+1}=(v,i)$ is (u,i') for u that has already appeared in the query-answer history is upper bounded by $\frac{2t}{n}=o\left(\frac{1}{\sqrt{n}}\right)$, and so the probability that such an event occurs in a sequence of $o(\sqrt{n})$ queries is $o(1)$. Therefore, in both processes, if the number of queries performed is $o(\sqrt{n})$, then for any algorithm $\mathcal{A}$, with probability $1-o(1)$ the sequence of answers to the queries of $\mathcal{A}$ is a sequence of uniformly selected distinct pairs (u,i'). This implies that the statistical distance between the corresponding distributions on query-answer histories is $o(1)$, and so it is not possible to distinguish between a random graph in $\mathcal{G}_1$ and a random graph in $\mathcal{G}_2$ with probability greater than $\frac{1}{2}+o(1)$.

The issue of multiple edges. As defined above, the graphs may have multiple edges. In order to avoid multiple edges, the distribution on answers to queries given any particular history is not as simple as when allowing multiple edges. However, the main observation is that the probability of selecting an entry that belongs to a row that already has some matched entries can only decrease. The only negative effect on our bounds is in the upper bound on the probability that the answer to a query (v,i) will be (u,i') where $u\in S$, which we can bound by $\frac{c\cdot d'}{n\cdot d-2t\cdot d}$, which is still $O\left(\frac{1}{\sqrt{n}}\right)$ for $t=o(\sqrt{n})$.

3.3 Proof of Item 3 in Theorem 3. Similarly to the proof of Item 2 in Theorem 3, to establish Item 3 in Theorem 3 we show that every n, every constant c and every $\ell=\Omega(n^2)$, $\ell<n^3/(16c^2)$, there exist two families of n-vertex graphs for which the following holds. In both families the number of length-2 paths is $\Theta(\ell)$, but in one family this number is a factor c larger than in the other family. However, it is not possible to distinguish with high constant probability between a graph selected randomly in one family and a graph selected randomly in the other family using $o\left(\frac{n^{3/2}}{\ell^{1/2}}\right)$ queries. (Note that when $\ell=\Omega(n^3)$, and in particular, $\ell\geq n^3/(16c^2)$, the lower bound is $\Omega(1)$, which is trivial.)

The first family, $\mathcal{G}_1$, is identical to the one defined in the proof of Item 2 in Theorem 3. That is, each graph is determined by $d=\left\lfloor\sqrt{2\ell/n}\right\rfloor$ matchings so that each vertex has degree d and $\ell(G)=n\cdot\binom{d}{2}<\ell$. In the second family, denoted $\mathcal{G}_2$, each graph is defined as follows. There is subset, S, of $s=\left\lceil\frac{4c\ell}{n^2}\right\rceil$ vertices, and a complete bipartite graph between S and $V\setminus S$. In addition, there are $d-s$ perfect matchings between vertices in $V\setminus S$. For an illustration, see Figure 2. Thus, each vertex in $V\setminus S$ has degree d, just like in $\mathcal{G}_1$. Now, for every $G\in\mathcal{G}_2$, using our assumption that $\ell<n^3/(16c^2)$ so that $s<n/4$,

$$\ell(G)\geq s\cdot\binom{n-s}{2}>s\cdot\binom{3n/4}{2}=\left\lceil\frac{4c\ell}{n^2}\right\rceil\cdot\binom{3n/4}{2}>c\ell\,. \tag{3.26}$$

The argument for proving the no algorithm can distinguish with high constant probability between a graph selected randomly in $\mathcal{G}_1$ and a graph selected randomly in $\mathcal{G}_2$, is similar to the one presented in the

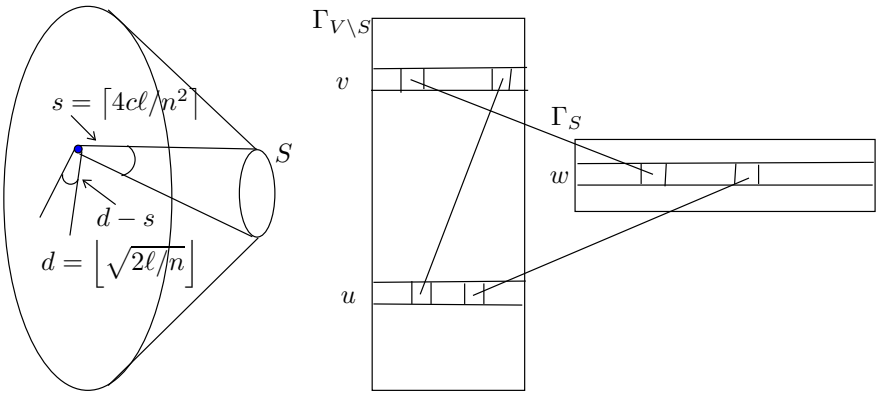

Figure 2: An illustration for the proof of Item 3 in Theorem 3. On the left-hand-side is a graph in $\mathcal{G}_2$, and on the right-hand-side are the corresponding tables, $\Gamma_{V\setminus S}$ and Γ_S. A connecting line between a pair of entries indicates that there is an edge between the two corresponding vertices.

proof of Item 2 in Theorem 3, and is actually somewhat simpler. As in the proof of Item 2 in Theorem 3, we define two processes, $\mathcal{P}_1$ and $\mathcal{P}_2$, where $\mathcal{P}_1$ is exactly as defined in the proof of Item 2 in Theorem 3.

The process $\mathcal{P}_2$. The process $\mathcal{P}_2$ maintains an $(n-s) \times d$ table, $\Gamma_{V\setminus S}$, and an $s \times (n-s)$ table, Γ_S. The rows of $\Gamma_{V\setminus S}$ corresponds to vertices in $V \setminus S$ and the rows in Γ_S correspond to vertices in S. A graph in $\mathcal{G}_2$ is determined by a perfect matching between the union of the entries in the two tables, where each row in $\Gamma_{V\setminus S}(v)$ contains exactly s entries that are matched with entries of Γ_S, one from each row. Here too we may assume that once a vertex v that appears in either a query or an answer is determined to belong to S (i.e., to label a row in Γ_S), then $\mathcal{A}$ terminates, since it has evidence to distinguish between the two families.

Given a query $q_{t+1} = (v, i)$, if v is a vertex that was not yet observed in the query-answer history, then $\mathcal{P}_2$ first determines whether it belongs to S or to $V \setminus S$. Let the number of vertices already determined to be in $V \setminus S$ be denoted by b (so that $b \le 2t$). With probability $\frac{s}{n-b}$ the vertex v is determined to belong to S (at which point $\mathcal{A}$ can terminate) and with probability $1 - \frac{s}{n-b}$ it is determined to belong to $V \setminus S$, so that it labels a randomly chosen yet-unlabeled row in $\Gamma_{V\setminus S}$. Next, the process decides whether the entry (v, i) corresponds to an edge whose other endpoint is in S or in $V \setminus S$. Let $b(v)$ be the number of entries in the row of v that have already been determined. Then, with probability $\frac{s}{d-b(v)}$ the entry is matched to a uniformly selected entry in Γ_S (so that $\mathcal{A}$ can terminate), and with probability $1 - \frac{s}{d-b(v)}$ it is matched to a yet-unmatched entry in Γ_S. This entry is selected as follows. For each row r in $\Gamma_{V\setminus S}$ (labeled or unlabeled), let $b(r)$ be the number of entries in r that are already matched. Then a row r is selected with probability $\frac{d-b(r)-s}{\sum_r (d-b(r)-s)}$. If the row is yet unlabeled then $\mathcal{P}_2$ uniformly selects a label $u \in \{1, \ldots, n\}$ among all yet-unused row labels. The index i' is selected uniformly among all yet-unmatched entries in the row r.

Analyzing the distributions on query-answer histories. Consider $\mathcal{P}_2$, and observe that if the number of queries performed is $o(n^{3/2}/\ell^{1/2})$ then the probability that a vertex v in a query (v, i) is determined to belong to S is

$$\begin{aligned} & o(n^{3/2}/\ell^{1/2}) \cdot \frac{s}{n - o(n^{3/2}/\ell^{1/2})} \\ = \;& o(n^{3/2}/\ell^{1/2}) \cdot \frac{\lceil (4c\ell)/n^2 \rceil}{n} \\ = \;& o\left(\frac{\ell^{1/2}}{n^{3/2}}\right) = o(1) \,. \end{aligned} \tag{3.27}$$

The second observation about $\mathcal{P}_2$ is that for every $t = o(n^{3/2}/\ell^{1/2})$, the probability that the answer to a query (v, i) will be (u, i') where $u \in S$ is upper-bounded by

$$\begin{aligned} \frac{s}{d - b(v)} &= \frac{\lceil (4c\ell)/n^2 \rceil}{\left\lfloor \sqrt{2\ell/n} \right\rfloor - o(n^{3/2}/\ell^{1/2})} \\ &= O(\ell^{1/2}/n^{3/2}) \,, \end{aligned} \tag{3.28}$$

and so the probability that such an event occurs in a sequence of $o(n^{3/2}/\ell^{1/2}))$ queries is $o(1)$. Finally, for both processes, the probability that an an answer to a query $q_{t+1} = (v, i)$ is (u, i') for u that has already appeared in the query-answer history is upper-bounded by $\frac{2t}{n} = o\left(n^{1/2}\ell^{1/2}\right)$, and so the probability that such

an event occurs in a sequence of $o(n^{3/2}/\ell^{1/2})$ queries is $o(n^2/\ell) = o(1)$.

Therefore, in both processes, if the number of queries performed is $o(n^{3/2}/\ell^{1/2})$, then for any algorithm $\mathcal{A}$, with probability $1 - o(1)$ the sequence of answers to the queries of $\mathcal{A}$ is a sequence of uniformly selected distinct pairs (u, i'). This implies that the statistical distance between the corresponding distributions on query-answer histories is $o(1)$, and so it is not possible to distinguish random graphs from the two families with probability greater than $\frac{1}{2} + o(1)$. The issue of multiple edges is dealt with as in the proof of Item 2 in Theorem 3,

4 Extending the Algorithm to Stars

In this section we give the algorithm for estimating the number of s-stars. Its analysis, which is similar to the analysis of the algorithm for estimating length-2-paths (2-stars) can be found in the full version of this paper [18].

Recall that an *s-star* is a graph over $s+1$ vertices in which one single vertex (the star *center*) is adjacent to all other vertices (and there are no edges between the other vertices). In particular, a length-2 path is a 2-star. The algorithm for approximating the number of s-stars for $s > 2$ is a natural extension of the algorithm we presented for the case of $s = 2$, and its analysis is very similar. Here we describe the modifications in the algorithm. Recall that $\nu_s(G)$ denotes the number of s-stars in a graph G. Here too we assume the algorithm is given a rough estimate $\tilde{\nu}_s$ for $\nu_s(G)$, such that $\frac{1}{2}\nu_s(G) \leq \tilde{\nu}_s \leq 2\nu_s(G)$, and this assumption is removed in the same manner as in the case of length-2 paths. We assume for simplicity that s is a constant, though it can also be a very slowly growing function of n (since the dependence on s is exponential).

ALGORITHM 3. (Estimating the number of s-stars in $\boldsymbol{G = (V, E)}$)
Input: ϵ, s, and $\tilde{\nu}_s$.

1. *Let $\beta \stackrel{\text{def}}{=} \frac{\epsilon}{32s}$, $t \stackrel{\text{def}}{=} \left\lceil \log_{(1+\beta)} n \right\rceil$, and*
$$\theta_1 \stackrel{\text{def}}{=} \frac{\epsilon^{\frac{s}{s+1}} \tilde{\nu}_s^{\frac{1}{s+1}}}{c_1(s) t^{\frac{2s}{s+1}}} ,$$
where $c_1(s)$ will be set in the analysis.
2. *Uniformly and independently select $\Theta\left(\frac{n}{\theta_1} \cdot \frac{\log t}{\epsilon^2}\right)$ vertices from V, and let S denote the multiset of selected vertices (that is, we allow repetitions).*
3. *For $i = 0, \ldots, t$, determine $S_i = S \cap B_i$ by performing a degree query on every vertex in S.*
4. *Let $L = \left\{ i : \frac{|S_i|}{|S|} \geq 2\frac{\theta_1}{n} \right\}$. If $\max_{i \in L} \left\{ \binom{(1+\beta)^{i-1}}{s} \cdot \theta_1 \right\} > 4\tilde{\nu}_s$ then terminate.*
5. *For each $i \in L$ run a slight variant of Algorithm 2 (that is described below) to get estimates $\{\hat{e}_{i,j}\}_{j \notin L}$ for $\{|E_{i,j}|\}_{j \notin L}$.*
6. *Output*
$$\hat{\nu}_s = \sum_{i \in L} n \cdot \frac{|S_i|}{|S|} \cdot \binom{(1+\beta)^i}{s} + \sum_{j \notin L} \frac{1}{s} \sum_{i \in L} \hat{e}_{i,j} \binom{(1+\beta)^j - 1}{s-1} .$$

The variant of Algorithm 2 used to get the estimates $\{\hat{e}_{i,j}\}_{j \notin L}$ is the following. For each $0 \leq p \leq i$ let

$$\theta_2(p) \stackrel{\text{def}}{=} \frac{\epsilon^{\frac{s+1}{s}} \tilde{\nu}_s^{\frac{1}{s}}}{c_2(s) t^{\frac{2s+1}{s}} (1+\beta)^{\frac{p}{s}}} , \tag{4.29}$$

where $c_2(s)$ grows at most exponentially with s. The minimum value p_0 of p is still the smallest value of p satisfying $\frac{1}{4}\theta_2(p+1) \leq n$. The sample size $s^{(p)}$ is still

$$s^{(p)} = \Theta\left(\frac{n}{\theta_2(p)} \left(\frac{t}{\beta}\right)^2 \log t \right) , \tag{4.30}$$

and in Step 3c we have:

3c. If $|S_i^{(p)}| < \frac{s^{(p)}}{n} \cdot \frac{1}{4(1+\beta)} \theta_2(p)$, then go to Step 4. Else, if $|S_i^{(p)}| > \frac{s^{(p)}}{n} \cdot \frac{4\tilde{\nu}_s}{\binom{(1+\beta)^{i-1}}{s}}$ then terminate.

THEOREM 4. *If $\frac{1}{2}\nu_s(G) \leq \tilde{\nu}_s \leq 2\nu_s(G)$, then with probability at least $2/3$, the output, $\hat{\nu}_s$, of Algorithm 3 satisfies $\hat{\nu}_s = (1 \pm \epsilon) \cdot \nu_s(G)$. The query complexity and running time of the algorithm are*

$$O\left(\frac{n}{\tilde{\nu}_s^{\frac{1}{s+1}}} + \min\left\{ n^{1-\frac{1}{s}}, \frac{n^{s-\frac{1}{s}}}{\tilde{\nu}_s^{1-\frac{1}{s}}} \right\} \right) \cdot \mathrm{poly}(\log n, 1/\epsilon) .$$

As noted previously, the assumption that the algorithm has an estimate $\tilde{\nu}_s$ for $\nu_s(G)$ is removed similarly to the way it was removed in the case of 2-stars.

5 Lower Bounds for Approximating the Number of s-Stars

We also have matching lower bounds similarly to what we had for the case of length-2 paths. For simplicity we state them for constant s but they can be extended to non-constant s.

THEOREM 5. *Let s be a constant.*

1. *Any (multiplicative) approximation algorithm for the number of s-stars must perform $\Omega\left(\frac{n}{(\nu_s(G))^{\frac{1}{s+1}}}\right)$ queries.*
2. *Any constant-factor approximation algorithm for the number of s-stars must perform $\Omega(n^{1-\frac{1}{s}})$ queries when the number of s-stars is $O(n^s)$.*
3. *Any constant-factor approximation algorithm for the number of s-stars must perform $\Omega\left(\frac{n^{s-\frac{1}{s}}}{(\nu_s(G))^{1-\frac{1}{s}}}\right)$ queries when the number of s-stars is $\Omega(n^s)$.*

The constructions are very similar to those used in the proofs of Items 1–3 of Theorem 3, and can be found in the full version of this paper [18].

6 Other Small Subgraphs

Other than s-stars, two additional natural extensions of length-2 paths are triangles and length-3 paths (or, more generally, length-L paths).

We first observe that there are lower bounds that are linear in the number of edges m when $m = \Theta(n)$, both for triangles and for length-3 paths. These lower bounds hold in the query model studied in this paper, that is, assuming the algorithm is allowed only degree queries and neighbor queries. Moreover, these lower bounds hold even if the algorithm is also allowed to sample edges uniformly. However, they do not hold if the algorithm is allowed vertex-pair queries, that is, if it may ask whether there is an edge between any two vertices u and v of its choice. Thus, it is possible that there are sublinear algorithms for approximating the number of these subgraphs assuming the algorithm is allowed vertex-pair queries. It can be verified that in the case of length-2 paths, and more generally, s-stars, the lower bounds hold even when allowed vertex-pair queries.[5]

THEOREM 6. *For $m = O(n)$ it is necessary to perform $\Omega(m)$ queries in order to distinguish with high constant probability between the case that a graph contains $\Theta(n)$ triangles and the case that it contains no triangles. This bound holds when neighbor and degree queries are allowed.*

[5] To verify this note that the lower bounds are essentially based on "hitting" a certain subset of vertices, either by querying one of these vertices or receiving one of them in an answer to a neighbor queries. If vertex-pair queries are allowed then the algorithm still needs to hit a vertex in this subset in one of its queries.

Proof. Consider the following two families of graphs. In the first family each graph consists of a complete bipartite graph between two vertices and all other vertices. In the second family each graph consists of a complete bipartite graph between two vertices and all other vertices but one, where this vertex is an isolated vertex. In addition there is an edge between the two vertices. Within each family the graphs differ only in the labeling of vertices and in the labeling of the edges incident to each vertex. Observe that in both families the two high-degree vertices have degree $n-2$ and the rest of the vertices have degree 2, with the exception of the single isolated vertex in the second family. By construction, each graph in the first family contains no triangles and each graph in the second family contains $n-3$ triangles. However, in order to distinguish between a random graph in the first family and a random graph in the second family it is necessary to either hit the isolated vertex in graphs of the second family, or to hit the edge between the two high-degree vertices in graphs of the second family, or to observe all neighbors of one of the high-degree vertices in the first family. In the latter case $n-2$ queries are necessary, and in the former cases $\Omega(n)$ queries are necessary (in order for one of these events to occur with constant probability).

THEOREM 7. *For $m = O(n)$ it is necessary to perform $\Omega(m)$ queries in order to distinguish with high constant probability between the case that a graph contains $\Theta(n^2)$ length-3 paths and the case that it contains no length-3 paths. This bound holds when neighbor and degree queries are allowed.*

Proof. Consider the following two families of graphs, where we assume for simplicity that n is even (otherwise there is an isolated vertex and the graph is defined over $n-1$ vertices where $n-1$ is even). In the first family each graph consists of two stars, where in each star there are $n/2$ vertices (including the center vertex). In the second family each graph consists of two stars, where in each star there are $n/2-1$ vertices (including the center vertex). In addition, there are two isolated vertices, and there is an edge between the two star centers. Graphs in the two families differ only in the labeling of vertices and in the labeling of the edges for the star centers. Observe that in both families the star centers have degree $n/2$. By construction, each graph in the first family contains no length-3 paths and each graph in the second family contains $\Theta(n^2)$ length-3 paths. However, in order to distinguish between a random graph in the first family and a random graph in the second family it is necessary to either hit one of the isolated vertices in graphs of the second family, or to hit the edge between the centers in graphs of the second family, or to observe all neighbors

of one of the centers in the first family. In the latter case $n/2$ queries are necessary, and in the former cases $\Omega(n)$ queries are necessary (in order for one of these events to occur with constant probability).

Acknowledgements We would like to thank several anonymous reviewers of SODA 2010 for their helpful comments.

References

[1] N. Alon, P. Dao, I. Hajirasouliha, F. Hormozdiari, and S. C. Sahinalp. Biomolecular network motif counting and discovery by color coding. *Bioinformatics*, 24(13):241–249, 2008.

[2] N. Alon and S. Gutner. Balanced families of perfect hash functions and their applications. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 435–446, 2007.

[3] N. Alon and S. Gutner. Balanced hashing, color coding and approximate counting. Technical Report TR09-012, Electronic Colloquium on Computational Complexity (ECCC), 2009.

[4] N. Alon, R. Yuster, and U. Zwick. Color coding. *Journal of the ACM*, 42:844–856, 1995.

[5] N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica*, 17:209–223, 1997.

[6] O. Amini, F. Fomin, and S. Saurabh. Counting subgraphs via homomorphisms. In *Automata, Languages and Programming: Thirty-Sixth International Colloquium (ICALP)*, pages 71–82, 2009.

[7] V. Arvind and V. Raman. Approximation algorithms for some parameterized counting problems. In *Proceedings of the 13th International Symposium on Algorithms and Computation (ISAAC)*, pages 453–464, 2002.

[8] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 16–24, 2008.

[9] A. Björklund, T. Husfeldt, P. Kasaki, and M. Koivisto. Counting paths and packings in halves. In *Proceedings of the Seventeenth Annual European Symposium on Algorithms (ESA)*, pages 578–586, 2009.

[10] B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM Journal on Computing*, 34(6):1370–1379, 2005.

[11] A. Czumaj, F. Ergun, 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*, 35(1):91–109, 2005.

[12] A. Czumaj and C. Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 175–183, 2004.

[13] B. Dost, T. Shlomi, N. Gupta, E. Ruppin, V. Bafna, and R. Sharan. QNet: A tool for querying protein interaction networks. In *Proceedings of the 11th Annual International Conference Research in Computational Molecular Biology (RECOMB)*, pages 1–15, 2007.

[14] R. Duke, H. Lefmann, and V. Rödl. A fast approximation algorithm for computing the frequencies of subgraphs in a given graph. *SIAM Journal on Computing*, 24(3):598–620, 1995.

[15] U. Feige. On sums of independent random variables with unbounded variance, and estimating the average degree in a graph. *SIAM Journal on Computing*, 35(4):964–984, 2006.

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

[17] O. Goldreich and D. Ron. Approximating average parameters of graphs. *Random Structures and Algorithms*, 32(4):473–493, 2008.

[18] M. Gonen, D. Ron, and Y. Shavitt. Counting stars and other small subgraphs in sublinear time. Technical Report TR09-083, Electronic Colloquium on Computational Complexity (ECCC), 2009.

[19] M. Gonen and Y. Shavitt. Approximating the number of network motifs. In *Proceedings of the 6th International Workshop On Algorithms And Models For The Web-Graph (WAW)*, pages 13–24, 2009.

[20] J. Grochow and M. Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In *Proceedings of the 11th Annual International Conference Research in Computational Molecular Biology (RECOMB)*, pages 92–06, 2007.

[21] D. Hales and S. Arteconi. Motifs in evolving cooperative networks look like protein structure networks. *Special Issue of ECCS'07 in The Journal of Networks and Heterogeneous Media*, 3(2):239–249, 2008.

[22] F. Hormozdiari, P. Berenbrink, N. Przulj, and S.C. Sahinalp. Not all scale-free networks are born equal: The role of the seed graph in ppi network evolution. *PLoS: Computational Biology*, 3(7):e118, 2007.

[23] R. Karp and M. Luby. Monte-carlo algorithms for enumeration and reliability problems. In *Proceedings of the Twenty-Fourth Annual Symposium on Foundations of Computer Science (FOCS)*, pages 56–64, 1983.

[24] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004.

[25] T. Kaufman, M. Krivelevich, and D. Ron. Tight bounds for testing bipartiteness in general graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004.

[26] I. Koutis. Faster algebraic algorithms for path and packing problems. In *Automata, Languages and Programming: Thirty-Fifth International Colloquium*

(ICALP), pages 575–586, 2008.

[27] I. Koutis and R. Williams. Limits and applications of group algebras for parameterized problems. In *Automata, Languages and Programming: Thirty-Sixth International Colloquium (ICALP)*, pages 653–664, 2009.

[28] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.

[29] 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)*, pages 327–336, 2008.

[30] M. Parnas and D. Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1–3):183–196, 2007.

[31] N. Przulj, D. G. Corneil, and I. Jurisica. Modeling interactome: Scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 2004.

[32] J. Scott, T. Ideker, R. Karp, and R. Sharan. Efficient algorithms for detecting signaling pathways in protein interaction networks. In *Proceedings of the 9th Annual International Conference Research in Computational Molecular Biology (RECOMB)*, pages 1–13, 2005.

[33] T. Shlomi, D. Segal, and E. Ruppin. QPath: a method for querying pathways in a protein-protein interaction network. *Bioinformatics*, 7:199, 2006.

[34] V. Vassilevska and R. Williams. Finding, minimizing, and counting weighted subgraphs. In *Proceedings of the Fourty-First Annual ACM Symposium on the Theory of Computing*, pages 455–464, 2009.

[35] S. Wernicke. Efficient detection of network motifs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):347–359, October 2006.

[36] R. Williams. Finding paths of length k in $o^*(2^k)$ time. *Information Processing Letters*, 109(6):315–318, 2009.