Approximating Average Parameters of Graphs

In Memory of Shimon Even (1935–2004)

Oded Goldreich*
Department of Computer Science
Weizmann Institute of Science
Rehovot, Israel.
oded.goldreich@weizmann.ac.il

Dana Ron†
Department of EE–Systems
Tel-Aviv University
Ramat-Aviv, Israel.
danar@eng.tau.ac.il

July 14, 2005

Abstract

Inspired by Feige (*36th STOC*, 2004), we initiate a study of sublinear randomized algorithms for approximating average parameters of a graph. Specifically, we consider the average degree of a graph and the average distance between pairs of vertices in a graph. Since our focus is on sublinear algorithms, these algorithms access the input graph via queries to an adequate oracle.

We consider two types of queries. The first type is standard neighborhood queries (i.e., *what is the i^{th} neighbor of vertex v?*), whereas the second type are queries regarding the quantities that we need to find the average of (i.e., *what is the degree of vertex v?* and *what is the distance between u and v?*, respectively).

Loosely speaking, our results indicate a difference between the two problems: For approximating the average degree, the standard neighbor queries suffice and in fact are preferable to degree queries. In contrast, for approximating average distances, the standard neighbor queries are of little help whereas distance queries are crucial.

Note: Part of this work (i.e., the material in Section 3.1) was posted on ECCC (as TR04-013).

Keywords: Sublinear-time algorithms, randomized approximation algorithms

*Part of this work was work done while being a fellow of the Radcliffe Institute for Advanced Study, Harvard University. The research was supported in part by the Israel Internet Association (ISOC-IL).

†Part of this work was done while being a fellow of the Radcliffe Institute for Advanced Study, Harvard University. The research was supported in part by the Israel Internet Association (ISOC-IL).

1 Introduction

In a recent work [6], Feige investigated the problem of estimating the average degree of a graph *when given direct access to the list of degrees* (of individual vertices). He observed two interesting ("phase transition") phenomena. Firstly, in contrast to the problem of estimating the average value of an arbitrary function $d : [n] \to [n-1]$ (where $[n] \stackrel{\text{def}}{=} \{1, ..., n\}$), sublinear-time approximations can be obtained when the function d represents the degree sequence of a simple graph over n vertices.[1] Secondly, whereas a $(2+\epsilon)$-approximation can be obtained in $O(\sqrt{n})$-time, for every constant $\epsilon > 0$, a better approximation factor cannot be achieved in sublinear time (i.e., a $(2-o(1))$-approximation requires time $\Omega(n)$).

Feige's work views the problem of estimating the average degree of a graph as a special case of estimating the average value of an arbitrary function $d : [n] \to [n-1]$. Our perspective is different: We view Feige's work as a sub-linear algorithm for a natural graph theoretic problem, which brings up two (open-ended) questions:

1. What type of operations (i.e., direct access queries to the input graph) are natural to consider for such an algorithm?

2. What other natural "average graph parameters" (i.e., averages of vertex-based quantities) are of interest?

In the following two subsections we briefly address these questions, and afterwards we present our results that refer to various combinations of "answers" to these questions.

1.1 Types of direct access queries

When viewing the problem of estimating the average degree in a graph as a special case of the problem of estimating the average value of an arbitrary function $d : [n] \to [n-1]$, it seems natural to restrict the algorithm to "degree queries". However, from the point of view of sublinear-time algorithms for graphs (cf., e.g., [8, 9, 11, 1, 10]), it is natural to allow also other types of queries to the graph. The most natural queries are neighbor queries; that is, queries of the form (v, i) that are answered by the i^{th} neighbor of v (or by a special symbol that indicates that v has less than i neighbors). In case of relatively dense graphs, it is also natural to consider adjacency queries (i.e., are vertices u and v adjacent in the graph). Thus, we consider two basic types of queries:

1. Standard neighbor (and adjacency) queries, which are natural in any algorithmic problem regarding graphs.

2. Problem-specific queries that associate values to vertices (or to sets of vertices), where our aim is to compute the average of these values. For example, in the case of approximating the average degree we consider degree queries.

We comment that degree queries can be emulated by a logarithmic number of neighbor queries (i.e., via binary search).

[1] Here we also assume that there are no isolated vertices in the graph (i.e., each vertex has degree at least 1).

1.2 Other natural averaging problems

In addition to the average degree of a graph, we consider two problems regarding distances in a graph. The first is approximating the all-pairs average distance in the graph, and the second is approximating the average distance of a fixed vertex to all the graph vertices. We refer to these problem by the terms all-pairs and single-source, respectively.

In addition to the standard neighbor queries, for the average distance approximation problems, we will also consider distance queries. That is, in both cases, we will consider queries of the form (u, v) that are answered by the distance between u and v in the graph.

1.3 Our results

Our results indicate that for one problem (i.e., approximating the average degree) augmenting the problem-specific oracle with neighbor queries helps, whereas for the other problems (i.e., approximating average distances) such an augmentation does not help. Moreover, as noted above, degree queries are not of great help (for approximating the average degree), whereas distance queries are crucial to approximating average distances in sublinear-time. In both cases, our algorithms do not use adjacency queries (and our lower bounds show that these queries do not help).

1.3.1 Approximating the Average Degree of a Graph

We present a sublinear algorithm that obtains an arbitrarily good approximation of the average degree, *while making only neighbor queries*. Specifically, for every constant $\epsilon > 0$, we obtain a $(1+\epsilon)$-approximation to the average degree of a graph $G = (V, E)$ in $\widetilde{O}(\sqrt{|V|})$-time, where the O-notation hides a polynomial dependence on ϵ.

Our result should be contrasted with Feige's results [6]: Recall that Feige showed that, when using only degree queries, a $(2-o(1))$-approximation (of the average degree of $G = (V, E)$) requires time $\Omega(|V|)$. Thus, neighbor queries are essential for sublinear-time algorithms that provide a $(2-o(1))$-approximation. On the other hand, he showed that (for every constant $\epsilon > 0$) a $(2+\epsilon)$-approximation can be obtained in $O(\sqrt{|V|})$-time (using only degree queries).

The running-time of our algorithm is essentially optimal: any constant-factor approximation of the average degree requires making $\Omega(\sqrt{|V|})$ queries of some graph $G = (V, E)$, even when allowed both neighbor and degree queries. Furthermore, a $(1+\epsilon)$-approximation requires $\Omega(\sqrt{|V|/\epsilon})$ queries.

The above represents a simplified account of the results. We recall that Feige [6] has provides his algorithm with a lower bound on the average degree of the input graph. This auxiliary input allows also to handle graphs that have isolated vertices (rather than assuming that each vertex has degree at least 1) and yields an improvement whenever the lower bound is better (than the obvious value of 1). Specifically, given a lower bound of ℓ (on the average degree), the complexity of Feige's algorithm is related to $\sqrt{|V|/\ell}$ rather than to $\sqrt{|V|}$. The same improvement holds also for our algorithms. Furthermore, we observe that our algorithms (as well as Feige's) can be adapted to work without this lower bound. Specifically, the complexity of the modified algorithm, which obtains no a priori information about the average degree, is related to $(|V|/\overline{d})^{1/2}$, where $\overline{d}$ denotes the actual average degree (which is, of course, not given to the algorithm). Thus, we get:

Theorem 1.1 *There exists an algorithm that makes only neighbor queries to the input graph and satisfies the following condition. On input $G = (V, E)$ and $\epsilon \in (0, 1)$, with probability at least 2/3,*

the algorithm halts within $\widetilde{O}((|V|/\overline{d})^{1/2} \cdot \mathrm{poly}(1/\epsilon))$ *steps and outputs a value in* $[\overline{d}, (1+\epsilon)\cdot\overline{d}]$*, where* $\overline{d} = 2|E|/|V|$.

Again, this running-time is essentially optimal in the sense that a $(1+\epsilon)$-approximation requires $\Omega((|V|/(\epsilon\overline{d}))^{1/2})$ queries, for every value of $|V|$ and $\overline{d} \in [2, o(|V|)]$ and $\epsilon \in [\omega(|V|^{-1/4}), o(|V|/\overline{d})]$.

1.3.2 Approximating Average Distances

We present a sublinear algorithm that obtains an arbitrarily good approximation of the average (all-pairs and single-source) distances, *while making* (only) *distance queries*. Specifically, we obtain a $(1+\epsilon)$-approximation of the (relevant) average distance of a graph $G = (V, E)$ in time $O(\sqrt{|V|}) \cdot \mathrm{poly}(1/\epsilon)$. Actually, as in the case of approximating the average degree, we obtained an improved performance as a function of the actual average distance.

Theorem 1.2 *There exists an algorithm that makes only distance queries to the input graph and satisfies the following condition. On input* $G = (V, E)$ *and* $\epsilon \in (0, 1)$*, with probability at least* $2/3$*, the algorithm halts within* $O((|V|/\overline{d}_G)^{1/2} \cdot \mathrm{poly}(1/\epsilon))$ *steps and outputs a value in* $[\overline{d}_G, (1+\epsilon)\cdot\overline{d}_G]$*, where* $\overline{d}_G$ *is the average of the all-pairs distances in* G*. A corresponding algorithm exists for the average distances to a given vertex* $s \in V$.

This running time is essentially optimal: any constant-factor approximation of the average distance in $G = (V, E)$ requires making $\Omega((|V|/\overline{d}_G)^{1/2})$ queries, even when allowed both distance and neighbor queries. Furthermore, a $(1+\epsilon)$-approximation requires $\Omega((|V|/(\epsilon\overline{d}_G))^{1/2})$ queries, for every value of $|V|$ and $\overline{d}_G = o(|V|)$ and $\epsilon = \omega(|V|^{-1})$.

We show that distance queries are essential for sublinear-time algorithms that provide any constant-factor approximation of the average distances. Specifically, *when using only neighbor queries*, a k-approximation of the average distance in $G = (V, E)$ requires making $\Omega(|E|/k^2 \log k)$ queries. In the case of the single-source problem, this means that (when using only neighbor queries) a constant-factor approximation is as hard to obtain as the exact value. In the case of the all-pairs problem, by emulating distance queries in a straightforward manner, we can obtain a $(1+\epsilon)$-approximation in time $O(\sqrt{|V|} \cdot |E|) \cdot \mathrm{poly}(1/\epsilon)$ *when using only neighbor queries*. For moderately sparse graphs, this yields an improvement over the straightforward approach of computing (or approximating) all pair-distances and computing the average of these $|V|^2$ values. Details follow.

If $|E| \ll |V|^{3/2}$ then our $O(\sqrt{|V|} \cdot |E|) \cdot \mathrm{poly}(1/\epsilon)$-time $(1+\epsilon)$-approximation is definitely preferable to computing the average of $|V|^2$ approximate values regardless of how the latter are obtained. On the other hand, if $|E| > |V|^{e_{\mathtt{mm}}-0.5}$, where $e_{\mathtt{mm}} \in [2, 2.376)$ is the matrix multiplication exponent (cf. [2]), then one can find all pair-distances as well as their average faster than the time it takes our algorithm approximates the latter (cf. [7, 12]). In the intermediate range[2] (of $|V|^{3/2} \gg |E| \gg |V|^{e_{\mathtt{mm}}-0.5}$, where $e_{\mathtt{mm}} - 0.5 < 1.876$), our algorithm should be compared against a host of all-pairs approximate distance algorithms and the preference may depend on additional parameters (e.g., the approximation sought and a priori bounds on the average all-pairs distance). Specific algorithms that may be relevant include those of [4, 3]. (The interested reader is referred to Zwick's survey [13] of algorithms for finding exact and approximate distances in graphs.)

[2]Indeed, the intermediate range exists provided $e_{\mathtt{mm}} > 2$ (or rather than $e_{\mathtt{mm}} = 2$ is not known).

2 Preliminaries

Throughout the work, all algorithms are probabilistic and have direct access to their input. That is, such algorithms are actually probabilistic oracle machines that have access to one or more oracles. These oracles will typically represent a graph in a way to be understood from the context. For example, we consider oracles that answer queries such as neighbor queries and degree queries. The explicit input to these algorithms will consist of relevant parameters that always include the number of vertices in the graph, which in turn determines the vertex set (i.e., for simplicity, we assume that all n-vertex graphs have $[n] \stackrel{\text{def}}{=} \{1, ..., n\}$ as their vertex set). As the basic definition of approximation algorithms, we use the following standard one.

Definition 2.1 *For $\epsilon > 0$, a $(1+\epsilon)$-*approximation *of a quantity $q : \{0,1\}^* \to (0, \infty)$ is an algorithm that on input x, with probability at least $2/3$, outputs a value in the interval $[q(x), (1+\epsilon) \cdot q(x)]$.*

The error probability can be decreased to 2^{-k} by invoking the basic algorithm for $O(k)$ times and outputting the median value. At times, when $\epsilon \ll 1$, for simplicity of presentation we allow the algorithm to output a value in the interval $[(1-\epsilon) \cdot q(x), (1+\epsilon) \cdot q(x)]$. (Indeed, the output can be "normalized" by division (by $1-\epsilon$).) Our algorithms will all be uniform in the sense that we actually present an algorithm that takes ϵ as a parameter.

When stating lower bounds that depend on several parameters, we mean that these bounds hold uniformly for all choices of these parameters (or all choices satisfying explicitly stated conditions). That is, when we say that a $(1+\epsilon)$-approximation of q requires $\Omega(f(n, \epsilon, p))$ queries, we means that there exists a constant $c > 0$ such for any possible value of the parameters n, ϵ and p and any $(1+\epsilon)$-approximation algorithm A of the quantity q, there exists an n-vertex graph G with $q(G) = p$ such that A makes at least $c \cdot f(n, \epsilon, p)$ queries. (Since all our lower bounds refer to the query complexity of algorithms, linear speed-up phenomena do not arise.)

Throughout this work, we assume that the neighbors of each vertex are listed in arbitrary order. This reasonable assumption facilitates the proofs of the lower bound, which can be modified to handle also the case where the said lists are sorted.

3 Approximating the Average Degree of a Graph

Let $G = (V, E)$ be a *simple* graph (i.e., having no parallel edges and no self-loops), where $|V| = n$, and let $d(v)$ denote the degree of vertex $v \in V$ in G. We denote by $\overline{d} \stackrel{\text{def}}{=} \frac{1}{n} \sum_{v \in V} d(v)$ the average degree in G. An algorithm for estimating $\overline{d}$ is allowed to perform two types of queries: *degree queries* and *neighbor queries*. Namely, for any vertex v of its choice the algorithm can obtain $d(v)$, and for any v and $j \leq d(v)$, the algorithm can obtain the j^{th} neighbor of v. Actually, it suffices to allow the algorithm to obtain a random neighbor of any desired (i.e., queried) vertex.

We start by describing an algorithm that is provided with an *a priori known lower bound* ℓ on the value of $\overline{d}$. We later eliminate the need for this a priori knowledge. We close this section with a proof that our algorithm has almost optimal running-time (when referring to its dependence on the size of the graph).

3.1 The Algorithm

Our algorithm is inspired by the work of Kaufman *et. al.* [10], and more specifically, by a subroutine presented in [10] for sampling edges "almost uniformly". The basic idea of our algorithm is to sample vertices and to put them into "buckets" according to their degrees such that in bucket B_i we have vertices with degree between $(1+\beta)^{i-1}$ and $(1+\beta)^i$ (where $\beta = \epsilon/c$ for some constant $c > 1$). If S is the sample, then we denote by S_i the subset of sampled vertices that belong to B_i. We will focus on the sets S_i that are *sufficiently large*, because we want $|S_i|/|S|$ to be a good approximation of $|B_i|/n$. Let us denote the set of the corresponding i's by L.

Suppose we take $(1/|S|)\sum_{i\in L}|S_i|(1+\beta)^{i-1}$ as our estimate for the average degree of the graph. Note that the expected value of $|S_i|/|S|$ is $|B_i|/n$ and that $(1/n)\sum_i |B_i|(1+\beta)^{i-1} \leq \overline{d}$. Hence, with high probability, for a sufficiently large sample S, we would be overestimating the average degree by a factor of at most $(1+\epsilon)$. The source of the overestimation is only the error in approximating $|B_i|/n$ by $|S_i|/|S|$. However, we may underestimate $\overline{d}$ by a factor of roughly 2. The reason is that the edges between large buckets and small buckets are only counted once, rather than twice, and the edges with both endpoints in small buckets are not counted at all. Yes, the "threshold of largeness" is set such that the number of vertices in small buckets is so small that we can discard all possible edges that have both end-points in small buckets. (This calls for taking a sample of size $\widetilde{O}(\sqrt{n})$, setting the threshold at $\mathrm{poly}(\log n)$, and concluding that the number of vertices in small buckets is at most $\sqrt{n}$.)

So far we have described a procedure that approximates $\overline{d}$ up to a factor of $2+\epsilon$ while using only degree queries (i.e., we obtain Feige's result [6] using a different analysis). To get beyond the "factor 2 barrier" we observe that the main source of approximation error is due to edges with one endpoint in a large bucket and the other endpoint in a small bucket. These edges were counted once (in our estimate for $\overline{d}$), whereas they need to be counted twice. Thus, we merely need to estimate the number of such edges, which can be done by estimating, for each large bucket, the fraction of edges that are incident to vertices in the bucket and whose other endpoint is in a small bucket. This estimate cannot be obtained using degree queries, but it can be obtained using "random neighbor" queries. Specifically, for every vertex v in a large S_i, we select uniformly a neighbor of v and check whether this neighbor resides in a small bucket. Adding our estimate of the number of edges between large buckets and small buckets to $(n/|S|)\sum_{i\in L}|S_i|(1+\beta)^{i-1}$ yields a $(1+\epsilon)$-approximation of $2|E|$ (and hence a $(1+\epsilon)$-approximation of $\overline{d} = 2|E|/n$).

We are now ready to present and analyze the algorithm in full detail. For $t = \lceil \log_{(1+\beta)} n \rceil + 1$, we define a partition of V into the following buckets:

$$B_i = \left\{ v : \; d(v) \in \left((1+\beta)^{i-1}, (1+\beta)^i \right] \right\}, \quad \text{for } i = 0, 1, \ldots, t-1 \; . \tag{1}$$

The algorithm refers to an a priori lower bound ℓ on $\overline{d}$, and the reader may think of $\ell = 1$ as in the foregoing motivating discussion. We will consider B_i to be large (and put $i \in L$) if the sample S contains at least $\Omega(\frac{\sqrt{\epsilon}}{t} \cdot \frac{|S|}{\sqrt{n/\ell}})$ representatives of B_i. For a large B_i, we let $\widetilde{\alpha}_i$ denote our approximation of the fractions of edges incident at B_i that have their other endpoint in a small bucket.

Average Degree Approximation Algorithm

1. Uniformly and independently select $K = \tilde{O}\left(\sqrt{n/\ell} \cdot \mathrm{poly}(1/\epsilon)\right)$ vertices from V, and let S denote the (multi-)set of selected vertices.
2. For $i = 0, 1, \ldots, \lceil \log_{(1+\beta)} n \rceil$, let $S_i = S \cap B_i$.
3. Let $L = \left\{ i : \frac{|S_i|}{|S|} \geq \frac{1}{t} \cdot \sqrt{\frac{\epsilon}{6} \cdot \frac{\ell}{n}} \right\}$, where $t \stackrel{\text{def}}{=} \lceil \log_{(1+\beta)} n \rceil + 1$.
4. For every $i \in L$ and every $v \in S_i$, select at random a neighbor u of v, and let $\chi(v) = 1$ if $u \in \bigcup_{j \notin L} B_j$, and $\chi(v) = 0$ otherwise. For every $i \in L$, let $\widetilde{\alpha}_i = |\{v \in S_i : \chi(v) = 1\}|/|S_i|$.
5. Output $\frac{1}{K} \cdot \sum_{i \in L} (1 + \widetilde{\alpha}_i) \cdot |S_i| \cdot (1+\beta)^i$.

Lemma 3.1 *For every $\epsilon < 1/2$ and $\beta \leq \epsilon/8$, the above algorithm outputs a value $\widetilde{\mathtt{d}}$ such that, with probability at least $2/3$, it holds that $(1-\epsilon) \cdot \overline{d} < \widetilde{\mathtt{d}} < (1+\epsilon) \cdot \overline{d}$.*

Proof: By definition of the buckets (i.e., B_i's), it holds that

$$\overline{d} \;\leq\; \frac{1}{n} \sum_{i=1}^{t} |B_i| \cdot (1+\beta)^i \;\leq\; (1+\beta) \cdot \overline{d}\,. \tag{2}$$

Let $\rho \stackrel{\text{def}}{=} (1/t)\sqrt{(\epsilon/8) \cdot \ell/n}$ be a density threshold. By our choice of the sample size K we have that with high probability,

$$\forall i \text{ s.t. } |B_i| \geq \rho \cdot n : \qquad \left(1 - \frac{\epsilon}{4}\right) \cdot \frac{|B_i|}{n} \;\leq\; \frac{|S_i|}{K} \;\leq \left(1 + \frac{\epsilon}{4}\right) \cdot \frac{|B_i|}{n} \tag{3}$$

and

$$\forall i \text{ s.t. } |B_i| < \rho \cdot n : \qquad \frac{|S_i|}{K} < \frac{1}{t}\sqrt{(\epsilon/6) \cdot \ell/n} \tag{4}$$

In particular, in the latter case we have that $i \notin L$. Thus, with high probability, for every $i \in L$ we have that $|S_i|/K$ is close to its expected value. Let us assume from this point on that this is in fact the case.

For $V_1, V_2 \subseteq V$, we denote by $E(V_1, V_2)$ the set of all *ordered* pairs of adjacent vertices with the first vertex in V_1 and the second vertex with V_2; that is,

$$E(V_1, V_2) \stackrel{\text{def}}{=} \{(v_1, v_2) : v_1 \in V_1 \;\&\; v_2 \in V_2 \;\&\; \{v_1, v_2\} \in E\}\,. \tag{5}$$

For each i, let $E_i \stackrel{\text{def}}{=} E(B_i, V)$; that is, E_i is the set of all *ordered* pairs of adjacent vertices such that the first vertex is in B_i. Thus, the E_i's are disjoint and *each edge contributes two pairs to the set* $\bigcup_i E_i$. Also,

$$|B_i| \cdot (1+\beta)^{i-1} \;<\; |E_i| \;\leq\; |B_i| \cdot (1+\beta)^i \tag{6}$$

Let $U \stackrel{\text{def}}{=} \{v \in B_i : \; i \notin L\}$ denote the set of vertices that reside in buckets that are deemed small by the sample S. Then, with high probability,

$$|U| \;\leq\; \left| \left\{ v \in B_i : \; |B_i| \leq (1/t) \cdot \sqrt{(\epsilon/4) \cdot n \cdot \ell} \right\} \right| \;\leq\; \sqrt{(\epsilon/4) \cdot n \cdot \ell} \tag{7}$$

because if $\frac{|B_i|}{n} \geq (1/t) \cdot \sqrt{(\epsilon/4) \cdot \ell/n}$ then $\frac{|S_i|}{|S|} \geq (1/t) \cdot \sqrt{(\epsilon/6) \cdot \ell/n}$. Now, let $E'_i \stackrel{\text{def}}{=} E(B_i, U) \subseteq E_i$ and $\alpha_i = |E'_i|/|E_i| \leq 1$. That is, for $i \in L$, the set E'_i contains only pairs of adjacent vertices with a single endpoint in $V \setminus U$, and the corresponding edge is counted only once in the sum $\sum_{i \in L} |E_i|$ (and similarly in $\sum_{i \in L} |E'_i|$). Thus:

$$\sum_{i \in L} |E'_i| = |E(V \setminus U, U)| \quad \text{and} \quad \sum_{i \in L} |E_i \setminus E'_i| = 2|E(V \setminus U, V \setminus U)| \tag{8}$$

By our choice of the sample size, K, we have that with high probability, for every $i \in L$ such that $\alpha_i \geq \epsilon/8$,

$$\left(1 - \frac{\epsilon}{4}\right) \cdot \alpha_i \;\leq\; \widetilde{\alpha}_i \;\leq\; \left(1 + \frac{\epsilon}{4}\right) \cdot \alpha_i \tag{9}$$

and if $\alpha_i < \epsilon/8$ then $\widetilde{\alpha}_i < \epsilon/4$.

Thus, with high probability, all the estimates that the algorithm has are quite good, in which case the following holds:

$$\begin{aligned}
\widetilde{\mathtt{d}} &= \frac{1}{K} \cdot \sum_{i \in L} (1 + \widetilde{\alpha}_i) \cdot |S_i| \cdot (1 + \beta)^i && (10)\\
&\leq \frac{1}{n} \cdot \sum_{i \in L} (1 + \widetilde{\alpha}_i) \cdot \left(1 + \frac{\epsilon}{4}\right) \cdot |B_i| \cdot (1 + \beta)^i && (11)\\
&\leq \frac{1 + \epsilon/4}{n} \cdot \Big(\sum_{\substack{i \in L \\ \alpha_i \geq \epsilon/8}} (1 + (1 + \epsilon/4) \cdot \alpha_i) \cdot (1 + \beta) \cdot |E_i| \\
&\qquad + \sum_{\substack{i \in L \\ \alpha_i < \epsilon/8}} (1 + \epsilon/4) \cdot (1 + \beta) \cdot |E_i| \Big) && (12)\\
&< \frac{(1 + \epsilon/4)^2 \cdot (1 + \beta)}{n} \cdot \sum_{i \in L} (1 + \alpha_i) \cdot |E_i| && (13)
\end{aligned}$$

where Eq. (11) uses our assumption that $|S_i|/K$ is close to its expected value for every $i \in L$, and Eq. (12) uses Eq. (6) and our assumption on the estimates $\widetilde{\alpha}_i$. Similarly,

$$\widetilde{\mathtt{d}} \geq \frac{(1 - \epsilon/4)^2}{n} \cdot \sum_{i \in L} (1 + \alpha_i) \cdot |E_i| \tag{14}$$

Using $\beta \leq \epsilon/8$ and $|E'_i| = \alpha_i \cdot |E_i|$, we have

$$\begin{aligned}
\widetilde{\mathtt{d}} &= \frac{(1 \pm (\epsilon/4))^2 \cdot (1 \pm \epsilon/8)}{n} \cdot \sum_{i \in L} (1 + \alpha_i) \cdot |E_i| && (15)\\
&= \frac{1 \pm (3\epsilon/4)}{n} \cdot \Big(\sum_{i \in L} |E_i \setminus E'_i| + \sum_{i \in L} |E'_i| + \sum_{i \in L} \alpha_i \cdot |E_i| \Big) && (16)\\
&= \frac{1 \pm (3\epsilon/4)}{n} \cdot \Big(\sum_{i \in L} |E_i \setminus E'_i| + 2 \sum_{i \in L} |E'_i| \Big) && (17)
\end{aligned}$$

Using Eq. (8) it follows that

$$\widetilde{\mathtt{d}} = \frac{1 \pm (3\epsilon/4)}{n} \cdot \Big(2|E(V \setminus U, V \setminus U)| + 2|E(V \setminus U, U)| \Big) \tag{18}$$

$$= \frac{1 \pm (3\epsilon/4)}{n} \cdot \left(2|E(V,V)| - 2|E(U,U)|\right) \quad (19)$$

$$= \frac{1 \pm (3\epsilon/4)}{n} \cdot \left(\overline{d}n \pm |U|^2\right) \quad (20)$$

Recalling (cf. Eq. (7)) that $|U|^2 \leq (\epsilon/4) \cdot \ell n$ (and $\ell \leq \overline{d}$), we get $\widetilde{\mathtt{d}} = (1 \pm \epsilon) \cdot \overline{d}$. The lemma follows. ■

Working without a degree lower bound. For sake of simplicity, we start by modifying the algorithm so that when given a valid lower bound ℓ, it does not output an overestimation of the average degree (except with small probability). This is done by simply decreasing the output by a factor of $1+\epsilon$. Thus, the output, $\widetilde{\mathtt{d}}$, of the algorithm satisfies $\Pr[(1-2\epsilon)\overline{d} < \widetilde{\mathtt{d}} < \overline{d}] \geq 2/3)$. Furthermore, by $O(1) + \log\log n$ repetitions, we may reduce the probability of error to below $1/(6\log n)$.

An interesting feature of our algorithm is that, with high probability, it does not output an overestimate of $\overline{d}$ even in case it is invoked with a parameter ℓ that is *higher* than the average degree $\overline{d}$ (i.e., is not a valid lower bound). To verify this feature, observe that the only place in the analysis where we rely on the assumption $\ell \leq \overline{d}$ is in bounding the underestimation error (i.e., when bounding the total number of edges with both endpoints in U). (We comment that also Feige's algorithm [6] has this feature, but for different reasons.)

This feature allows us to present a version of our algorithm that does not require an a priori lower bound on the average degree. Specifically, let our algorithm be denoted by A. Then, starting with $\ell = n/2$, we may proceed in at most $2\log_2 n$ iterations as follows. We invoke A with the current value of ℓ, and let $\widetilde{\mathtt{d}}$ denote the output obtained. If $\widetilde{\mathtt{d}} \geq \ell$ then we halt and output $\widetilde{\mathtt{d}}$, otherwise we proceed to the next iteration while setting $\ell \leftarrow \ell/2$. In case all iterations were completed and still $\widetilde{\mathtt{d}} < \ell$ in the last iteration (i.e., $\widetilde{\mathtt{d}} < 1/2n$) then the graph must have no edges and we halt outputting $\widetilde{\mathtt{d}} = 0$.

Let $\ell_j = n/2^j$ be the parameter used in the j-th invocation of algorithm A, and let $\widetilde{\mathtt{d}}_j$ denote the corresponding output. Then, with probability at least $2/3$, for every iteration j that took place, it holds that $\widetilde{\mathtt{d}}_j \leq \overline{d}$ and if $\overline{d} \geq \ell_j$ then $\widetilde{\mathtt{d}}_j \geq (1-2\epsilon)\overline{d}$. In this case, assuming the graph contains any edges at all,[3] the algorithm will stop after at most $\log(n/\overline{d}) + O(1)$ iterations, and will output a value that is in the interval $[(1-2\epsilon)\overline{d}, \overline{d}]$. Thus, the overall running-time of the algorithm is $\mathrm{poly}(\epsilon^{-1}\log n) \cdot \sqrt{n/\overline{d}}$. **Theorem 1.1 follows**.

3.2 A Lower Bound

We observe that any constant approximation algorithm must perform $\Omega(\sqrt{n})$ queries. A more general bound, which depends also on the approximation parameter $\epsilon > 0$ and on the actual degree of the graph, is stated next.

Theorem 3.2 *For any n, $\overline{d} \in [2, o(n)]$ and $\epsilon \in (\omega(1/\overline{d}n), o(n/\overline{d}))$, a $(1+\epsilon)$-approximation of the average degree of $G = (V,E)$ requires $\Omega((n/(\epsilon\overline{d}))^{1/2})$ queries, where $\overline{d} = 2|E|/n$. This holds even if the algorithm is allowed neighbor and adjacency queries as well as degree queries.*

[3] In case the graph contains no edges, the algorithm will complete all iterations with no output (because $\overline{d} = 0 < \ell_j$ whereas $\widetilde{\mathtt{d}}_j = 0$ for each $j \leq 2\log n$), and thus output the correct value (i.e., 0) at the last step. In this case, the overall running-time of the algorithm is $\mathrm{poly}(\epsilon^{-1}\log n) \cdot n$. Clearly one can modify the algorithm so that its complexity is never more that $O(n)$ (i.e., the complexity of computing the exact average degree), by stopping once ℓ_j goes below $\mathrm{poly}(\epsilon^{-1}\log n)/n$ for an appropriate polynomial in $\log n$ and ϵ^{-1}.

Proof: For parameters n and $k \in (\overline{d}, o(n))$, we consider (randomly labeled versions of) two graphs. The first graph, denoted $G_{n,k}$, consists of a $\overline{d}$-regular subgraph over $n-k$ vertices and a $\overline{d}$-regular subgraph over the remaining k vertices (so that the two subgraphs are not connected). This graph has average degree $\overline{d}$. The second graph, denoted $G'_{n,k}$, consists of a $\overline{d}$-regular subgraph over $n-k$ vertices, and a clique over the remaining k vertices. The graph $G'_{n,k}$ has average degree $((n-k)\cdot\overline{d}+k(k-1))/n = (1-o(1))\cdot(\overline{d}+k^2/n)$. In particular, if we let $k = (\epsilon\overline{d}n)^{1/2}$, then we get an average degree of approximately $(1+\epsilon)\overline{d}$. However, in order to distinguish between (random labelings of) the two graphs, an algorithm must hit one of the k vertices in the small component, and hence $\Omega(n/k) = \Omega((n/(\epsilon\overline{d}))^{1/2})$ queries are necessary. ■

On non-simple graphs. Recall that we required upfront that the graph be simple (i.e., have no parallel edges or self-loops). We note that if parallel edges (or weighted edges) are allowed, then estimating the average degree of a graph requires $\Omega(n)$ queries (even when both degree queries and neighbor queries are allowed). Consider the following two graphs (or rather families of graphs): one graph consists of a cycle over all vertices (with a single edge between each pair of consecutive vertices) and the other consists of a cycle over $n-2$ vertices, and a pair of vertices with $c\cdot n$ parallel edges between them. The average degree in the first graph is 2 whereas the average degree in the second graph is roughly $2+c$. But distinguishing between the two (families of) graphs requires $\Omega(n)$ queries. Thus, there is a gap between the query complexity of estimating the average degree of simple graphs and non-simple graphs.

4 Approximating the Average Distance from a Single Source

Let $G=(V,E)$ be a simple undirected *connected* graph, where $n=|V|$ and $m=|E|$. For some given ("designated") vertex $s \in V$ we are interested in the average distance of s to the graph's vertices. That is, suppose we have access to an oracle that for any vertex $v \in V$ provides us with the distance, denoted $\mathrm{dist}_G(s,v)$, between s and v (in G). We would like to estimate the *average* distance, denoted $\overline{d}_G(s)$, of vertices in the graph from s; that is, $\overline{d}_G(s) = \frac{1}{n}\sum_{v\in V}\mathrm{dist}_G(s,v)$.

We first consider algorithms that make only distance queries. We present an algorithm (in Sec. 4.1) and a roughly matching lower bound (in Sec. 4.2). We later discuss the case in which the algorithm is also allowed neighbor queries (resp., only allowed neighbor queries); see Sec. 4.3 (resp., Sec. 4.4).

4.1 An Algorithm

We start with the basic version of our result.

Theorem 4.1 *There exists an algorithm that, for any given $\epsilon \in (0,1)$, makes $O\left(\sqrt{n}/\epsilon^2\right)$ distance queries and provides a $(1+\epsilon)$-approximation of the average distance of a given vertex to all graph vertices.*

The algorithm merely selects uniformly and independently $q = \Theta\left(\sqrt{n}/\epsilon^2\right)$ vertices $v_1,...,v_q$, performs the distance queries $\mathrm{dist}_G(s,v_i)$ for $i=1,...,q$, and outputs the average of the answers received. We show that, with high probability, the algorithm's output is an $(1+\epsilon)$-approximation of $\overline{d}_G$.

Let $d_{\max}$ be the maximum distance of any vertex v from s. For each value $i = 0, \ldots, d_{\max}$ let p_i denote the fraction of vertices at distance i from s. Let η be a random variable that takes value i with probability p_i, and let $\eta_1, \ldots, \eta_q$ be independent random variables that are distributed the same as η. By definition, $\mathrm{Exp}[\eta] = \overline{d}_G(s)$, and the output of our algorithm is distributed as $\frac{1}{q}\sum_{j=1}^{q}\eta_j$. Hence, we are interested in upper bounding the probability that $\frac{1}{q}\sum_{j=1}^{q}\eta_j$ deviates from its expected value, $\overline{d}_G(s)$, by more than $\epsilon \cdot \overline{d}_G(s)$. By Chebyshev's inequality

$$\Pr\left[\left|\frac{1}{q}\sum_{j=1}^{q}\eta_j - \mathrm{Exp}[\eta]\right| \geq \epsilon \cdot \mathrm{Exp}[\eta]\right] \leq \frac{\mathrm{Var}[\eta]}{q \cdot \epsilon^2 \cdot \mathrm{Exp}[\eta]^2} \tag{21}$$

Since $q = \Theta\left(\sqrt{n}/\epsilon^2\right)$, it suffices to show that the ratio between $\mathrm{Var}[\eta] = \mathrm{Exp}[\eta^2] - \mathrm{Exp}[\eta]^2$ and $\mathrm{Exp}[\eta]^2$ is $O(\sqrt{n})$. This follows from the next lemma, by using $\ell = 1/2$.

Lemma 4.2 *For η and p_i as defined above, $\mathrm{Exp}[\eta^2] \leq \sqrt{2n/\ell} \cdot \mathrm{Exp}[\eta]^2$, for any $\ell \leq \mathrm{Exp}[\eta]$.*

Since all distances are integers, and all are non-negative with the exception of $\mathrm{dist}(s,s) = 0$, we know that $\mathrm{Exp}[\eta] \geq \frac{n-1}{n} \geq 1/2$, which means that $\ell = 1/2$ can always be used. Thus, Theorem 4.1 follows from Lemma 4.2 (when specialized to the obvious case of $\ell = 1/2$), but we will use the more general statement of the lemma later.

Proof: By the definitions of η and $d_{\max}$,

$$\mathrm{Exp}[\eta^2] = \sum_{i=0}^{d_{\max}} p_i \cdot i^2 \leq d_{\max} \cdot \mathrm{Exp}[\eta] \tag{22}$$

We next observe that by definition of $d_{\max}$, for every $i \leq d_{\max}$ we have that $p_i \geq 1/n$, and so

$$\mathrm{Exp}[\eta] = \sum_{i=0}^{d_{\max}} p_i \cdot i > \frac{d_{\max}^2}{2n} \tag{23}$$

By multiplying the bound $\mathrm{Exp}[\eta] \geq \ell$ (provided in the lemma's hypothesis) by Eq. (23), we get that $\mathrm{Exp}[\eta]^2 \geq \frac{\ell \cdot d_{\max}^2}{2n}$ and so

$$\frac{\sqrt{\ell} \cdot d_{\max}}{\sqrt{2n}} \leq \mathrm{Exp}[\eta] \tag{24}$$

Finally, we multiply Equations (22) and (24) and get that

$$\mathrm{Exp}[\eta^2] \cdot \frac{\sqrt{\ell} \cdot d_{\max}}{\sqrt{2n}} \leq d_{\max} \cdot \mathrm{Exp}[\eta]^2 \tag{25}$$

and the lemma follows. ■

An improved algorithm. As in Section 3, a better algorithm can be obtained, provided we are given an a priori lower bound on the average distance. Denoting such a lower bound by ℓ, Lemma 4.2 implies that using a sample of size $q = \Theta(\epsilon^{-2} \cdot \sqrt{n/\ell})$ will do. Actually, as in Section 3, we do not actually need this lower bound, and the algorithm can function without it and perform as well. That is:

Theorem 4.3 *There exists an algorithm that, on input a graph $G=(V,E)$, a vertex s and parameter $\epsilon \in (0,1)$, makes $O(\epsilon^{-2}(n/\overline{d}_G(s))^{1/2})$ distance queries and provides a $(1+\epsilon)$-approximation of the average distance of vertices in G to s* (i.e., $\overline{d}_G(s)$).

Proof: The algorithm proceeds in $(1+\log n)$ iterations. In the i^{th} iteration, the algorithm behaves as if $\ell_i \stackrel{\text{def}}{=} n/2^i$ is a lower bound on $\overline{d}$. The algorithm uses the estimate obtained in the i^{th} iteration if and only if it exceeds ℓ_i by a factor of eight (otherwise it continue to the next iteration). (In the last iteration, the estimate is used unconditionally.)

We first note that, with probability at least 3/4, the algorithm does not produce output in any of the first $t \stackrel{\text{def}}{=} \lfloor \log(n/\overline{d}) \rfloor$ iterations (i.e., in any iteration i such that $\overline{d} < \ell_i$). The reason being that in order to halt in such an iteration $i \leq t$ it must hold that the estimate is at least $8\ell_i = (8\ell_i/\overline{d}) \cdot \overline{d} > 8\overline{d}$, and by Markov inequality this may happen with probability at most $\overline{d}/8\ell_i$. Thus, the probability of the bad event is upperbounded by $\sum_{i=1}^{t} \overline{d}/8\ell_i$, which in turn equals $\sum_{i=1}^{\lfloor \log(n/\overline{d}) \rfloor} (2^i\overline{d}/8n) < 1/4$. Finally, we observe that if this bad event does not occur then we are exactly in the situation analyzed above (i.e., the generalization of Theorem 4.1 to an arbitrary valid lower bound ℓ, which follows by Lemma 4.2). ■

Reflection. Underlying the proof of Theorem 4.3 is a general phenomenon that can be applied to any randomized approximation algorithm, and is beneficial provided that the algorithm performs better when given a valid lower bound on the quantity it needs to approximate. We are unaware of a prior use of this reasoning.

4.2 A Lower Bound

In this subsection we prove the essential optimality of the algorithm presented in the previous subsection.

Theorem 4.4 *For any n, $\overline{d} \in (2, o(n))$ and $\epsilon \in (\omega(1/\overline{d}n), o(n/\overline{d}))$, any algorithm that performs only distance queries and provides a $(1+\epsilon)$-approximation of the average distance of vertices in $G=(V,E)$ from $s \in V$, where $\overline{d}_G(s) = \overline{d}$, must ask $\Omega((n/(\epsilon\overline{d}))^{1/2})$ queries.*

Proof: For parameters n and $k \in (\omega(1), o(n))$, consider a (randomly labeled version of a) graph, denoted $G_{n,k}$, consisting of a star of $n-k$ vertices centered at s and a path of length k also starting at vertex s. (The reader may think of such a graph as a broom; see Figure 1.)

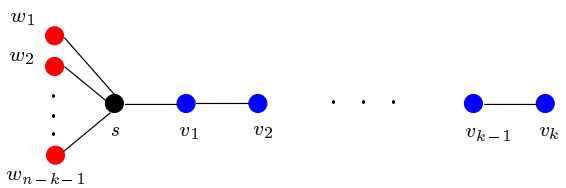

Figure 1: An illustration of the "broom-like" graph $G_{n,k}$.

By definition, the average distance of $G_{n,k}$ from s is

$$\overline{d}_{G_{n,k}}(s) \;=\; \frac{(n-k-1)\cdot 1 + \sum_{i=1}^{k} i}{n} \;=\; 1 + \frac{k^2-k-2}{2n} \;=\; 1 + (1-o(1)) \cdot \frac{k^2}{2n} \qquad (26)$$

Given $\overline{d} \in (2, o(n))$ and $\epsilon \in (1/\sqrt{\overline{d}n}, o(n/\overline{d}))$, we set k so that $1 + (k^2/2n) = \overline{d}$ (i.e., $k \approx (2(\overline{d}-1)n)^{1/2}$) and $k' \approx (2((1+\epsilon)\cdot\overline{d}-1)n)^{1/2}$. Thus, $\overline{d}_{G_{n,k}}(s) = (1-o(1))\cdot\overline{d}$ and $\overline{d}_{G_{n,k'}}(s) = (1+\epsilon)\cdot\overline{d}$. First, we observe that any $(1+\epsilon)$-approximation algorithm must make $\Omega(n/k') = \Omega((n/(1+\epsilon)\overline{d})^{1/2})$ queries in order to hit a vertex on the path (which is a necessary condition for distinguishing $G_{n,k}$ from $G_{n,k'}$). This establishes the claim for (say) $\epsilon > 1/10$. For the case of $\epsilon \leq 0.1$, we note that in order to distinguish $G_{n,k}$ from $G_{n,k'}$ the algorithm must hit one of the $k'-k$ vertices that are at distance greater than k from s in $G_{n,k'}$, which yields the lower bound of $\Omega(n/(k'-k)) = \Omega((n/\epsilon\overline{d})^{1/2})$. ■

4.3 Adding Access to Neighbor and Adjacency Queries

A natural question is whether providing access to neighbor and adjacency queries, in addition to distance queries, can improve the query complexity of the average degree estimation problem. We answer this question negatively.

Theorem 4.5 *Let $\overline{d} \in (2, o(n))$ and $\epsilon \in (\omega(1/\overline{d}n), o(n/\overline{d}))$, and consider algorithms that are allowed distance queries, neighbor queries, adjacency queries and degree queries. Any such algorithm that provides a $(1+\epsilon)$-approximation of the average distance of vertices in $G = (V, E)$ from $s \in V$ where $\overline{d}_G(s) = \overline{d}$, must perform $\Omega((n/\epsilon\overline{d})^{1/2})$ queries.*

For $k < k' < n$, consider the "broom-like" graphs $G_{n,k}$ and $G_{n,k'}$ defined in the proof of Theorem 4.5. Note that for an algorithm that is allowed neighbor and degree queries it is no longer true that the algorithm cannot distinguish (randomly labeled versions of) these graphs in $o(n/k')$ queries. In particular, a single degree query at vertex s distinguishes the two graphs (because the degree of s is $n-k-1$ and $n-k'-1$, respectively). Thus, a modification of the construction is due for proving Theorem 4.5. The proof follows.

Proof: We consider two cases depending on the value of ϵ. In case $\epsilon \geq 1/4$, we consider a

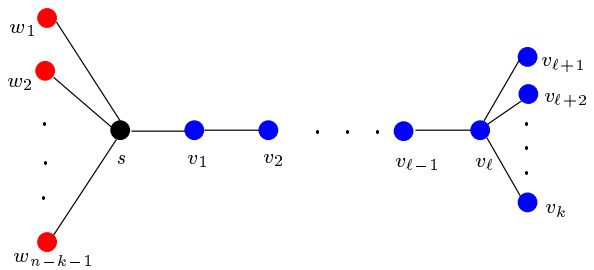

Figure 2: An illustration of the "two-edged broom" graph $G_{n,k,\ell}$.

"two-edged broom" graph $G_{n,k,\ell}$ with a "stick" of length ℓ connecting a star of size $n-k$ (centered at s) and a star of size $k-\ell$ (on the other side); see Figure 2. That is, as in $G_{n,k}$, the designated vertex s is the center of a star of $n-k$ vertices, with leaves denoted $w_1, \ldots, w_{n-k-1}$, and is the origin of a path of length ℓ, where the vertices on the path are $v_1, \ldots, v_\ell$. In addition, v_ℓ is also the center of a star, where the neighbors of v_ℓ are $v_{\ell+1}, \ldots v_k$ (as well as $v_{\ell-1}$). Observe that for any particular choice of n and $k = o(n)$, the degree of s in $G_{n,k}$ and in $G_{n,k,\ell}$ is the same. Moreover, the only difference between the two graphs is that in $G_{n,k,\ell}$ the path starting at s does not have length k but rather ends with a star after ℓ steps. It follows that (randomly labeled versions of) $G_{n,k}$ and

$G_{n,k,\ell}$ can only be distinguished by hitting a vertex that is *not* in the big ($(n-k)$-vertex) star. In "hitting" a vertex we mean that the vertex either corresponds to a queried vertex (in any type of query), or is the answer to a neighbor query. Thus, $\Omega(n/k)$ queries are required to distinguish the aforementioned graphs.

Recall that $\overline{d}_{G_{n,k}}(s) = 1 + (1 - o(1)) \cdot (k^2/2n)$. By definition of $G_{n,k,\ell}$,

$$\overline{d}_{G_{n,k,\ell}}(s) = \frac{n - k - 1 + \frac{\ell(\ell+1)}{2} + (k-\ell)(\ell+1)}{n} \tag{27}$$

$$= \frac{n + k\ell - (\ell^2/2) - \ell/2 - 2}{n} = 1 + (1 - o(1)) \cdot \frac{\ell(2k-\ell)}{2n} \tag{28}$$

Thus, given $\overline{d} \in (2, o(n))$ and $\epsilon > 1/\overline{d}^2$, we shall set k and ℓ such that $\overline{d}_{G_{n,k}}(s) = (1+\epsilon)\overline{d}$ while $\overline{d}_{G_{n,k,\ell}}(s) < \overline{d}$. In particular, we shall take $k \approx \sqrt{1+\epsilon} \cdot (2(\overline{d}-1)n)^{1/2}$, and $\ell \approx (\sqrt{1+\epsilon} - \sqrt{\epsilon}) \cdot (2(\overline{d}-1)n)^{1/2}$. Under this setting (and $\epsilon \geq 1/4$), we have $n/k = \Omega((n/\epsilon\overline{d})^{1/2})$, and the theorem follows for $\epsilon \geq 1/4$.

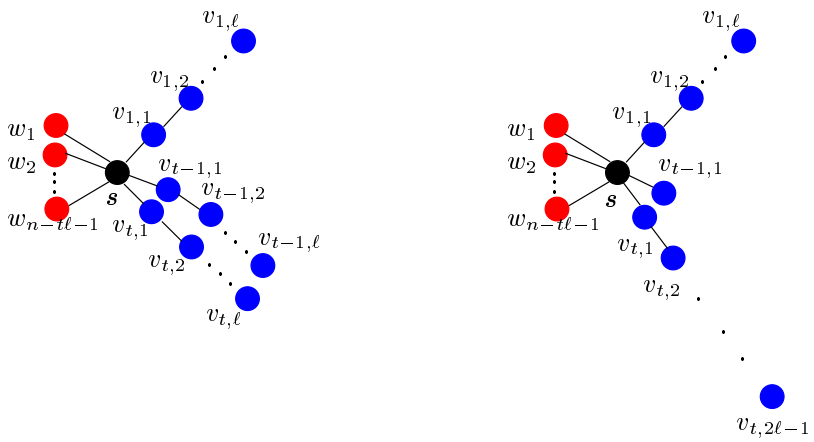

Figure 3: The "multi-stick broom" graph $G'_{n,k,\ell}$.

We now turn to the case of $\epsilon < 1/4$. In this case we use a different graph $G'_{n,t,\ell}$ that consists of a star of size $n - t\ell$ centered at s and t paths, each of length ℓ, emanating from s. That is, vertex s is the center of a star of $n - t\ell$ vertices, with leaves denoted $w_1, \ldots, w_{n-t\ell-1}$, and t paths such that the i^{th} path is denoted $v_{i,1}, \ldots, v_{i,\ell}$ and $v_{i,1}$ is connected to s. (See Figure 3.) We also consider the graph $G''_{n,t,\ell}$ that is identical to $G'_{n,t,\ell}$ except that the $t-1^{\text{st}}$ path has length 1 and the last path has length $2\ell - 1$. The average distance from s in $G'_{n,t,\ell}$ (resp., $G''_{n,t,\ell}$) is approximately $1 + (t \cdot \ell^2/2n)$ (resp., $1 + (t \cdot \ell^2/2n) + (\ell^2/n)$):

$$\overline{d}_{G'_{n,t,\ell}}(s) = \frac{n - t\ell - 1}{n} + \frac{t \cdot (1 + o(1)) \cdot \ell^2}{2n} \approx 1 + \frac{t \cdot \ell^2}{2n} \tag{29}$$

$$\overline{d}_{G''_{n,t,\ell}}(s) \geq \overline{d}_{G'_{n,k,\ell}}(s) + \frac{(1 + o(1)) \cdot (2\ell - 1)^2}{2n} - 2 \cdot \frac{(1 + o(1)) \cdot \ell^2}{2n} \approx \overline{d}_{G'_{n,k,\ell}}(s) + \frac{\ell^2}{n} \tag{30}$$

Given $\overline{d} \in (2, o(n))$ and $\epsilon \in (\omega(1/\overline{d}n), 1/4)$, we set the parameters such that $t\ell^2 \approx 2(\overline{d}-1)n$ and $\ell = \sqrt{\epsilon\overline{d}n}$ (i.e., $t \approx 2(\overline{d}-1)/(\epsilon\overline{d})$). Thus, the ratio between the averages distances to s in the two graphs is $1 + \epsilon$. In order to distinguish these graphs, the algorithm must hit a vertex on one

of the last two paths, which requires making $\Omega(n/\ell)$ queries, establishing the $\Omega((n/\epsilon\overline{d})^{1/2})$ lower bound. ■

4.4 Using Only Neighbor and Adjacency Queries

If we allow only neighbor and adjacency queries, then the problem becomes significantly harder.

Theorem 4.6 *Let $k > 1$ be a given approximation factor. Every algorithm that is allowed only neighbor, adjacency and degree queries must perform $\Omega(m/(k \log k))$ queries in order to obtain a k-approximation of $\overline{d}_G(s)$ in graphs G with m edges, provided $m \in (\Omega(n), O(n^2/k \log k))$.*

Proof: Here too we consider two distributions over graphs, denoted $\mathcal{D}_1$ and $\mathcal{D}_2$. In both distributions, the vertices are partitioned into $t = \Theta(k \log k)$ equal-size subsets $V_1, \ldots, V_t$. Each V_j is partitioned into two subsets, denoted L_j and R_j, where $|L_j| = \alpha \cdot n/t$ and $|R_j| = (1-\alpha) \cdot n/t$ for $\alpha \in (0, 0.5]$ that satisfies $|L_j| \cdot |R_j| = \alpha(1-\alpha) \cdot (n/t)^2 = m/t$. Such α exists provided that $(n/t)^2/4 \geq m/t$ (i.e., $m \leq n^2/4t$). The vertex s belongs to L_1. The edges of the graphs in the support of the two distributions are defined as follows: For each $j = 1, \ldots, t$, there is an almost complete bipartite graph between L_j and R_j; the only exception are two, randomly selected, missing edges. When $j = 1$, these edges are restricted not to be incident to s. The end-points of these missing edges are used to connect between vertices that belong to the different V_j's.

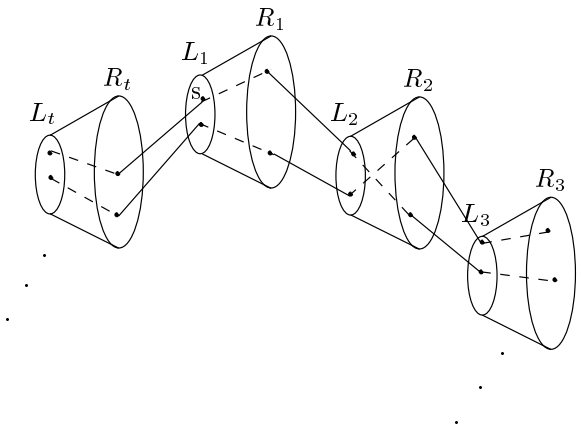

Figure 4: An illustration for graphs in the support of $\mathcal{D}_1$.

Specifically, let the endpoints of these missing edges be denoted $v_{j,1}, v_{j,2} \in L_j$ and $w_{j,1}, w_{j,2} \in R_j$. We refer to these vertices as terminal vertices, and to the edges we put between them as connecting edges. In $\mathcal{D}_1$ we add edges between the terminal vertices so as to form a cycle structure among the V_j's. Namely, for each $j = 1, \ldots, t-1$, we put the edges $(v_{j,1}, w_{j,1})$ and $(v_{j,2}, w_{j,2})$, and we put the edges $(v_{t,1}, w_{1,1})$, and $(v_{t,2}, w_{1,2})$. That is, the connecting edges are placed according to the structure of a 4-regular (non-simple) t-vertex graph that forms a "double cycle" (i.e., a cycle in which each edge appears with multiplicity 2). In $\mathcal{D}_1$ we connect the terminal vertices according to some 4-regular expander graph with vertex set $[t]$. That is, if (i, j) is an edge of the expander then we connect one of the (four) terminal vertices of V_i to one of the terminal vertices of V_j (while using each terminal vertex exactly once). Thus, each graph in the support of each distribution is

determined by the choice of the missing edges within each V_j (which in turn determine the terminal vertices of the graph as well as all connecting edges). For an illustration, see Figure 4.

By definition of $\mathcal{D}_1$, for every graph G in the support of $\mathcal{D}_1$, it holds that $\overline{d}_G(s) = \Theta(t)$. On the other hand, since in $\mathcal{D}_2$ the edges between the terminal vertices are determined by a 4-regular (t-vertex) expander graph, $\overline{d}_G(s) = \Theta(\log t)$ holds for every graph G in the support of $\mathcal{D}_2$. Thus, the ratio between the average distances is $\Omega(t/\log t) \geq k$, where equality holds for a suitable choice of $t = \Omega(k \log k)$.

Consider the execution of an algorithm that can perform only neighbor and adjacency queries. As long as the algorithm does not perform a query concerning a connecting edge, the distribution on the answers it gets is exactly the same for both distributions on graphs. Recall that there are $(m/t) - 2$ non-connecting edges that are incident to vertices in each V_j and two connecting edges. Thus, $\Omega(m/t)$ queries are required to distinguish between the two (graph) distributions. (Note that degree queries are of no help towards distinguishing the two distributions, whereas a single distance query (explicitly prohibited here) allows to distinguish these two distributions.) The lower bound follows by our choice of $t = O(k \log k)$. ■

5 Approximating All-Pairs Average Distance

In continuation to Section 4, we now turn to the question of estimating the average distance between all pairs of vertices. That is, for any given graph G over n vertices, let $\overline{d}_G = \frac{1}{n^2} \sum_{u,v \in V} \mathrm{dist}(u,v)$ denote the average distance between pairs of vertices in the graph.

5.1 An Algorithm

We show that the result for estimating the average distance to a single source, which was stated in Theorem 4.3, can be extended to estimating the average distance between all pairs of vertices.

Theorem 5.1 *There exists an algorithm that, on input a graph $G = (V,E)$ an a parameter $\epsilon \in (0,1)$, makes $O((n/\overline{d}_G)^{1/2}/\epsilon^2)$ distance queries and provides a $(1+\epsilon)$-approximation of $\overline{d}_G$.*

Proof: We focus on the basic version that is analogous to the one stated in Theorem 4.1, and comment that its modification is analogous to the proof of Theorem 4.3. Here, the basic algorithm takes a sample of $\Theta(\sqrt{n}/\epsilon^2)$ uniformly selected pairs of vertices, and outputs the average over the distances between these pairs. Below we modify the proof of Theorem 4.1 to show that the foregoing basic algorithm provides a $(1+\epsilon)$-approximation of $\overline{d}_G$.

Let $d_{\max}$ be redefined here to be the maximum distance between any pair of vertices, let p_i be the fraction of pairs of vertices (among all n^2) that are at distance i and let η be distributed according to the p_i's (i.e., $\Pr[\eta = i] = p_i$). Below we show that $\mathrm{Exp}[\eta^2] = O(\sqrt{n} \cdot \mathrm{Exp}[\eta]^2)$, and the correctness of the basic algorithm follows (as in the proof of Theorem 4.1). In fact, analogously to Lemma 4.2, for any lower bound ℓ of $\overline{d}_G$, we prove that

$$\mathrm{Exp}[\eta^2] = O(\sqrt{n/\ell} \cdot \mathrm{Exp}[\eta]^2)\,. \tag{31}$$

Needless to say, the proof is analogous to the proof of Lemma 4.2. The main challenge is to establish $\mathrm{Exp}[\eta] = \Omega(d_{\max}^2/n)$, and the rest follows as before.[4] Consider any pair of vertices v_0 and v_d that

[4] Eq. (22) hold as before by the definitions of $d_{\max}$ and η. Once we establish $\mathrm{Exp}[\eta] = \Omega(d_{\max}^2/n)$, which is analogous to Equation (23), we derive Equation (31) in the same manner that Lemma 4.2 was established.

are at distance $d \stackrel{\text{def}}{=} d_{\max}$ from each other, and let $v_1, \ldots, v_{d-1}$ be the vertices on the shortest path between them. The main observation here is that, for every $1 \leq i \leq d/3$, and every vertex w in the graph, it holds that $\text{dist}(v_i, w) + \text{dist}(w, v_{d-i}) \geq d/3$ (because $\text{dist}(v_i, v_{d-i}) = d - 2i \geq d/3$ whereas $\text{dist}(v_i, w) + \text{dist}(w, v_{d-i}) \geq \text{dist}(v_i, v_{d-i})$). Hence, here we have that

$$\text{Exp}[\eta] \;\geq\; \frac{\sum_{i=1}^{d/3} \sum_{w \in [n]} (\text{dist}(v_i, w) + \text{dist}(w, v_{d-i}))}{n^2} \;\geq\; \frac{(d/3) \cdot n \cdot (d/3)}{n^2} \;=\; \Omega\left(\frac{d_{\max}^2}{n}\right) \tag{32}$$

as claimed. ■

5.2 Lower Bounds

It is not hard to verify that lower bounds analogous to the ones stated in Theorems 4.5 and 4.6 hold also for approximating the average of all-pairs distances (i.e., $\overline{d}_G$). That is, for a graph $G = (V, E)$, any $(1+\epsilon)$-approximation algorithm that uses distance queries must make $\Omega((n/\epsilon\overline{d}_G)^{1/2})$ queries, whereas any constant factor approximation algorithm that uses only standard queries must make $\Omega(|E|)$ such queries. The bound is proved using the same graphs that were used in the proofs of Theorems 4.5 and 4.6. We merely need to verify that the average all-pair distances of these graphs essentially maintain the relative behavior of the single-source counterparts. This is most obvious for the graphs used in the proof of Theorem 4.6. As for the graphs used in the proof of Theorem 4.5, we note that

$$\overline{d}_{G_{n,k}} \;>\; \frac{n-k-1}{n} \cdot (1 + \overline{d}_{G_{n,k}}(s)) + \frac{k}{n} \cdot \frac{k}{2} \;\approx\; 2 + \frac{k^2}{n} \;=\; 2 \cdot \overline{d}_{G_{n,k}}(s) \tag{33}$$

$$\overline{d}_{G_{n,k,\ell}} \;\leq\; \frac{n-k}{n} \cdot (1 + \overline{d}_{G_{n,k,\ell}}(s)) + (1 + o(1)) \cdot \frac{(2k-\ell)\ell}{2n} \;\approx\; 2 \cdot \overline{d}_{G_{n,k,\ell}}(s) \tag{34}$$

$$\overline{d}_{G'_{n,t,\ell}} \;=\; \frac{n-t\ell}{n} \cdot (1 + \overline{d}_{G'_{n,k,\ell}}(s)) + \frac{t\ell}{n} \cdot \ell 2 + o(1) \;\approx\; 2 \cdot \overline{d}_{G'_{n,k,\ell}}(s) \tag{35}$$

$$\overline{d}_{G''_{n,t,\ell}} \;\geq\; \overline{d}_{G'_{n,k,\ell}} + 2 \cdot \frac{\ell^2}{n} \;\approx\; 2 \cdot \overline{d}_{G''_{n,k,\ell}}(s) \tag{36}$$

6 Extensions

The results of Sections 4 and 5 extend to the directed versions of these averaging problems: For the all-pairs problem, we require that the directed graph be strongly connected (so that all distances are defined). For the case of the single-source problem, it suffices to require that all vertices are reachable from the source.

References

[1] B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the Minimum Spanning Tree Weight in Sublinear Time. In *Proc. of the 28th ICALP*, pages 190–200, 2001.

[2] D. Coppersmith and S. Winograd. Matrix Multiplication via Arithmetic Progression. *Journal of Symbolic Computation*, Vol. 9, pages 251–280, 1990.

[3] D. Dor, S. Halperin, and U. Zwick. All Pairs Almost Shortest Paths. *SIAM Journal on Computing*, Vol. 29, pages 1740–1759, 2000.

[4] M. L. Elkin. Computing Almost Shortest Paths. Technical Report MCS01–03, Faculty of Mathematics and Computer Science, Weizmann Institute of Science, 2001.

[5] S. Even. *Graph Algorithms*. Computer Science Press, 1979.

[6] U. Feige. On sums of independent random variables with unbounded variance, and estimating the average degree in a graph. To appear in *Proc. of the 36th STOC*, 2004.

[7] Z. Galil and O. Margalit. All Pairs Shortest Paths for Graphs with Small Integer Length Edges. *Information and Computation*, Vol. 54, pages = 243–254, 1997.

[8] O. Goldreich and D. Ron. Property Testing in Bounded Degree Graphs. *Algorithmica*, Vol. 32 (2), pages 302–343, 2002.

[9] O. Goldreich and D. Ron. A Sublinear Bipartitness Tester for Bounded Degree Graphs. *Combinatorica*, Vol. 19 (3), pages 335–373, 1999.

[10] T. Kaufman, M. Krivelevich, and D. Ron. Tight Bounds for Testing Bipartiteness in General Graphs. In *Proc. of RANDOM'03*, pages 341–353, 2003.

[11] M. Parnas and D. Ron. Testing the diameter of graphs. *Random Structures and Algorithms*, Vol. 20 (2), pages 165–183, 2002.

[12] R. Siedel. On the All-Pairs-Shortest-Path Problem in Unweighted Undirected Graphs. *Journal of Computer and System Sciences*, Vol. 51, pages 400–403, 1995.

[13] U. Zwick. Exact and approximate distances in graphs - a survey. *Proceedings of the 9th Annual European Symposium on Algorithms (ESA)*, pages 33–48, 2001.