Approximating the Minimum Vertex Cover in Sublinear Time and a Connection to Distributed Algorithms

Michal Parnas
School of Computer Science
The Academic College of Tel-Aviv-Yaffo
Tel-Aviv, Israel
michalp@mta.ac.il

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

Abstract

For a given graph G over n vertices, let OPT_G denote the size of an optimal solution in G of a particular minimization problem (e.g., the size of a minimum vertex cover). A randomized algorithm will be called an *α-approximation algorithm with an additive error* for this minimization problem, if for any given additive error parameter $\epsilon > 0$ it computes a value $\widetilde{OPT}$ such that, with probability at least 2/3, it holds that $OPT_G \leq \widetilde{OPT} \leq \alpha \cdot OPT_G + \epsilon n$.

Assume that the maximum degree or average degree of G are bounded. In this case, we show a reduction from local distributed approximation algorithms for the vertex cover problem to sublinear approximation algorithms for this problem. This reduction can be modified easily and applied to other optimization problems that have local distributed approximation algorithms, such as the dominating set problem.

We also show that for the minimum vertex cover problem, the query complexity of such approximation algorithms must grow at least linearly with the average degree $\bar{d}$ of the graph. This lower bound holds for every multiplicative factor α and small constant ϵ as long as $\bar{d} = O(n/\alpha)$. In particular this means that for dense graphs it is not possible to design an algorithm whose complexity is $o(n)$.

Keywords. Sublinear approximation algorithms, distributed algorithms, minimum vertex cover.

*This research was supported by the Israel Science Foundation (grant number 89/05).

1 Introduction

As the need for computers to process massive data sets increases, the need for sublinear time (or space) algorithms becomes evident. In recent years efforts were made to design sublinear algorithms for a variety of basic computational problems and in two main frameworks: the sublinear-space algorithms associated with the streaming model (see [22] for a survey) and the sublinear-time algorithms associated with the framework of property testing [24, 11] (see [10, 8, 23] for surveys). We note that sublinear time algorithms were presented also outside the domain of property testing. Notable examples include the sublinear-time algorithms for estimating the size of a minimum spanning tree of a graph (e.g., [2, 4]) and the sublinear-time algorithms for clustering (e.g., [14, 21, 5]).

In this paper we further the study of sublinear-time algorithms for *combinatorial optimization problems that are NP-hard and for which polynomial-time approximation algorithms are known.* A typical example is the minimum vertex cover problem. The factor-two approximation algorithm of Gavril (*cf.* [9]) is one of the jewels of computer science. This algorithm runs in linear time and provides a relatively good approximation to a problem that is NP-hard to optimize or even to approximate up to a factor of 1.3606 [6]. We note also that Khot and Regev [16] showed, based on the unique games conjecture, that the size of a minimum vertex cover is hard to approximate to within $2-\gamma$, for any constant γ. Given that approximation seems unavoidable if we seek an efficient algorithm (i.e., one running in polynomial-time), can we obtain any non-trivial approximation in sublinear time?

Needless to say, the study of sublinear-time algorithms requires a specification of the way the algorithm obtains portions of the input. We follow the standard conventions by which such algorithms are modeled as oracle machines that may make *neighborhood queries*. Specifically, the query (v, i) is answered with the i-th neighbor of vertex v or with a special symbol in case v has less than i neighbors. We also allow *degree queries*, that is, the algorithm may query for every vertex v what is its degree[1].

1.1 Our Results

It is easy to see that a sublinear-time approximation for problems such as the minimum vertex cover problem, is not possible if we insist on the standard notion of a relative approximation (e.g., a factor two approximation). The reason being that such a relative approximation algorithm should obtain a good approximation even in the case that the optimum solution is tiny (e.g., the minimum vertex cover is a singleton). We thus relax the definition by allowing an additional additive error that is related to the optimum solution of a worst-case instance (e.g., the minimum vertex cover of a clique). This notion is formalized next.

Relative approximation with an additive error. For a given a graph G over n vertices, let OPT_G denote the size of an optimal solution in G of some minimization problem (e.g., the size of a minimum vertex cover). We say that a value $\widetilde{OPT}$ is an (α, ϵ)-*estimate* of OPT_G if

$$OPT_G \leq \widetilde{OPT} \leq \alpha \cdot OPT_G + \epsilon n.$$

[1]Clearly it is possible to replace each degree query by at most d neighbor queries, where d is the maximum degree of the graph. However, for the sake of simplicity we allow our algorithms also degree queries.

In the case of maximization problems we require that $OPT_G/\alpha - \epsilon n \leq \widetilde{OPT} \leq OPT_G$.

An algorithm that is given $\epsilon > 0$ as an input parameter and computes with probability at least 2/3 an (α, ϵ)-estimate of OPT_G for some value of α, is an *α-approximation algorithm with an additive error*. Note that ϵ is a parameter to the algorithm, whereas α is determined by the capabilities of the specific algorithm. For the sake of succinctness, whenever there is no cause for confusion, we use the term *α-approximation algorithm*, with the understanding that the algorithm is allowed an additive error.

We focus on the query complexity of such approximation algorithms. The running time of the algorithms we present is polynomial in the query complexity, and the lower bounds on the query complexity hold without any computational assumptions.

A reduction from local distributed algorithms. We show a reduction from *local* distributed approximation algorithms to sublinear approximation algorithms. The term *distributed* algorithms refers to the computation of processors that are connected via a communication network, and the term *local* refers to such algorithms that use a small number of communication rounds. Thus, during a computation of such a *local algorithm*, each processor can only obtain information from its "close" vicinity.

We consider local distributed algorithms that approximate a global function $f(G)$ of the graph, where $f(G)$ is the optimum, taken over all admissible subsets of vertices (e.g., vertex covers of G), of a (bounded) weighted sum of quantities associated with the individual vertices. We observe that a good approximation of $f(G)$ can be obtained in sublinear time by selecting a few vertices at random and emulating the distributed algorithm only for them (This requires a partial emulation also of their vicinities.). Below we demonstrate this idea for the case of the minimum vertex cover.

Let G be a distributed network with n nodes and degree at most d, and let $\mathcal{D}$ be a (possibly randomized) distributed algorithm that computes in k rounds a vertex cover C such that, with high (constant) probability, $|C| \leq \alpha \cdot VC_G$, where VC_G is the size of a minimum vertex cover in the graph, and $\alpha > 1$. Then, we obtain a randomized sublinear α-approximation algorithm that outputs an estimate $\widetilde{VC}$, such that with probability at least 2/3:

$$VC_G \leq \widetilde{VC} \leq \alpha \cdot VC_G + \epsilon n .$$

The query complexity of the sublinear algorithm is $O(d^k/\epsilon^2)$.

Applications of the reduction. To demonstrate the applicability of this reduction we present a very simple approximate distributed algorithm for the minimum vertex cover problem, and show how using our reduction we can get a sublinear $(2\log d + 1)$-approximation algorithm for the size of the minimum vertex cover. The query complexity of this sublinear algorithm is $O(d^{\log d}/\epsilon^2)$. Using our reduction and the distributed algorithm of Kuhn, Moscibroda, and Wattenhofer [17] for the minimum vertex cover we can get a c-approximation algorithm, where $c > 2$, using $d^{O(\log d)}/\epsilon^2$ queries. To obtain a 2-approximation algorithm, the number of queries performed is $d^{O(\log(d)/\epsilon^3)}$. By applying the reduction to a recent result of Marko [19] it is possible to get a 2-approximation algorithm by performing $d^{O(\log(d/\epsilon))}$ queries.

We can apply the reduction to other optimization problems as well. The first observation is that we can use the sublinear approximation algorithm for the minimum vertex cover to approximate the size of the maximum matching of a graph, since for any maximum matching M and any minimum vertex cover C it holds $|M| \leq |C| \leq 2|M|$. In addition, using the local distributed algorithm in [17], we can get an $O(\log d)$-approximation algorithm for the minimum size of a dominating set using $d^{O(\log d)}/\epsilon^2$ queries. Similar results can be obtained based on [17] for other covering and packing problems, such as finding the size of a minimum edge-cover of all isomorphic instances of a given subgraph. This covering problem was also studied by Marko and Ron [20] who built on the approach presented in this paper.

An improved reduction. The complexity of emulating k rounds of a distributed algorithm at a vertex in a graph of maximum degree d grows like d^k. We observe that, in many cases, we may suspend the emulation whenever we encounter a vertex of degree significantly greater than the average degree $\bar{d}$. In these cases (which include the minimum vertex cover problem and the minimum dominating set problem), the dependence on d can be replaced by $O(\min\{d, \bar{d}/\epsilon\})$, where the complexity is bounded in expectation. Furthermore, the algorithm does not need to be given d nor $\bar{d}$ as input, and the same complexity can be obtained if only neighbor queries (but not degree queries) are allowed.

A lower bound. Complexities that grow with the graph degree (or the average degree) seem essential to our approach. In contrast, in property testers for bounded-degree (and general sparse) graphs, the complexity many times decreases when the graph becomes dense (see, e.g., [12, 15] and [11, 1]). This raises the question of whether the complexity of approximation algorithms for the minimum vertex cover needs to grow with the degree. We show that the answer is affirmative. Specifically, we show that $\Omega(\bar{d})$ queries are necessary for any α-approximation algorithm (with an additive error $\epsilon < 1/4$) for VC_G, as long as $\bar{d} = O(n/\alpha)$. The lower bound holds even if we allow also vertex-pair queries, on top of the standard neighborhood queries. That is, for any pair of vertices u, v, the algorithm may query whether (u, v) is an edge in the graph.

1.2 Other Related Work

The minimum vertex cover problem was previously considered in the context of property testing (of bounded-degree graphs) [12]. In this context, the question is whether an algorithm can decide in sublinear time and with high probability whether a graph has a vertex cover of a certain size ρn or whether it is ϵ-*far* from having a vertex cover of this size. The latter means that more than $\epsilon \cdot dn$ edges must be removed from the graph so that it have a vertex cover of size ρn. It was observed in [12] that the NP-hardness results for approximating the minimum vertex cover, imply that this task is hard as well.

In the case of bounded-degree graphs, one can view our formulation of the sublinear approximation problem for the minimum vertex cover as an extension of the property testing task. Namely, the goal is to distinguish between the case that the graph has a vertex cover of size ρn and the case in which it is ϵ-far from having a vertex cover of size $\alpha \cdot \rho n$. However, we believe that our view of the problem as an approximation of the size of the vertex cover, where both a multiplicative and an additive error are allowed, is more natural.

Bogdanov, Obata and Trevisan [1] considered the question of obtaining lower bounds on the query complexity of sublinear approximation algorithms for the minimum vertex cover problem (that is, putting computational issues aside). They showed that every approximation algorithm for the minimum vertex cover that outputs an estimate with a multiplicative error of at most 7/6, has query complexity $\Omega(n)$. Furthermore, Trevisan (private communications) showed that for any constant γ, a $(2-\gamma)$-factor approximation cannot be obtained in $o(\sqrt{n})$ queries. For completeness we sketch Trevisan's lower bound in Section 6. Both lower bounds hold also when the algorithm is allowed an additive error of ϵn for a constant ϵ.

1.3 Open Problems

As stated above, we show a sublinear approximation algorithm for the minimum vertex cover problem when the maximum degree or the average degree of the graph are bounded. Our lower bound states that in order to get a constant approximation in dense graphs, the number of queries necessary is at least linear in the number of vertices n. However, it is still open whether there exists an approximation algorithm for the minimum vertex cover problem for dense graphs, whose complexity is sublinear in the number of edges of the graph.

2 Preliminaries

We consider undirected simple graphs G with n vertices and m edges. The degree of a vertex v is denoted by $d(v) = d_G(v)$. We denote the maximum degree of the graph by $d = d_G$, and the average degree by $\bar{d} = \bar{d}_G$ (that is, $\bar{d} = \frac{2m}{n}$). In all that follows, for the sake of simplicity, we omit floor and ceiling notation. Let OPT_G denote the size of an optimal solution in G of a minimization problem (e.g., the size of a minimum vertex cover). In this paper we are interested in randomized algorithms that compute an estimate, $\widetilde{OPT}$ of OPT_G.

Definition 1 *Let $\alpha \geq 1$ and $0 \leq \epsilon \leq 1$. We say that a value $\widetilde{OPT}$ is an* (α, ϵ)-estimate *of OPT_G if*

$$OPT_G \leq \widetilde{OPT} \leq \alpha \cdot OPT_G + \epsilon n.$$

An algorithm that is given ϵ as an input parameter and computes with probability at least 2/3 an (α, ϵ)-estimate of OPT_G for some value of α, is an α-approximation algorithm with an additive error*.*

As stated in the introduction a similar definition can be made for maximization problems.

Since we concentrate on the minimum vertex cover problem in this paper, we denote by VC_G the size of a minimum vertex cover of G, and denote by $\widetilde{VC}$ the estimate of VC_G output by the approximation algorithm.

We note that in general it is not possible to obtain a purely multiplicative approximation (i.e., $\epsilon = 0$) for the minimum vertex cover problem in sublinear-time. In particular it is not possible to distinguish in time $o(n)$ between a graph that is an independent set (for which the minimum vertex cover is of size 0) and a graph that contains a single, randomly selected, edge (for which the minimum vertex cover is of size 1). On the other hand, suppose that we have an α-approximation algorithm with an additive error for the minimum vertex cover problem. Then, as the following claim shows, it is possible to use it in order to find a purely multiplicative estimate, where the running time will depend on $\frac{n}{VC_G}$. That is, the smaller is the minimum vertex cover, the larger is the query complexity of the new algorithm.

Claim 1 *Let A be an α-approximation algorithm with an additive error for some minimization problem, and let $\nu(G) = \frac{OPT_G}{n} > 0$, where OPT_G is the optimal solution. Then we can obtain an estimate $\widehat{OPT}$ that with probability at least $2/3$ satisfies $OPT_G \le \widehat{OPT} \le 2\alpha \cdot OPT_G$, by performing $O\Big(Q_A(\nu(G)/4) \cdot \log^2(1/\nu(G))\Big)$ queries, where $Q_A(\epsilon)$ is the query complexity of A as a function of ϵ and is assumed to be monotonically non-decreasing with $1/\epsilon$. (The function Q_A may depend on other parameters, which are not stated explicitly (e.g., the maximum degree, d).)*

We note that the approximation factor of 2α in the claim can be reduced to $(1+\gamma)\alpha$ for any $\gamma < 1$ by introducing a dependence on $1/\gamma$, but for simplicity, we state it just for $\gamma = 1$.

Proof: As is described in detail below, the multiplicative estimate is achieved by running algorithm A with decreasing values of ϵ. In each iteration we have (with high probability) an interval to which OPT_G belongs. We stop when the interval is sufficiently small.

Assume that algorithm A is given, in addition to the additive approximation parameter ϵ, a confidence parameter, δ, and provides an (α, ϵ)-estimate with probability at least $1-\delta$ (instead of $2/3$). Clearly, this can be done at the cost of increasing the complexity of A by a multiplicative factor of of $\log(1/\delta)$. We shall run an iterative procedure, where in iteration i we execute algorithm A with $\epsilon = \epsilon_i = 2^{-i}$, and $\delta = \delta_i = (1/3) \cdot 2^{-i}$. Let $\widetilde{OPT}_i$ be the output of A in the i'th iteration. By the definition of algorithm A as an α-approximation algorithm with an additive error, with probability at least $1 - \frac{1}{3} \cdot 2^{-i}$, we have that $OPT_G \le \widetilde{OPT}_i \le \alpha OPT_G + 2^{-i}n$, or equivalently

$$\frac{1}{\alpha} \cdot (\widetilde{OPT}_i - 2^{-i}n) \;\le\; OPT_G \;\le\; \widetilde{OPT}_i \tag{1}$$

The procedure terminates once the ratio between the upper bound and the lower bound on OPT_G in Equation (1) is at most 2α, conditioned on the lower bound being positive. That is, when $\frac{\widetilde{OPT}_i}{\widetilde{OPT}_i - 2^{-i}n} \le 2$ and $\widetilde{OPT}_i - 2^{-i}n > 0$. It then outputs the lower bound, $\frac{1}{\alpha} \cdot (\widetilde{OPT}_i - 2^{-i}n)$.

Assume that in each iteration we have that $OPT_G \le \widetilde{OPT}_i \le \alpha OPT_G + 2^{-i}n$, where by our setting of the δ_i's, this holds with probability at least $2/3$. Then a ratio of at most 2α is necessarily obtained for $i \le \log(4/\nu(G))$. To verify this, let $\widetilde{OPT}_i = \beta \cdot OPT_G = \beta \cdot \nu(G) \cdot n$, where we have that $1 \le \beta \le \alpha + \frac{1}{2}$. Then $\frac{\widetilde{OPT}_i}{\widetilde{OPT}_i - 2^{-i}n} = \frac{\beta}{\beta - (1/2)}$. Since $\beta \ge 1$, this ratio is at most 2, as required. Also note that since $\widetilde{OPT}_i \ge \nu(G) \cdot n$, we also have that $\widetilde{OPT}_i - 2^{-i}n > 0$ for $i \le \log(4/\nu(G))$.

The upper bound on the total number of queries follows from our bound on the iteration i upon which the procedure terminates. ■

We also note that Vizing's theorem implies a trivial approximation for VC_G in terms of m and d (equivalently, n, $\bar{d}$ and d). Specifically, Vizing's theorem says that every graph can be edge-colored with $d+1$ colors. Therefore, every graph has a matching of size at least $\frac{m}{d+1}$, and hence $VC_G \ge \frac{m}{d+1} = n \cdot \frac{\bar{d}}{2(d+1)}$. This lower bound for VC_G depends on the ratio between $\bar{d}$ and d (or, more precisely, $\bar{d}$ and $d+1$): The more regular the graph is (i.e., as $\bar{d}$ is closer to d), the larger is the lower bound.

3 From Local Distributed Approximation Algorithms to Sublinear Approximation Algorithms

In this section we show a general reduction from local distributed approximation algorithms to sublinear approximation algorithms. The distributed algorithms should obey certain conditions, which we specify in Subsection 3.1.

A distributed algorithm that runs on a (synchronous) network G consists of some number k of communication rounds. In each round every node in G can send messages to its neighbors. After k rounds, each node completes its computation. In particular, if the goal of the algorithm is to select a vertex cover for the graph, then after k rounds each node decides whether it belongs to the cover or not.

As long as the number of rounds k is smaller than the diameter of the network, such algorithms are by nature *local*, as any node can gather information about nodes that are at most distance k away. Still the goal of many such algorithms can be global: that is, to compute some global function of the network. There is usually a tradeoff between the locality of the computation and the quality of the global approximation achieved.

We first state the reduction for the specific minimum vertex cover problem, and then explain what should hold so that the reduction is true also for other problems and settings. We also assume first that the maximum degree d of the network is known in advance. In Section 3.2 we show how to remove this assumption and get an algorithm whose query complexity depends on the average degree $\bar{d}$, or more precisely on $O(\bar{d}/\epsilon)$, rather than on d, in expectation.

Theorem 1 *Let G be a distributed network with n nodes and degree at most d. Let $\mathcal{D}$ be a deterministic distributed algorithm that computes in k rounds a vertex cover C. Then it is possible to design a randomized sublinear approximation algorithm that outputs an estimate $\widetilde{VC}$, such that with probability at least $2/3$:*

$$|C| \le \widetilde{VC} \le |C| + \epsilon n.$$

The query complexity of the sublinear algorithm is $O(d^k/\epsilon^2)$, where the algorithm uses only neighbor and degree queries.

Proof: The sublinear algorithm is obtained from algorithm $\mathcal{D}$ by applying the following reduction algorithm:

Algorithm 1 (Reduction algorithm)

1. *Uniformly and independently select $s = 8/\epsilon^2$ vertices from G. Denote the subset (multiset) of selected vertices by S.*
2. *For each $v \in S$, consider the subgraph $G_k(v)$ induced by the k-neighborhood of v.*
3. *For each $v \in S$, run Algorithm $\mathcal{D}$ on $G_k(v)$, where the degree of vertices in $G_k(v)$ that are at distance exactly k from v is as it is in G.*
   Set $X_v = 1$ if $\mathcal{D}$ decided to add v to the vertex cover C, and otherwise $X_v = 0$.
4. *Output $\widetilde{VC} = \frac{n}{s} \cdot \sum_{v \in S} X_v + \frac{\epsilon}{2} n$.*

The simple but important observation is that for any vertex v, if we run Algorithm $\mathcal{D}$ on the subgraph $G_k(v)$, then it makes the same decision about vertex v as it would if we would run $\mathcal{D}$ for k rounds on the whole graph G. But this is clear since in k rounds no information that originated in a vertex whose distance from v is greater than k can reach v. In other words, the decision of vertex v can only depend on messages sent by vertices at distance at most k from v.

Note that the degree of vertices in $G_k(v)$ that are at distance exactly k from v is viewed by Algorithm 1 as it is in G. This is done since in the distributed algorithm vertices at distance k from v can check their degrees without receiving any messages from vertices farther away from v. Thus what v sees in k rounds in G is identical to what it sees by running $\mathcal{D}$ only on its k-neighborhood $G_k(v)$ (although other vertices in $G_k(v)$ may have a different view than what they have in G).

Therefore the variables X_v defined in Step 3 of the algorithm are assigned the correct values. That is, $X_v = 1$ if and only if $v \in C$, where C is the vertex cover computed by the distributed algorithm. Now using an additive Chernoff bound, if $s = 8/\epsilon^2$, then with probability at least 2/3 the deviation of $\frac{1}{s} \cdot \sum_{v \in S} X_v$ from the expected value $\frac{1}{n} \cdot |C|$, is at most $\frac{\epsilon}{2}$, implying that $|C| \leq \widetilde{VC} \leq |C| + \epsilon n$.

As to the query complexity of Algorithm 1: building the subgraph $G_k(v)$ can be done by constructing a BFS tree of depth k rooted at v, using $O(d^k)$ neighbor queries. Then we can run the algorithm $\mathcal{D}$ on the subgraph induced by the BFS tree, where the degree of the leaves of the tree is as in G. This requires $O(d^k)$ degree queries. ■

As to the running time of the sublinear algorithm obtained by applying the reduction in Theorem 1, it depends of course on the number of operations performed by the distributed algorithm in each of the vertices of the graph. For the algorithms we present below, it will also be sublinear.

3.1 Extending the Reduction to Other Settings

The above reduction can be generalized to other problems and settings as follows:

1. **Randomized distributed algorithms:** If the distributed algorithm is randomized then it is still possible to get a similar reduction with the following simple changes.

   First the sublinear algorithm should take a slightly larger sample so that its error probability is smaller. Then using a union bound and adding the error probability of the distributed algorithm, we will get a total error of at most 2/3 as before.

   Second, if the k-neighborhoods of two vertices u, v sampled by the reduction algorithm intersect, then when we run the distributed algorithm on $G_k(u)$ and $G_k(v)$, we must make sure that it uses the same coin tosses for vertices in $G_k(u) \cap G_k(v)$.

2. **Reduction to other problems:** A similar reduction can be designed for any local distributed algorithm that approximates some global function $f(G)$ of the graph, where $f(G)$ is the optimum, taken over all admissible subsets of vertices (e.g., vertex covers of G), of a (bounded) weighted sum of quantities associated with the individual vertices.

   For example, it is possible to get a sublinear approximation algorithm for the dominating set problem, which obtains an $(O(\log d), \epsilon)$-estimate of the minimum size (weight) of a dominating set, using $d^{O(\log d)}/\epsilon^2$ queries. This is done by applying our reduction to the distributed algorithm of [17] for the dominating set problem.

3.2 Dependence on the average degree

It is possible to replace the dependence that the sublinear algorithm has on the maximum degree d, with a dependence on the average degree $\bar{d}$, or more precisely on $O(\bar{d}/\epsilon)$, in expectation. This is useful when the maximum degree is high, whereas the average degree is bounded. In particular, it is possible that $d = \Omega(n)$ and $\bar{d} = O(1)$.

The high level idea. The idea is simple: Suppose that we know the average degree $\bar{d}$ of the graph, or can approximate it in sublinear time. Then we can modify any approximation algorithm for the minimum vertex cover problem as follows: First we add to the vertex cover all vertices of degree greater than $2\bar{d}/\epsilon$, and remove them and all edges incident with them from the graph. Since there are at most $\epsilon \cdot n/2$ vertices of degree greater than $2\bar{d}/\epsilon$ in the original graph, then the number of vertices added by this initial stage is at most $\epsilon \cdot n/2$. We can now run the approximation algorithm on the resulting graph whose degree is at most $2\bar{d}/\epsilon$, setting the additive approximation parameter to $\epsilon/2$.

There are sublinear-time procedures for approximating the average degree in a graph [7, 13]. Their drawback is that their running time is $\Theta(\sqrt{n/\bar{d}})$, while we are interested in a running time that depends only on $\bar{d}$. Furthermore, this bound is tight for any constant factor approximation [13]. However, the source of the difficulty of obtaining such an estimate is that the approximation algorithm is required not to *under-estimate* the average degree by more than a constant multiplicative factor. For our application, we only need to take care not to *over-estimate* the average degree by too much. In addition, there shouldn't be too many vertices with a degree that is higher than the estimate we obtain. In particular, we show how to find, with a high probability, a value $\hat{d}$, such that:

1. The value $\hat{d}$ is upper bounded by $O(\bar{d}/\epsilon)$;
2. The graph has at most $\frac{\epsilon}{2}n$ vertices with degree greater than $\hat{d}$.

(We shall actually impose a somewhat stronger requirement than that stated in the first item for reasons that will be explained below.)

Let $\mathcal{D}$ be a distributed algorithm that constructs a vertex cover whose size is at most $\alpha \cdot VC_G$ in $k = k(d)$ rounds[2]. We assume that $k = k(d)$ does not grow too fast as a function of d. Specifically, we make the (relatively weak) assumption that $k(2d) = O(2^{k(d)})$. Note that this holds even if $k(d)$ is itself a tower function in d. As described above, we can modify Algorithm $\mathcal{D}$ so that, given $\hat{d}$, it first adds to the vertex cover all vertices of degree greater than $\hat{d}$ and removes all edges incident with these vertices from the graph. Thus (with a high enough probability) the number of vertices added to the vertex cover by this initial step is at most $\frac{\epsilon}{2}n$, and the size of the minimum vertex cover of the resulting graph is at most VC_G.

Now Algorithm $\mathcal{D}$ continues its execution on the resulting graph with the bound d set to $\hat{d}$. The number of iterations of the modified Algorithm $\mathcal{D}$ is hence $k(\hat{d})$ and the size of the final vertex cover $C_{\hat{d}}$ is:

$$VC_G \le |C_{\hat{d}}| \le \alpha \cdot VC_G + \frac{\epsilon}{2}n \qquad (2)$$

[2]The number of rounds k may be a function of other parameters of the graph, but here we are interested in the dependence on the degree, and hence the dependence on other parameters is kept implicit.

In reality we do not actually modify Algorithm $\mathcal{D}$, but we use the above idea to modify the reduction (Algorithm 1) which is applied to Algorithm $\mathcal{D}$, and thus get a sublinear approximation algorithm that has a dependence on $O(\bar{d}/\epsilon)$. To be precise, since there is a small, but non-zero probability that $\hat{d}$ is greater than $c \cdot \bar{d}/\epsilon$ (for some constant c), such a bound holds in expectation.

The procedure. The following procedure will be used by the modified sublinear algorithm to find the value $\hat{d}$. The procedure actually ensures something somewhat stronger than $\hat{d}$ being upper bounded by $O(\bar{d}/\epsilon)$ with high probability. Rather, for every sufficiently large i it gives a bound, which decreases exponentially with i and with $k(2^i)$, on the probability that $\hat{d} = 2^i$. This is required so as to upper bound the expected running time of the reduction algorithm that is given $\hat{d}$ as input.

```
Procedure Average-Degree-Bound
  1. $i = 1$, Found = FALSE
  2. While Found is FALSE:
     (a) Set $\delta_i = 2^{-(i+1)\cdot(k(2^{i+1})+1)}$, sample $q_i = \Theta(\log(1/\delta_i)/\epsilon)$ vertices, uniformly
         at random, and for each sampled vertex query its degree.
     (b) If there are at most $(\epsilon/4)q_i$ vertices in the sample with degree greater
         than $2^i$, then Found = TRUE, else $i = i + 1$.
  3. Let $\hat{d} = 2^i$ and output $\hat{d}$.
```

Claim 2 *Let $\hat{d}$ be the output of Procedure Average-Degree-Bound, and for any integer $0 \le j \le n$, let $V_{>j} = \{v : \deg(v) > j\}$ be the set of vertices of degree greater than j. Then:*

1. *With probability at least $5/6$, $|V_{>\hat{d}}| \le \frac{\epsilon}{2}n$.*
2. *With probability one, $\hat{d} \le 2d$, and for each $i \ge \log(16\bar{d}/\epsilon)$, the probability that $\hat{d} = 2^i$ is at most δ_{i-1}, where δ_i is as defined in the procedure.*
3. *If for every x, $k(2x) \le 2^{k(x)-1} - 1$, then the expected running time of Procedure Average-Degree-Bound is $O(k(\tilde{d}) \cdot \log^2(\tilde{d})/\epsilon)$, where $\tilde{d} = \min\{2d, 16\bar{d}/\epsilon\}$.*

Proof: Let t be the minimum integer such that $|V_{>2^t}| \le \frac{\epsilon}{2}n$. Hence, for each $i < t$, we have that $|V_{>2^i}| > \frac{\epsilon}{2}n$. By a multiplicative Chernoff bound, for each $i < t$, the probability that the i'th sample contains at most $(\epsilon/4)q_i$ vertices with degree greater than 2^i (that is, less than half the expected number) is $\exp(-\Omega(\epsilon \cdot q_i)) \le \delta_i$. Recall that $\delta_i = 2^{-(i+1)\cdot(k(2^i)+1)}$, which is at most $2^{-2(i+1)} \le 2^{-i-3}$ (since $k(x) \ge 1$ for every $x \ge 1$). Therefore, the probability that $\hat{d} < 2^t$ is upper bounded by

$$\sum_{i=1}^{t-1} \delta_i < 2^{-3} \cdot \sum_{i=1}^{\infty} 2^{-i} < 1/6 .$$

But if $\hat{d} \ge 2^t$ then $|V_{>\hat{d}}| \le \frac{\epsilon}{2}n$, and we have established the first item of the claim.

We now turn to the second item. First observe that once $2^i \geq d$ then there are no vertices with degree greater than 2^i, and so $\hat{d} \leq 2d$ as claimed. Now, let E_i be the event that the sample in the $i'th$ iteration of the procedure contains more than $(\epsilon/4)q_i$ vertices with degree greater than 2^i. Note that by the definition of the average degree, $\bar{d}$, we have that $|V_{>8\bar{d}/\epsilon}| < \frac{\epsilon}{8}n$. Therefore, by a multiplicative Chernoff bound, once $2^i \geq 8\bar{d}/\epsilon$, the probability of E_i occurring is bounded by $\exp(-\Omega(\epsilon \cdot q_i)) \leq \delta_i$. Hence, for every $i \geq \log(16\bar{d}/\epsilon)$,

$$\Pr[\hat{d} = 2^i] = \left(\prod_{j=1}^{i-1} \Pr[E_j]\right)(1 - \Pr[E_i]) \leq \prod_{j=\log(8\bar{d}/\epsilon)}^{i-1} \delta_j \leq \delta_{i-1}$$

and we have established the second item of the claim.

Recall that by the premise of this item, $k(2^{i+1}) \leq 2^{k(2^i)-1} - 1$. Therefore, by the definition of δ_i we have:

$$\log(1/\delta_i) \;=\; (i+1)\cdot(k(2^{i+1})+1) \;\leq\; 2^{k(2^i)-1+\log(i+1)} \;\leq\; 2^{k(2^i)+(i-1)} \;\leq\; 2^{i\cdot k(2^i)} \;=\; 2^{-i}/\delta_{i-1} \quad (3)$$

Using the second item of the claim and the definition of $\tilde{d}$, the expected running time of Procedure Average-Degree-Bound is upper bounded by

$$\begin{aligned}
\sum_{i=1}^{\log\tilde{d}-1} q_i + \sum_{i=\log\tilde{d}}^{\log d+1} q_i \cdot \delta_{i-1} &\leq \log\tilde{d}\cdot q_{\log\tilde{d}-1} + O(1/\epsilon)\cdot \sum_{i=\log\tilde{d}}^{\log d+1} \log(1/\delta_i)\cdot\delta_{i-1} && (4)\\
&\leq \log\tilde{d}\cdot q_{\log\tilde{d}-1} + O(1/\epsilon)\cdot \sum_{i=\log\tilde{d}}^{\log d+1} 2^{-i} && (5)\\
&= O(\log\tilde{d}\cdot q_{\log\tilde{d}-1}) && (6)\\
&= O(k(\tilde{d})\cdot\log^2(\tilde{d})/\epsilon) && (7)
\end{aligned}$$

where Equation (4) follows from the definition of q_i, Equation (5) follows from Equation (3), and Equation (7) follows again from the definition of q_i. We have thus established the third item of the claim. ■

We now present the modified sublinear algorithm:

```
Algorithm 2 (Reduction algorithm with dependence on $\hat{d}$)

1. Call Procedure Average-Degree-Bound and let $\hat{d}$ be the value it returns.
2. Uniformly and independently select $s = 32/\epsilon^2$ vertices from $G$, and set $k = k(\hat{d})$.
   (Recall that $k(\cdot)$ is the number of rounds performed by Algorithm $\mathcal{D}$ as a function of
   the degree-bound.)
3. For each sampled vertex $v$:
   (a) Construct the subgraph $G_k(v)$, by constructing the BFS tree rooted at $v$, with the
       following change: if we reach a vertex that has degree greater than $\hat{d}$, then we do
       not continue the BFS from that vertex.
   (b) If the degree of $v$ is greater than $\hat{d}$ then set $X_v = 1$. Otherwise, remove from
       $G_k(v)$ all vertices with degree greater than $\hat{d}$ and their incident edges. Let $G'_k(v)$
       be the resulting graph.
   (c) If $X_v$ was not yet set to 1 then run Algorithm $\mathcal{D}$ on $G'_k(v)$, where $\hat{d}$ is passed to
       it as a bound on the maximum degree in $G'_k(v)$. Set $X_v = 1$ if $v$ is selected to be
       added to the vertex cover, and otherwise $X_v = 0$.
4. Output $\widetilde{\mathrm{VC}} = \frac{n}{s} \cdot \sum_{v \in S} X_v + \frac{\epsilon}{4} n$.
```

Claim 3 *If Algorithm $\mathcal{D}$ is a deterministic distributed algorithm that constructs a vertex cover of size at most $\alpha \cdot \mathrm{VC}_G$ in $k(d)$ rounds, then Algorithm 2 is an α-approximation algorithm with an additive error. If $k(2x) \leq 2^{k(x)-1} - 1$ for every x, then the expected query complexity of Algorithm 2 is $O\left(\tilde{d}^{k(\tilde{d})}/\epsilon^2\right)$ where $\tilde{d} = \min\{2d, 16\bar{d}/\epsilon\}$.*

Proof: Since $s = 32/\epsilon^2$, by an additive Chernoff bound, with probability at least 5/6,

$$\left| \frac{1}{s} \cdot \sum_{v \in S} X_v - \frac{1}{n} \cdot |C_{\hat{d}}| \right| \leq \frac{\epsilon}{4}$$

where $C_{\hat{d}}$ is as defined in the text preceding Equation (2). By adding up the probability that $\hat{d}$ does not have the first property stated in Claim 2 with the probability that $\widetilde{VC}$ deviates by more than $\frac{\epsilon}{2} n$ from $|C_{\hat{d}}|$ we get that with probability at least 2/3:

$$VC_G \leq \widetilde{VC} \leq \alpha \cdot VC_G + \epsilon n \, . \tag{8}$$

By the second item of Claim 2, the expected number of queries performed by Procedure Average-Degree-Bound is $O\left(k(\tilde{d}) \cdot \log^2(\tilde{d})/\epsilon\right)$. For any setting of $\hat{d}$, the number of queries performed in step 3 of the algorithm is $O(\hat{d}^{k(\hat{d})}/\epsilon^2)$. By the third item of Claim 2, the total expected number of queries performed by Algorithm 2 is:

$$
\begin{aligned}
& O\left(k(\tilde{d}) \cdot \log^2(\tilde{d})/\epsilon\right) + O\left(\tilde{d}^{k(\tilde{d})}/\epsilon^2\right) + \sum_{i=\log \tilde{d}+1}^{\log d+1} O\left(2^{i \cdot k(2^i)}/\epsilon^2\right) \cdot \delta_{i-1} \\
= \;& O\left(\tilde{d}^{k(\tilde{d})}/\epsilon^2\right) + O(1/\epsilon^2) \cdot \sum_{i=\log \tilde{d}+1}^{\log d+1} 2^{i \cdot k(2^i)} \cdot \delta_{i-1} && (9) \\
= \;& O\left(\tilde{d}^{k(\tilde{d})}/\epsilon^2\right) + O(1/\epsilon^2) \cdot \sum_{i=\log \tilde{d}+1}^{\log d+1} 2^{i \cdot k(2^i)} \cdot 2^{-i\cdot(k(2^i)+1)} && (10) \\
= \;& O\left(\tilde{d}^{k(\tilde{d})}/\epsilon^2\right) + O(1/\epsilon^2) \cdot \sum_{i=\log \tilde{d}+1}^{\log d+1} 2^{-i} && (11) \\
= \;& O\left(\tilde{d}^{k(\tilde{d})}/\epsilon^2\right) && (12)
\end{aligned}
$$

where $\tilde{d} = \min\{2d, 16\bar{d}/\epsilon\}$. ■

4 Applying the Reduction to the Minimum Vertex Cover Problem

In order to demonstrate how our reduction can be applied to a specific algorithm, we first present a simple distributed approximation algorithm for the size of a minimum vertex cover. Although this algorithm is not the best distributed algorithm known for this problem, it is easy to describe and has a short and self contained proof of correctness. Thus applying the reduction to it will result in a simple sublinear algorithm for the vertex cover problem. We later show how the reduction can be applied to better approximation algorithms. We assume for now that the maximum degree d of the graph is known.

Algorithm 3 (Distributed Approximation algorithm for vertex cover)

```
1. Let $d$ be an upper bound on the degree of vertices in the graph (given as input to the algorithm).
2. Let $C = \emptyset$ be the initial vertex cover.
3. For $i = 1$ to $\log d$ do:
    (a) Each vertex whose degree is at least $d/2^i$ adds itself to $C$.
    (b) Remove from the graph all edges that are incident with the vertices added to $C$.
4. Output $|C|$.
```

Theorem 2 *Algorithm 3 constructs a vertex cover C such that $VC_G \leq |C| \leq (2\log d + 1) \cdot VC_G$.*

Proof: First it is clear that $|C| \geq VC_G$ since the algorithm removes edges from the graph only if at least one of their endpoints is added to C, and therefore C is a vertex cover.

As to the upper bound on the size of C: let O be a minimum vertex cover of G of size VC_G, and let $\overline{O}$ be all the vertices not in O. We will prove shortly that in each iteration at most $2 \cdot VC_G$ new vertices are added from $\overline{O}$ to the cover C. Since the number of iterations is at most $\log d$ (because after $\log d$ iterations all remaining vertices of the graph have degree 0), then the total number of vertices added to C from $\overline{O}$ is at most $2 \cdot VC_G \cdot \log d$. The claim follows by adding to this the VC_G vertices in O that may also be added by the algorithm to C.

To complete the proof we prove that on each iteration at most $2 \cdot VC_G$ new vertices are added from $\overline{O}$ to the cover C. Indeed suppose we are at the beginning of the i'th iteration. At this point the degree of all vertices is at most $d/2^{i-1}$. Thus the number of edges remaining between O and $\overline{O}$ is at most $VC_G \cdot d/2^{i-1}$. Denote by X_i the number of vertices in $\overline{O}$ of degree at least $d/2^i$ at the beginning of the $i'th$ iteration (recall that there are no edges inside $\overline{O}$ since O is a vertex cover). Therefore, $X_i \cdot d/2^i \leq VC_G \cdot d/2^{i-1}$, and so $X_i \leq 2VC_G$ as claimed. ■

We now apply the reduction described in Algorithm 1 to Algorithm 3 in order to obtain a sublinear randomized algorithm that outputs with high probability an $(2\log d + 1, \epsilon)$-estimate of VC_G. The query complexity of the algorithm is $O\left(d^{\log d}/\epsilon^2\right)$, since the number of rounds of the distributed algorithm is $k = \log d$. If we apply the reduction of Algorithm 2, then the resulting expected query complexity is $O\left(\tilde{d}^{\log \tilde{d}}/\epsilon^2\right)$, where $\tilde{d} = \min\{2d, 16\bar{d}/\epsilon\}$.

4.1 Better approximation algorithms for the minimum vertex cover

Kuhn, Moscibroda, and Wattenhofer [17] prove the following theorem as part of a general result regarding distributed computation of covering and packing problems:

Theorem 3 ([17]) *For any integer k such that $k = O(\log d)$, it is possible to find a vertex cover, whose size is up to a factor of $2 \cdot d^{5/k}$ larger than the size of the minimum vertex cover, in $O(k)$ rounds. In particular, for $\gamma \in (0, 1)$ it is possible to find a vertex cover whose size is at most $(2+\gamma)$ times the size of a minimum vertex cover in $O((\log d)/\gamma^3)$ rounds.*

By combining Theorem 1 with Theorem 3 we can get an $(O(1), \epsilon)$-estimate of VC_G using $d^{O(\log d)}/\epsilon^2$ queries. To get a $(2, \epsilon)$-estimate, we shall perform $d^{O(d/\epsilon^3)}$ queries. Recently Marko [19] showed that it is possible to improve the dependence on γ in the distributed setting from polynomial to logarithmic. Namely, she described a distributed algorithm that finds a vertex cover whose size is at most $(2+\gamma)$ larger than the minimum vertex cover in $O(\log(d/\gamma))$ rounds. Thus it is possible to get a $(2, \epsilon)$-estimate by performing $d^{O(\log(d/\epsilon))}$ queries. In both cases we can replace the dependence on d by a dependence on $\Theta(\bar{d}/\epsilon)$. We note that Marko's algorithm is similar to the $O(\log n)$-rounds distributed algorithm for the maximal independent set of Luby [18].

5 A Lower Bound that is Linear in the Average Degree

In this section we prove the following lower bound on the number of queries needed to approximate the minimum size of a vertex cover. Note that for dense graphs, that is, graphs for which the average degree is $\Theta(n)$, our result shows that any constant factor approximation requires $\Omega(n)$ queries. Also recall that by Vizing's theorem, $VC_G \geq \frac{m}{d+1} = n \cdot \frac{\bar{d}}{2(d+1)} \geq \frac{\bar{d}}{2}$. Since the size of a minimum vertex cover is at most $n-1$, any lower bound on the query complexity of α-approximation algorithms can hold only when the average degree is $O(n/\alpha)$, as is the case in Theorem 4.

Theorem 4 *For any n, $\alpha > 1$, $b \le \frac{n-1}{4\alpha}$ and $\epsilon < 1/4$, every α-approximation algorithm of the minimum vertex cover of graphs with average degree $\Theta(b)$ requires $\Omega(b)$ queries. This lower bound holds when all types of queries are allowed (i.e., neighbor queries, degree queries and vertex-pair queries).*

Proof: We first describe the idea behind the proof and then proceed with the details. We will define two families of n-vertex graphs denoted $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively, such that the average degree of the graphs in each one of the families is $\Theta(b)$. All graphs in $\mathcal{G}_1$ will have a vertex cover of size $b \le \frac{n-1}{4\alpha}$, while the size of the minimum vertex cover of graphs in $\mathcal{G}_2$ is $\frac{n-1}{2}$.

Thus, by definition, for the graphs in $\mathcal{G}_1$, any α-approximation algorithm should output (w.h.p.) an estimate $\widetilde{VC} \le \alpha \cdot \left(\frac{n-1}{4\alpha}\right) + \epsilon n$, while for graphs in $\mathcal{G}_2$ it should hold that $\widetilde{VC} \ge \frac{n-1}{2}$. Since $\epsilon < 1/4$ we have that $\alpha \cdot \left(\frac{n-1}{4\alpha}\right) + \epsilon n < \frac{n-1}{2}$. Hence in order to prove the theorem it suffices to show that no algorithm that performs $o(b)$ queries can distinguish between a graph uniformly selected from $\mathcal{G}_1$ and a graph uniformly selected from $\mathcal{G}_2$.

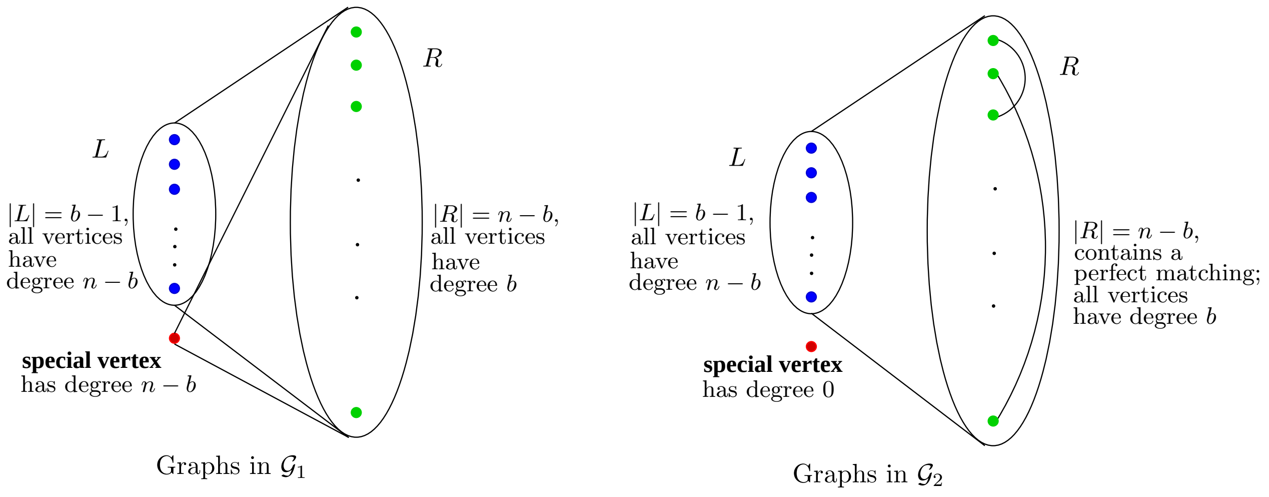
Description of the Families: In both families, all graphs contain (an isomorphic copy of) the following graph, G_0, as a subgraph. The graph G_0 is a complete bipartite graph over $n-1$ vertices whose left-hand-side, denoted L, contains $b-1$ vertices, and whose right-hand-side, denoted R, contains $n-b$ vertices. We assume for simplicity that $n-1$ and $n-b-1$ are even. The construction can easily be adapted to deal with the case that one of them is odd.

In the graphs that belong to $\mathcal{G}_1$, the single remaining vertex, which we refer to as the *special* vertex, is incident to all vertices in R. The graphs in $\mathcal{G}_1$ differ in the identity of the special vertex. In addition, the graphs in $\mathcal{G}_1$ differ in the labelings of the edges between vertices in R and the special vertex, where we allow all possible labelings.

In the graphs that belong to $\mathcal{G}_2$, the single remaining special vertex has no incident edges. Instead, there is a perfect matching between the vertices in R, where for each possible perfect matching there is a graph in $\mathcal{G}_2$. See Figure 1 for an illustration of the two families of graphs.

Average Degree and Vertex Cover size: By construction, in both families of graphs, all graphs have average degree $\Theta(b)$: In $\mathcal{G}_1$ there are $n-b$ vertices with degree b and b vertices with degree $n-b$, and in $\mathcal{G}_2$ there are $n-b$ vertices with degree b, $b-1$ vertices with degree $n-b$, and a single vertex with degree 0. As for the size of the minimum vertex cover, all graphs in $\mathcal{G}_1$ have a vertex cover of size $b \le \frac{n-1}{4\alpha}$, and the minimum vertex cover for graphs in $\mathcal{G}_2$ is $\frac{n-1}{2}$ (since all vertices but the special vertex can be matched).

Executing the Approximation Algorithm on the Families: In order to finish the proof we must show that no algorithm that performs $o(b)$ queries can distinguish between a graph uniformly selected from $\mathcal{G}_1$ and a graph uniformly selected from $\mathcal{G}_2$. Consider the execution of any given approximation algorithm when the graph is either uniformly selected from $\mathcal{G}_1$ or is uniformly selected from $\mathcal{G}_2$. In particular, we may think of a uniformly selected graph from each of the families as being constructed in the course of the execution of the algorithm. That is, whenever the algorithm performs a query, the answer is determined according to the conditional distribution given all past



L
$|L| = b-1$, all vertices have degree $n-b$
special vertex has degree $n-b$
R
$|R| = n-b$, all vertices have degree b
Graphs in $\mathcal{G}_1$
L
$|L| = b-1$, all vertices have degree $n-b$
special vertex has degree 0
R
$|R| = n-b$, contains a perfect matching; all vertices have degree b
Graphs in $\mathcal{G}_2$

Figure 1: An illustration of the families of graphs in the lower bound proof.

queries and answers. The simple, but important observation is the following. As long as the following two events do not occur, the conditional distribution on the answers is identical for both distributions on graphs:

Event 1: The special vertex is revealed (in the course of any type of query).

Event 2: A matching edge between vertices in R is revealed (in the course of either a vertex-pair query or a neighbor query).

To verify this recall that the graphs in $\mathcal{G}_1$ and the graphs in $\mathcal{G}_2$ differ only in two related aspects: (1) In $\mathcal{G}_1$ every vertex $v \in R$ has the special vertex as one of its b neighbors, while in $\mathcal{G}_2$ every $v \in R$ has a single matching neighbor in R as one of its b neighbors (in both families all other neighbors are the vertices in L); (2) in $\mathcal{G}_1$ the special vertex is incident to every $v \in R$ (and thus has degree $n-b$), while in $\mathcal{G}_2$ it is an isolated vertex. Other than these differences, the incidence relation is determined by the common subgraph, G_0. Therefore, as long as the algorithm only views edges and vertices from G_0, it sees exactly the same distribution on answers.

We note that in order to fully formalize this argument one has to define two processes that interact with any given approximation algorithm. One process constructs a uniformly selected graph from $\mathcal{G}_1$ in the course of the interaction, and the other process constructs a uniformly selected graph from $\mathcal{G}_2$. We believe that defining such processes (which tends to be cumbersome) is not necessary in this case. For an example of a lower bound in which such processes were defined see [12].

Hence it remains to show that if the algorithm performs $o(b)$ queries then the probability that one of the above events occurs (in either distribution) is $o(1)$.

Event 1: The probability that the special vertex is observed in the t'th query when this query is either a degree query or a vertex-pair query, is $O\left(\frac{1}{n-2(t-1)}\right)$. The reason is that after $t-1$ queries, at most $2(t-1)$ different vertices were observed. Since $t = o(b) = o(n/4\alpha)$, this probability is $O(1/n)$. The probability that the special vertex is observed when answering a neighbor query (for graphs selected from $\mathcal{G}_1$, as the special vertex has no neighbors in graphs in $\mathcal{G}_2$) is $O\left(\frac{1}{b-t+1}\right)$. The reason is that after $t-1$ queries, the number of edges already incident to any vertex is at most $t-1$. Since $t = o(b)$, this probability is $O(1/b)$.

Event 2: As for matching edges (in graphs that belong to $\mathcal{G}_2$, since there are no matching edges in graphs that belong to $\mathcal{G}_1$), the probability of obtaining such an edge in the t'th query where $t = o(b) = o(n)$ is $O(1/n)$ when the query is a vertex-pair query, and is $O(1/b)$ when it is a neighbor query. The claim concerning neighbor queries is straightforward. To verify the claim concerning vertex-pair queries, consider any two vertices $u, v \in R$, and assume that the algorithm performs a vertex-pair query on this pair after performing $t-1 = o(n)$ previous queries (in which no matching edge was yet obtained). Let $H(v)$ be the subset of vertices $w \in R$ for which the algorithm queried the pair v, w (and obtained a negative answer, or else a matching edge was revealed), and define $H(u)$ analogously. Note that as long as no matching edge was revealed, conditioning on previous answers to neighbor queries does not influence the probability that there is a matching edge between v and u. This is true because as long as no matching edge was revealed, all answers to neighbor queries correspond to edges between vertices in R and vertices in L (in G_0), whereas the matching of vertices in R is selected independently (from G_0).

Consider all perfect matchings between vertices in R that are consistent with the answers that the algorithm received in its previous $t-1$ queries. We claim that amongst these, the number of matchings in which u and v are not matched is a factor of $\Omega(n)$ times larger than the number of matchings in which u and v are matched. To see why this is true, consider each matching M in which u and v are matched. Let (w, z) be any matched pair such that $w \notin H(v)$ and $z \notin H(u)$. Define $M_{w,z}$ to be the same as M except that v is matched with w and u is matched with z. Since the number of such pairs (w, z) is at least $\frac{1}{2} \cdot (n - b - 2 - |H(v)| - |H(u)|) = \Omega(n)$, we get $\Omega(n)$ different matchings for each matching M. Furthermore, the matchings defined for different M's differ in at least one edge.

Therefore, if we sum over all $o(b)$ queries, the probability that one of the events mentioned above occurs is $o(1)$, and the theorem follows. ■

6 A Lower Bound of $\sqrt{n}$ queries for a $(2-\gamma)$ approximation

In this section we give Trevisan's lower bound of $\Omega(\sqrt{n})$ on the number of queries required to obtain a $(2-\gamma)$-approximation on the minimum size of a vertex cover for any constant $0 < \gamma < 1$. In order to obtain this bound we consider two families of d-regular graphs: The family $\mathcal{G}(n, d)$ consists of all d-regular graphs over n vertices and $\mathcal{G}(n/2, n/2, d)$ consists of all d-regular bipartite graphs where each side of the partition is of size $n/2$. Clearly, $VC_G \le n/2$ for every $G \in \mathcal{G}(n/2, n/2, d)$. We shall prove:

Lemma 4 *With probability $1 - o(1)$ over the uniform choice of a random graph G in $\mathcal{G}(n, d)$ we have $\mathrm{VC}_G \ge (1 - \beta(d))n$ where $\beta(d) = O(\log(d)/d)$.*

It was shown in [15] that every algorithm that performs $o(\min\{\sqrt{n}, n/d\})$ queries cannot distinguish with more than negligible probability ($o(1)$) between a graph selected uniformly at random from $\mathcal{G}(n, d)$, and a graph selected uniformly at random from $\mathcal{G}(n/2, n/2, d)$. As a corollary we get:

Corollary 5 *For every two given constants γ and ϵ, there exists a choice of a constant d such that obtaining with a constant probability a $(2-\gamma,\epsilon)$-estimate of the size of a minimum vertex cover of graphs over n vertices and degree d, requires $\Omega(\sqrt{n})$ queries.*

We note that graphs in the families $\mathcal{G}(n,d)$ and $\mathcal{G}(n/2,n/2,d)$ may contain multiple edges and hence the bound in Corollary 5 holds for such graph. It is possible to slightly modify the construction and proofs so that the bound holds for graphs without multiple edges, as shown in [15].

Proof of Lemma 4. Consider any fixed subset $B \subset V$ of size βn, where $\beta = \beta(d)$ will be set subsequently. We would like to upper bound the probability, over the random choice of G in $\mathcal{G}(n,d)$, that B is an independent set. This is equivalent to upper bounding the probability that $V \setminus B$ is a vertex cover (where $|V \setminus B| = (1-\beta)n$).

In order to select a graph G uniformly at random from $\mathcal{G}(n,d)$ we proceed in d phases, where in each phase we select a random matching uniformly at random. In particular, the matching can be selected by choosing an arbitrary order over the vertices and matching vertex i in this order with some uniformly selected vertex that has not yet been matched. The order can be such that the vertices in B are the first βn vertices. For each matching (where the matchings are independent), the probability that no edge is selected between any two vertices in B is upper bounded by

$$\begin{aligned}\prod_{i=0}^{\beta n-1}\frac{n-\beta n-i}{n-2i} &< \prod_{i=0}^{(\beta/2)n}\frac{n-\beta n-i}{n-2i}\\ &< \left(\frac{(1-3\beta/2)n}{(1-\beta)n}\right)^{(\beta/2)n} \qquad (13)\\ &< (1-\beta/2)^{(\beta/2)n} \;<\; \exp\left(-\frac{1}{4}\beta^2\cdot n\right) \qquad (14)\end{aligned}$$

where in Equation (13) we have used the fact that $\frac{a}{b} \le \frac{a-x}{b-2x}$ for every $a<b$ such that $2a>b$ and $x<b/2$, where we set $a=n-\beta n-i, b=n-2i$ and $x=((\beta/2)n-i)$ for every $0<i\le(\beta/2)n$. The probability that no edge is selected between two vertices in B in all d perfect matchings is $\exp\left(-\frac{1}{4}\beta^2\cdot n\cdot d\right)$.

By using the bound $\binom{n}{k} \le 2^{n\cdot H(k/n)}$, where $H(\cdot)$ is the binary entropy function (e.g., see [3, page 284]), the number of subsets B of size βn is

$$\binom{n}{\beta n} \le 2^{n\cdot H(\beta)} < 2^{n\cdot\beta\log(1/\beta)}\ . \qquad (15)$$

By taking a union bound over all subsets B of size βn, we get that the probability that some subset B of size βn is an independent set (and hence $V\setminus B$ is a vertex cover) is at most

$$\exp\left(n\cdot\beta\cdot\log(1/\beta)-\frac{1}{4}\beta^2\cdot n\cdot d\right) = \exp\left(\beta\cdot n\cdot\left(\log(1/\beta)-\frac{1}{4}\beta\cdot d\right)\right)\ . \qquad (16)$$

By taking $\beta = c\cdot\log(d)/d$ for some sufficiently large constant c, the above expression is $o(1)$ as required. ■

Acknowledgments

We would like to thank Oded Goldreich for helpful conceptual discussions. We would also like to thank Asaf Shapira for a helpful comment. We thank Luca Trevisan for telling us about the lower bound described in Section 6, and for suggesting that we present a sketch of this lower bound in the paper. Finally we thank the anonymous referees of this paper for their helpful comments and suggestions.

References

[1] A. Bogdanov, Kenji Obata, and L. Trevisan. A lower bound for testing 3-colorability in bounded-degree graphs. In *Proceedings of the Forty-Third Annual Symposium on Foundations of Computer Science*, pages 93–102, 2002.

[2] 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.

[3] T. Cover and J. Thomas. *Elements of Information Theory.* Wiley, 1991.

[4] 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*, pages 175–183, 2004.

[5] A. Czumaj and C. Sohler. Sublinear-time approximation for clustering via random sampling. In *Automata, Languages and Programming: Thirty-First International Colloquium*, pages 396–407, 2004.

[6] I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–486, 2005.

[7] U. Feige. On sums of independent random variables with unbounded variance, and estimating the average degree in a graph. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 594–603, 2004.

[8] E. Fischer. The art of uninformed decisions: A primer to property testing. *Bulletin of the European Association for Theoretical Computer Science*, 75:97–126, 2001.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman, 1979.

[10] O. Goldreich. Combinatorial property testing - a survey. In *Randomization Methods in Algorithm Design*, pages 45–60, 1998.

[11] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *JACM*, 45(4):653–750, 1998.

[12] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.

[13] O. Goldreich and D. Ron. Approximating average parameters of graphs. In *Proceedings of the Tenth International Workshop on Randomization and Computation (RANDOM)*, pages 363–374, 2006.

[14] P. Indyk. A sublinear-time approximation scheme for clustering in metric spaces. In *Proceedings of the Fortieth Annual Symposium on Foundations of Computer Science*, pages 154–159, 1999.

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

[16] S. Khot and O. Regev. Vertex cover might be hard to approximate to within $2 - \epsilon$. In *Proceedings of the 18th IEEE Conference on Computational Complexity*, 2003.

[17] F. Kuhn, T. Moscibroda, and R. Wattenhofer. The price of being near-sighted. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 980–989, 2006. Also appeared as Technical Report 229, of Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland.

[18] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(2):1036–1055, 1986.

[19] S. Marko. Distance approximation in bounded-degree and general sparse graphs. Master's thesis, Dept. of Computer Science, Weizmann Institute of Science, 2005. An extended abstract based on this thesis appeared in Random 2006.

[20] S. Marko and D. Ron. Approximating the distance to properties in bounded degree and general sparse graphs. In *Proceedings of the Tenth International Workshop on Randomization and Computation (RANDOM)*, pages 475–486, 2006.

[21] N. Mishra, D. Oblinger, and L. Pitt. Sublinear time approximate clustering. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 439–447, 2001.

[22] S. Muthu. Data streams: Algorithms and applications. Available from `www.cs.rutgers.edu/~muthu`.

[23] D. Ron. Property testing. In *Handbook on Randomization, Volume II*, pages 597–649, 2001.

[24] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.