Electronic Colloquium on Computational Complexity, Report No. 94 (2005)

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

Michal Parnas
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

We consider the problem of estimating the size, $VC(G)$, of a minimum vertex cover of a graph G, in sublinear time, by querying the incidence relation of the graph. We say that an algorithm is an (α, ϵ)-approximation algorithm if it outputs with high probability an estimate $\widehat{VC}$ such that $VC(G) - \epsilon n \leq \widehat{VC} \leq \alpha \cdot VC(G) + \epsilon n$, where n is the number of vertices of G.

We show that the query complexity of such algorithms must grow at least linearly with the average degree $\bar{d}$ of the graph. In particular this means that for dense graphs it is not possible to design an algorithm whose complexity is $o(n)$.

On the positive side we first describe a simple $(O(\log(\bar{d}/\epsilon), \epsilon)$-approximation algorithm, whose query complexity is $\epsilon^{-2} \cdot (\bar{d}/\epsilon)^{\log(\bar{d}/\epsilon)+O(1)}$.

We then show a reduction from local distributed approximation algorithms to sublinear approximation algorithms. Using this reduction and the distributed algorithm of Kuhn, Moscibroda, and Wattenhofer [KMW05] we can get an $(O(1), \epsilon)$-approximation algorithm, whose query complexity is $\epsilon^{-2} \cdot (\bar{d}/\epsilon)^{O(\log(\bar{d}/\epsilon)}$.

ISSN 1433-8092

1 Introduction

As the need for computers to store and process massive data sets increases, the need for sublinear algorithms becomes evident. In recent years efforts were made to design sublinear algorithms for the central problems of computer science. These efforts can be classified into a few classes of problems.

- First, sublinear-time algorithms that obtain approximate solutions to classic problems of computer science. Among these, we mention as an example the study of sublinear algorithms for clustering (e.g., [Ind99b, MOP01, CS04b]), and the study of sublinear algorithms for estimating the weight of a minimum spanning tree of a graph (e.g., [CRT01, CS04a]). Other works in this area include [Ind99a, BCIS05].

- Another main direction of research is the area of property testing [RS96, GGR98] (see [Ron01, Fis01] for surveys). In this area the goal is to design sublinear-time algorithms, and in many cases even constant-time algorithms, to solve approximate versions of decision problems. In some cases, property testing algorithms can be adapted to obtain approximate solutions as described in the previous item (see for example [GGR98] and [ADPR03]). Recently the area of property testing was extended to tolerant property testing and distance approximation (see [PRR04] for definitions). In distance approximation the goal is to approximate in sub-linear time the distance of an object to having a given property.

- A third direction is research on streaming algorithms. In the streaming model the algorithm receives its input as a stream of data. The goal is to compute a certain desired output using space that is sublinear (preferably polylogarithmic) in the input size. For a survey on streaming see [Mut].

We extend these efforts for designing sublinear algorithms to one of the fundamental problems of computer science: approximating the size of a minimum vertex cover in a graph, or alternatively approximating the size of a maximum matching of a graph. While solving the first problem exactly is NP-complete, there exist polynomial-time algorithms that find an exact solution for the second problem. However, for both problems no sublinear-time approximation algorithms were known.

1.1 Definitions and Our Results

We consider undirected simple graphs $G = (V, E)$ where $|V| = n$ and $|E| = m$. 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, $\bar{d} = \frac{2m}{n}$). Let $VC(G)$ denote the minimum size of a vertex cover of G. In this paper we are interested in randomized algorithms that compute an estimate, $\widehat{VC}$ of $VC(G)$.

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

$$VC(G) - \epsilon n \leq \widehat{VC} \leq \alpha \cdot VC(G) + \epsilon n.$$

*An algorithm that is given ϵ as a parameter and computes with probability at least $2/3$ an (α, ϵ)-estimate of $VC(G)$ for some value of α, is an (α, ϵ)-*approximation algorithm.

We note that it is not possible to obtain a purely multiplicative approximation (i.e., $\epsilon = 0$) in sublinear-time in general. In particular it is not possible to distinguish between a graph that is an independent set (for

which the vertex cover is of size 0) and a graph that contains a single, randomly selected, edge (for which the vertex cover is of size 1) in time $o(n)$. On the other hand, suppose we have an (α, ϵ)-approximation algorithm. Then it is possible to use it in order to find a purely multiplicative estimate, where the running time will depend on $1/\nu(G)$ where $\nu(G) = \frac{VC(G)}{n}$.

The Model. We assume that the approximation algorithm is given access to a graph G in the form of a *query oracle*, where we consider three types of queries:

- *Vertex-pair* queries: For any pair of vertices $u, v \in V$, the algorithm may query whether there is an edge $(u, v) \in E$.
- *Neighbor* queries: For any vertex v and index i, the algorithm may query who is the i'th neighbor of v (if v has less than i neighbors then the algorithm receives a special symbol).
- *Degree* queries: For any vertex v, the algorithm may query what is the degree $d(v)$ of v.

Our Results. We first present a simple sublinear $(2 \log d + 1, \epsilon)$-approximation algorithm for the size of the minimum vertex cover. The query complexity of this algorithm is $O(\epsilon^{-2} \cdot d^{\log d})$. When the average degree $\bar{d}$ is significantly smaller than the average degree d, we can improve on the above and obtain an $(O(\log(\bar{d}/\epsilon)), \epsilon)$-approximation algorithm whose query complexity is $\epsilon^{-2} \cdot \left(\bar{d}/\epsilon\right)^{\log(\bar{d}/\epsilon)+O(1)}$.

Next we show a reduction from local distributed approximation algorithms to sublinear approximation algorithms. Using this reduction and the distributed algorithm of Kuhn, Moscibroda, and Wattenhofer [KMW05] we can get an $(O(1), \epsilon)$-approximation algorithm for the minimum vertex cover using $\epsilon^{-2} \cdot \tilde{d}^{O(\log \tilde{d})}$ queries, where $\tilde{d} = \min\{d, 8\bar{d}/\epsilon\}$. This reduction allows us also to obtain an $(O(\log \tilde{d}), \epsilon)$-estimate of the minimum size of a dominating set using $\epsilon^{-2} \cdot \tilde{d}^{O(\log \tilde{d})}$ queries, by applying the algorithm in [KMW05].

Finally we show that $\Omega(\bar{d})$ queries are necessary for obtaining an (α, ϵ)-estimate of $VC(G)$ for every $\epsilon < 1/8$ and when $\bar{d} = O(n/\alpha)$. This bound holds when the algorithm is allowed all types of queries. In particular this implies that for dense graphs (that is, $\bar{d} = \Theta(n)$), and constant α, a linear (in n) number of queries is necessary.

2 General Lower Bounds on $VC(G)$

We start by giving two simple lower bounds on $VC(G)$ as a function of n, d and $\bar{d}$. Since $VC(G) \leq n - 1$ we immediately get some approximation of $VC(G)$. We state our bounds in terms of the size of a maximal matching M in G. Since $|M| \leq VC(G) \leq 2|M|$, bounds on M directly imply bounds on $VC(G)$. We note that the first lemma slightly improves the bound given in [BDD$^+$04].

Lemma 1 *In every graph G there is a matching of size at least $\frac{m}{2d-2} = \frac{n}{2} \cdot \frac{\bar{d}}{2(d-1)}$.*

Observe that the lower bound in Lemma 1 depends on the ratio between $\bar{d}$ and d (or, more precisely, $\bar{d}$ and $d - 1$). This bound is not exactly tight, in the sense that it is known that for regular graphs ($\bar{d} = d$) there is a perfect matching of size $n/2$, while the lower bound is roughly $n/4$. However, the lemma demonstrates the following general behavior: the more regular the graph is (i.e., as $\bar{d}$ is closer to d), the larger is the lower bound.

Proof: Consider the following process for finding a maximal matching M: Starting from the original graph G, until there are no remaining edges in the graph, or all vertices with non-zero degree have the same degree in the current graph, pick an edge (v, w) that is incident to a vertex with minimum degree in the current graph. Add the edge to M and remove this edge and all other edges that are incident to v and w.

If the above procedure stops when there are still edges in the graph, then there is perfect matching between the remaining vertices, and we add all these edges to M.

We claim that $|M| \geq m/(2d-2)$. To verify this observe that in each of the first steps (before we reach, if at all, a perfect matching), for each edge that is added to the matching, we "throw away" at most $(2d-2)$ edges. In the last step, if m' is the remaining number of edges, then we have a matching of size at least m'/d. ■

When the graph is connected but very sparse (i.e., $d < 2$), then the following lower bound slightly improves the bound in Lemma 1.

Lemma 2 *In every connected graph there is a matching of size at least* $\frac{n-1}{d}$.

Proof: Since the graph is connected it has a spanning tree T. Using T we construct a matching M as follows: Let u be a leaf of the tree, and let v be it's parent in the tree. Add the edge (u, v) to M, and remove from T this edge and all other edges that are incident to v. Now T is disconnected into a few connected subtrees. Continue in the same manner with each one of the subtrees until all edges of T are removed.

As to the size of M, note that the spanning tree has $n-1$ edges, and for each edge that is added to M, at most d edges are removed from T. Thus $|M| \geq (n-1)/d$. ■

It directly follows from Lemma 2 that if a graph G has $CC(G) > 1$ connected components, then there is a matching of size at least $\frac{n-CC(G)}{d}$.

3 A Simple Sublinear Approximation Algorithm

We first present an algorithm for approximating the size of a minimum vertex cover where the algorithm is not sublinear. We later show how to use this algorithm in order to obtain a sublinear approximation algorithm. We assume for now that the maximum degree d of the graph is known. We later remove this assumption.

Algorithm 1 (Approximate size of minimum vertex cover)

1. *Let $C = \emptyset$ be the initial vertex cover, and let $i = 1$.*
2. *While $E \neq \emptyset$ do:*
   (a) *Add all vertices whose degree is at least $d/2^i$ to C.*
   (b) *Remove from the graph all edges that are incident with the vertices added to C.*
   (c) *i = i+1.*
3. *Output $|C|$.*

Theorem 1 *Algorithm 1 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 use Algorithm 1 in order to obtain a sublinear randomized algorithm that outputs with high probability an (α, ϵ)-estimate of $VC(G)$. The query complexity of the algorithm is $O\left(\epsilon^{-2} \cdot d^{\log d}\right)$. We later show how to replace the dependence on the maximum degree d with a dependence on the average degree $\bar{d}$.

The idea is to select uniformly and independently $\Theta(1/\epsilon^2)$ vertices, and for each sampled vertex to determine if it would be added by Algorithm 1 to the vertex cover. This is done by examining for each sampled vertex v, the $\log d$-neighborhood of v.

Algorithm 2 (Sublinear approximation algorithm for $VC(G)$)

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

The query complexity of Algorithm 2 is as claimed previously: building the subgraph $G_k(v)$ can be done by constructing a BFS tree of depth $k = \log d$ rooted at v, using $O(d^k)$ neighbor queries. Then we can run Algorithm 1 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.

Theorem 2 *Let G be an undirected graph with degree at most d. Then Algorithm 2 is a $(2 \log d + 1, \epsilon)$-approximation algorithm for the size of the minimum vertex cover.*

Proof: By an additive Chernoff bound, if $s = 2/\epsilon^2$, then the probability that $\widehat{VC} > |C| + \epsilon n$ or $\widehat{VC} < |C| - \epsilon n$ (where $|C|$ is the output of Algorithm 1) is at most $2 \cdot e^{-2s\epsilon^2} < 1/3$. The theorem follows by applying Theorem 1. ■

3.1 Dependence on the average degree

It is possible to replace the dependence that the algorithm has on the maximum degree d by a dependence on the average degree $\bar{d}$, or more precisely on $O(\bar{d}/\epsilon)$. For the sake of simplicity we present this modification for Algorithm 2, but it can be applied to any (α, ϵ)-approximation algorithm for the size of the minimum vertex cover.

In order to achieve the dependence on the average degree we slightly modify Algorithm 1 and Algorithm 2. We first define a procedure that will be used by the modified algorithms. For any integer $0 \leq i \leq n$, let $V_{\geq i} = \{v : \deg(v) \geq i\}$ be the set of vertices of degree at least i.

Procedure Compute $\hat{d}$:

1. Sample $q = O(1/\epsilon)$ vertices, and for each sampled vertex query its degree.
2. Set $\hat{d}$ to be 1 plus the $(\epsilon/4)q$ largest degree in the sample.

Claim 3 *With probability at least $5/6$ over the choice of the sample selected by Procedure Compute $\hat{d}$ it holds: (1) $|V_{\geq \hat{d}}| \leq \epsilon n/2$ and (2) $\hat{d} \leq 8\bar{d}/\epsilon$.*

Proof: To verify the first claim, consider an ordering of the vertices in G according to their degree (from large to small, breaking ties arbitrarily). Let $\tilde{d}_{\epsilon/2}$ be the degree of the $\frac{\epsilon}{2}n$ vertex in this order. By definition

$$|V_{\geq \tilde{d}_{\epsilon/2}}| \geq \frac{\epsilon}{2}n\,, \quad \text{while} \quad |V_{\geq \tilde{d}_{\epsilon/2}+1}| < \frac{\epsilon}{2}n \tag{1}$$

Therefore, if the number of sampled vertices that belong to $V_{\geq \tilde{d}_{\epsilon/2}}$ is at least $\frac{\epsilon}{4}q$, then necessarily $|V_{\geq \hat{d}+1}| < \frac{\epsilon}{2} \cdot n$. But by a multiplicative Chernoff bound, the probability that there are less than $\frac{\epsilon}{4}q$ vertices in the sample that belong to $V_{\geq \tilde{d}_{\epsilon/2}}$, is $\exp^{-\Omega(\epsilon q)}$, which for our choice of q is a small constant.

To verify the second claim observe that $|V_{\geq 8\bar{d}/\epsilon}| \leq \frac{\epsilon}{8}n$. Hence, once again by a multiplicative Chernoff bound, the probability that there are at least $\frac{\epsilon}{4}q$ vertices in the sample that belong to $V_{\geq 8\bar{d}/\epsilon}$ is $\exp^{-\Omega(\epsilon q)}$. But if there are less than $\frac{\epsilon}{4}q$ vertices in the sample that belong to $V_{\geq 8\bar{d}/\epsilon}$, then necessarily $\hat{d} \leq 8\bar{d}/\epsilon$ as claimed. ■

We now describe the modifications needed in Algorithms 1 and 2. First we add the following two initial steps to Algorithm 1:

1. Call Procedure Compute $\hat{d}$.
2. Add to the vertex cover all vertices in the graph of degree $\geq \hat{d}$, and remove all their incident edges.

By Claim 3 with probability at least $5/6$ the number of vertices added to the vertex cover by the second step of the modified algorithm is at most $\frac{\epsilon}{2}n$, and the resulting graph has a maximum degree of at most $8\bar{d}/\epsilon$. Now Algorithm 1 continues its execution on the resulting graph. The number of iterations of the modified algorithm is hence $\log \hat{d}$, and the size of the final vertex cover is at most $(2\log \hat{d} + 1) \cdot VC(G) + \frac{\epsilon}{2}n$.
As to Algorithm 2, it is modified correspondingly as follows:

1. Call Procedure Compute $\hat{d}$ and set $s = 16/\epsilon^2$.

2. Use the modified Algorithm 1 to decide for each selected vertex whether it belongs to the vertex cover or not (where there is no need to call Procedure Compute $\hat{d}$ for each selected vertex since $\hat{d}$ was already computed).

Since $s = 16/\epsilon^2$, by an additive Chernoff bound, with probability at least $5/6$, $\left| \widehat{VC} - |C_{\hat{d}}| \right| \leq \frac{\epsilon}{2}n$, where $C_{\hat{d}}$ is the vertex cover constructed by Algorithm 1 given $\hat{d}$. By adding up the probability that $\hat{d}$ does not have the properties stated in Claim 3 with the probability that $\widehat{VC}$ deviates by more than $\frac{\epsilon}{2}n$ from $|C_{\hat{d}}|$ we get that with probability at least $2/3$ we obtain an estimate $\widehat{VC}$ such that

$$VC(G) - (\epsilon/2)n \leq \widehat{VC} \leq (2\log \hat{d} + 1) \cdot VC(G) + \epsilon n \; . \tag{2}$$

Note that if, when constructing the $(\log \hat{d})$-neighborhood of a selected vertex v (by constructing its BFS tree), we reach a vertex that has degree greater than $\hat{d}$, then we do not need to continue the BFS from that vertex. Hence the number of queries performed by the sublinear algorithm is

$$O\left(\epsilon^{-2} \cdot \hat{d}^{\log \hat{d}}\right) = O\left(\epsilon^{-2} \cdot \left(8\bar{d}/\epsilon\right)^{\log(8\bar{d}/\epsilon)}\right) = \epsilon^{-2} \cdot (\bar{d}/\epsilon)^{\log(\bar{d}/\epsilon)+O(1)} \tag{3}$$

Note that $\bar{d} \leq d$ and so the modified algorithm cannot have a worse performance than the original algorithm.

3.2 Obtaining a purely multiplicative approximation

Recall that in the introduction we defined $\nu(G) = \frac{VC(G)}{n}$. In this subsection we explain how to use any (α, ϵ)-approximation algorithm A (e.g., Algorithm 2) in order to get a purely multiplicative factor approximation at a cost that depends on $1/\nu(G)$, conditioned on $\nu(G) > 0$. That is, the smaller is the minimum vertex cover, the larger is the query complexity of the algorithm.

Specifically, if the query complexity of A is $Q_A(n, d, \bar{d}, \epsilon)$ (where this function is assumed to be monotonically non-decreasing with $1/\epsilon$), then we can obtain an estimate $\widehat{VC}$ that with probability at least $2/3$ satisfies $VC(G) \leq \widehat{VC} \leq 2\alpha \cdot VC(G)$ by performing $O\Big(Q_A(n, d, \bar{d}, \nu(G)/4) \cdot \log^2(1/\nu(G))\Big)$ queries. As is described in detail below, this is done by running algorithm A with decreasing values of ϵ. In each iteration we have (with high probability) an interval to which $VC(G)$ belongs. We stop when the interval is sufficiently small. We note that the approximation factor of 2α can be reduced to $(1+\gamma)\alpha$ for any $\gamma < 1$ by introducing a dependence on $1/\gamma$, but for simplicity, we show it just for $\gamma = 1$.

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 $\widehat{VC}_i$ be the output of A in the i'th iteration, and let $\widetilde{VC}_i = \widehat{VC}_i + 2^{-i}n$. By definition of algorithm A, with probability at least $1 - \frac{1}{3} \cdot 2^{-i}$, we have that $VC(G) \leq \widetilde{VC}_i \leq \alpha VC(G) + 2^{-i+1}n$, or equivalently

$$\frac{1}{\alpha} \cdot (\widetilde{VC}_i - 2^{-i+1}n) \;\leq\; VC(G) \;\leq\; \widetilde{VC}_i \tag{4}$$

The procedure terminates once the ratio between the upper bound and the lower bound on $VC(G)$ in Equation (4) is at most 2α, conditioned on the lower bound being positive. That is, when $\frac{\widetilde{VC}_i}{\widetilde{VC}_i - 2^{-i+1}n} \leq 2$ and $\widetilde{VC}_i - 2^{-i+1}n > 0$. It then outputs the lower bound, $\frac{1}{\alpha} \cdot (\widetilde{VC}_i - 2^{-i+1}n)$. Assume that in each iteration

we have that $VC(G) \leq \widetilde{VC}_i \leq \alpha VC(G) + 2^{-i+1}n$, 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 $i \leq \log(4/\nu(G))$. To verify this, let $\widetilde{VC}_i = \beta VC(G) = \beta\nu(G)n$, where we have that $1 \leq \beta \leq \alpha + \frac{1}{2}$. Then $\frac{\widetilde{VC}_i}{\widetilde{VC}_i - 2^{-i+1}n} = \frac{\beta}{\beta-(1/2)}$. Since $\beta \geq 1$, this ratio is at most 2, as required. Also note that since $\widetilde{VC}_i \geq \nu(G) \cdot n$, we also have that $\widetilde{VC}_i - 2^{-i+1}n > 0$ (for $i \leq \log(4/\nu(G))$).

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

4 From Local Distributed Approximation Algorithms to Sublinear Approximation Algorithms

In this section we show a reduction from distributed approximation algorithms to sublinear approximation algorithms. 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 the number of rounds 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 problem of vertex cover, and then specify what should hold so that the reduction is true also for other problems and settings.

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

$$|C| - \epsilon n \leq \widehat{VC} \leq |C| + \epsilon n.$$

The query complexity of the sublinear algorithm is $O(\epsilon^{-2} \cdot d^k)$.

Proof: The sublinear algorithm is obtained from algorithm A analogously to the way Algorithm 2 was obtained from Algorithm 1:

Algorithm 3 (Reduction algorithm)

1. *Uniformly and independently select $s = 2/\epsilon^2$ vertices from G. Let the subset (multiset) of selected vertices be denoted 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 A 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 A decided to add v to the vertex cover C, and otherwise $X_v = 0$.
4. *Output $\widehat{VC} = \frac{n}{s} \cdot \sum_{v \in S} X_v$.*

The simple but important observation is that for any vertex v, if we run Algorithm A on the subgraph $G_k(v)$, then it makes the same decision about vertex v as it would if we would run A 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 3 as it 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 A only on it's 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 $s = 2/\epsilon^2$, then with probability at least $2/3$ we get: $|C| - \epsilon n \leq \widehat{VC} \leq |C| + \epsilon n$. ■

Kuhn, Moscibroda, and Wattenhofer [KMW05] prove the following theorem:

Theorem 4 ([KMW05]) *For any integer k such that $k = O(\log d)$, it is possible to compute 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 a sufficiently small γ, it is possible to get a $2(1+\gamma)$-approximation in $O((\log d)/\gamma)$ rounds.*

By combining Theorem 3 with Theorem 4 we can get an $(O(1), \epsilon)$-estimate of $VC(G)$ using $\epsilon^{-2} \cdot d^{O(\log d)}$ queries. Here too we can replace the dependence on d by a dependence on $\Theta(\bar{d}/\epsilon)$.

4.1 Extension of the Reduction to Other Settings

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

1. **Randomized 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 $G_k(u)$, 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 distributed algorithm that approximates some global function $f(G)$ of the graph, as long as f is a weighted sum of some subset of the vertex set and the weights are in a bounded range. For example, it is possible to obtain an $(O(\log \tilde{d}), \epsilon)$-estimate of the minimum size (weight) of a dominating set using $\epsilon^{-2} \cdot \tilde{d}^{O(\log \tilde{d})}$ queries, where $\tilde{d} = \min\{d, 8\bar{d}/\epsilon\}$, by applying the algorithm in [KMW05].

5 A Lower Bound

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

$\Theta(n)$, we get that any constant factor approximation requires $\Omega(n)$ queries. Also recall that by Lemma 1, every graph with m edges and maximum degree d has a vertex cover of size $\frac{m}{2d-2} = n \cdot \frac{\bar{d}}{4(d-1)}$ where $\bar{d}$ is the average degree. Since $d < n$, this lower bound is greater than $\frac{\bar{d}}{4}$. Since the size of a minimum vertex cover is at most $n-1$, any lower bound on the query complexity of (α, ϵ)-approximation algorithms can hold only when the average degree is $O(n/\alpha)$, as is the case in Theorem 5.

Theorem 5 *For any n, $\alpha > 1$, $b \leq \frac{n-1}{4\alpha}$ and $\epsilon < 1/8$, an (α, ϵ)-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.*

Proof: We define two families of n-vertex graphs denoted $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. 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.

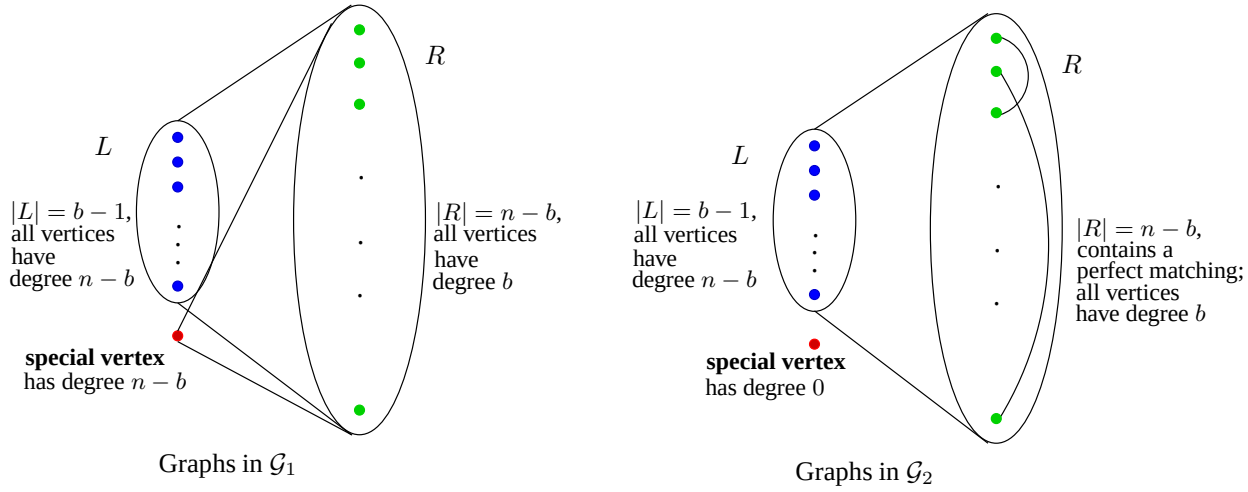

Figure 1: An illustration of the families of graphs in the lower bound proof.

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 withe 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 $b \leq \frac{n-1}{4\alpha}$, 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).

By definition, for graphs in $\mathcal{G}_1$, any (α, ϵ)-approximation algorithm should output (w.h.p.) an estimate $\widehat{VC} \leq \alpha \cdot \left(\frac{n-1}{4\alpha}\right) + \epsilon n$ while for graphs in $\mathcal{G}_2$ it should hold that $\widehat{VC} \geq \frac{n-1}{2} - \epsilon n$. Since $\epsilon < 1/8$ we have

that $\alpha \cdot \left(\frac{n-1}{4\alpha}\right) + \epsilon n < \frac{n-1}{2} - \epsilon n$. Hence 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 $\mathcal{G}_2$.

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 queries and answers. The simple, but important observation is the following. As long as the special vertex is not revealed (in the course of any type of query) and a matching edge between vertices in R is not revealed (in the course of either a vertex-pair query or a neighbor query), the conditional distribution on the answers is identical for both distributions on graphs.

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 [GR02].

Hence it remains to show that if the algorithm performs $o(b)$ queries then the probability that one of these events occurs (in either distributions) is $o(1)$. 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 $O\left(\frac{n-2(t-1)}{n}\right)$. 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 $\mathcal{G}_1$) is $O\left(\frac{1}{b-t+1}\right)$. The reason is that after $t-1$ queries, the number of edges already incident to a vertex is at most $t-1$. Since $t = o(b)$, this probability is $O(1/b)$.

As for matching edges (in graphs that belong to $\mathcal{G}_2$), 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), and define $H(u)$ analogously.

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 $\Omega(n)$ 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. ■

References

[ADPR03] N. Alon, S. Dar, M. Parnas, and D. Ron. Testing of clustering. *SIAM Journal on Discrete Math*, pages 393–417, 2003.

[BCIS05] M. Bădoiu, A. Czumaj, P. Indyk, and C. Sohler. Facility location in sublinear time. In *Automata, Languages and Programming: Thirty-Second International Colloquium*, 2005.

[BDD+04] T. Biedl, E.D. Demaine, C.A. Duncan, R. Fleischer, and S.G. Kobourov. Tight bounds on maximal and maximum matchings. *Discrete Mathematics*, 285(1–3):7–15, 2004.

[CRT01] B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. In *Automata, Languages and Programming: Twenty-Eighth International Colloquium*, pages 190–200, 2001.

[CS04a] 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.

[CS04b] 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.

[Fis01] 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.

[GGR98] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *JACM*, 45(4):653–750, 1998.

[GR02] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.

[Ind99a] P. Indyk. Sublinear time algorithms for metric space problems. In *Proceedings of the Thirty-First Annual ACM Symposium on the Theory of Computing*, pages 428–434, 1999.

[Ind99b] 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.

[KMW05] F. Kuhn, T. Moscibroda, and R. Wattenhofer. The price of being near-sighted. Technical Report 229, of Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, 2005.

[MOP01] 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.

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

[PRR04] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. ECCC Report TR04-010, 2004.

[Ron01] D. Ron. Property testing. In *Handbook on Randomization, Volume II*, pages 597–649, 2001.

[RS96] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'