Sublinear Algorithms March 13, 2012

Lecture 14 – 15

Lecturer: Sofya Raskhodnikova *Scribe(s): Grigory Yaroslavtsev*

1 Approximating the size of the Vertex Cover

We will consider a sublinear algorithm for approximating the size of the minimum Vertex Cover, proposed by Parnas and Ron [4]. The algorithm can query degrees of vertices in the graph and for each vertex v can query the i-th neighbor of v for $1 \leq i \leq deg(v)$. Best known sublinear algorithm for approximating the size of the minimum Vertex Cover is due to Onak, Ron, Rosen and Rubinfeld [3]. They give an algorithm with complexity $\tilde{O}(\bar{d} \cdot poly(1/\epsilon))$, where $\bar{d}$ is the average degree in the graph.

We will denote the size of the minimum Vertex Cover in a graph $G(V, E)$ as $VC(G)$. Note that if G has maximum degree at most d, then $VC(G) \geq n/d$. This is why if we are looking for a sublinear algorithm, we can not output the Vertex Cover itself. Our algorithms will give an approximation of the size of the Vertex Cover.

The main idea in the sublinear algroithms we will see today is to use distributed alorithms that output a Vertex Cover. We will do a reduction of our problem to its distributed version via random sampling.

1.1 2-approximation for Vertex Cover

Vertex Cover is a Set-Cover problem with the size of each set equal to 2. We can use a k-approximation for the k-Set-Cover problem to get a 2-approximation for it. This is done using the following LP for Set-Cover:

$$\begin{array}{lll} \text{Minimize:} & \sum_{i \in V} x_i & \\ \text{Subject to:} & x_i + x_j \geq 1 & \forall (i,j) \in E \\ & x_i \geq 0 & \forall i \in V. \end{array}$$

An integral solution to this LP with $x_i \in \{0, 1\}$ for all i gives an optimum solution to the Vertex Cover problem. If we relax the integrality constraints and let x_i's be real variables $x_i \in [0, 1]$, then we can solve the LP above in polynomial time. Given a solution $\{x_i\}$ to the LP we use deterministic rounding: construct a set $C = \{i | x_i \geq 1/2\}$. Every edge $e \in E$ will be covered by C because at least one of the variables in the constraint corresponding to e is at least $1/2$. The size of the solution $|C| \leq 2 \sum_{i \in V} x_i \leq 2OPT_{LP} \leq 2 \cdot VC(G)$, where OPT_{LP} is the optimum value of the objective function in the LP, which gives a lower bound on $VC(G)$.

Dinur and Safra [1] showed that it is NP-hard to approximate Vertex Cover to a factor better than 1.36 and Khot and Regev [2] showed that it assuming Unique Games Conjecture it is hard to approximate to a factor better than $2 - \epsilon$ for any $\epsilon > 0$.

1.2 Sublinear algorithm for the size of Vertex Cover

In the discussion of sublinear algorithms for Vertex Cover we will use an approximation, which has both multiplicative factor and an additive term.

Definition 1 ((α, ϵ)-approximation)**.** *We say that $\hat{y}$ is an (α, ϵ)-approximation for $VC(G)$, if:*

$$VC(G) \leq \hat{y} \leq \alpha VC(G) + \epsilon |V|,$$

for $\alpha > 1, \epsilon < 1$.

Theorem 2 ([4]). *There exists* $(2\log d + 1, \epsilon)$*-approximation algorithm* Vertex Cover *problem, which makes* $\frac{d^{O(\log d)}}{\epsilon^2}$ *queries.*

To explain the main idea we introduce a model for distributed computation of Vertex Cover. We think of an input graph as a communication network, with each node being a processor, such that the network has the following properties:

- Each node can talk directly to its neighbours in 1 round.
- Network is synchronous (a node can translate a message to all of its neighbours simultaneously).
- Every node knows the bound on the maximum degree.
- Nodes perform k rounds of communication and then every node decides, whether it is in the Vertex Cover.

Now we show how to do the reduction of the original problem to the distributed problem. Suppose we have an algorithm D for the distributed Vertex Cover problem. We show how to construct a sublinear algorithm based on it.

First, suppose that an algorithm D for the distributed problem is deterministic. We will need the following definition of neighbourhood.

Definition 3 (k-neighborhood)**.** *For a vertex v, its k-neighbourhood is a subgraph of all nodes and edges that can be reached from v by paths of length at most k. If the maximum degree in the graph is d, then for any vertex the size of its k-neighbourhood is $\sum_{i=0}^{k} d^i \leq d^{k+1}$.*

Claim 4. *If there exists a deterministic exact (α-approximate) distributed algorithm D for* Vertex Cover*, which takes k rounds, then there exists is an $(1, \epsilon)$-approximation ((α, ϵ)-approximation) sublinear algorithm for* Vertex Cover*, which makes $O\left(\frac{d^{k+1}}{\epsilon^2}\right)$ queries.*

Proof. First, let's assume that we are given an exact distributed algorithm D. Our sublinear algorithm will pick a sample set S of vertices of size $|S| = \frac{8}{\epsilon^2}$ uniformly at random and then run D for each of them to decide, whether or not it is in the Vertex Cover. Let's denote the set constructed as above as S_C. Then our algorithm will output an approximation to the size of the vertex cover as $A = |S_C||V|/|S| + \frac{\epsilon}{2}|V|$.

Note that to run D for a vertex v, we only need to simulate D on the k-neighbourhood of v, because any message that can affect the decision of v has to reach it within k rounds and thus has to be sent by some node in its k-neighbourhood. Now correctness follows from the observation that our algorithm selects a vertex into a Vertex Cover if and only if D does. The guarantee on approximation follows from the Chernoff bound. If for a vertex v we use an indicator random variable X_v for an event that the algorithm above selects v into the vertex cover then we can use additive Chernoff bound to get the desired approximation gurarantee. We have $\mathbb{E}[X_v] = \frac{1}{|V|}\sum_{v\in V} X_v$ and $VC(G) = n\mathbb{E}[X_v]$. By the Chernoff bound:

$$\Pr\left[\left|\frac{1}{|S|}\sum_{v\in S} X_v - \mathbb{E}[X_v]\right| \geq \frac{\epsilon}{2}\right] \leq \frac{1}{3}.$$

Note that we add an additive term $\frac{\epsilon}{2}|V|$, which guarantees that our output A satisfies $VC(G) \leq A \leq VC(G) + \epsilon|V|$ with probability at least $1/3$.

However, exact distributed algorithms for Vertex Cover with small number of rounds do not exist, so we are going to use an approxiation algorithm for distributedt Vertex Cover. If D gives a multiplicative α-approximation for the Vertex Cover, then then the same analysis as above will give an (α, ϵ)-approximation. □

Now we describe an approximate distributed algorithm for Vertex Cover.

input : G(V,E)

$C = \emptyset$
for $i = 1$ **to** $\log d$ **do**
  Each vertex of degree $\geq d/2^i$ adds itself to C
  For all vertrices added in the previous step, remove all edges adjacent to them from the graph.
  Recompute the degrees in the residual graph
end
Output C

Algorithm 1: Distributed-Vertex-Cover(G).

Theorem 5. *For the algorithm above we have* $OPT \leq |C| \leq (2\log d + 1)OPT$.

Proof. Clearly, the algorithm above returns C, such that $|C| \geq VC(G)$.

The second inequality follows from an auxiliry claim:

Claim 6. *Let* Θ *be the minimum Vertex Cover and* $\bar{\Theta} = V \setminus \Theta$. *Then the number of new nodes from* $\bar{\Theta}$ *added to* C *at each iteration is at most* $2|\Theta|$.

Proof. Let $G_i(V_i, E_i)$ be the graph remaining at the beginning of iteration i. For each node in G_i, its degree is at most $2\frac{d}{2^i}$. Let X_i be the nodes from $\bar{\Theta}$ added in iteration i.

Because $\bar{\Theta}$ is an independent set, there are no edges going from X_i to $\bar{\Theta}$. Let $e(X_i, \Theta)$ be the number of edges between X_i and Θ (see Figure 1).

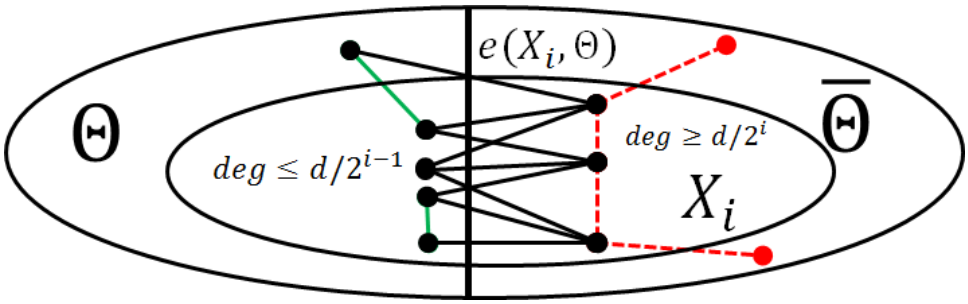

Figure 1: Partition.

We have $|X_i|\frac{d}{2^i} \leq e(X_i, \Theta) \leq |\Theta|\frac{2d}{2^i}$. This is why $|X_i| \leq 2|\Theta|$. □

Using the claim above we have $|C| \leq |\Theta| + \log d \cdot 2VC(G) \leq (2\log d + 1)VC(G)$. □

1.3 $(2, \epsilon)$-approximation based on Maximal Matching

Our algorithm will have the same approximation as the best known algorithm in [3], but the running time will be $2^{O(d)}poly(\frac{1}{\epsilon})$, while the best known running time is $\tilde{O}(d \cdot poly(1/\epsilon))$

A *maximal* matching is a matching, which can not be extended by adding edges to it. For each maximal matching M we have $|M| \leq VC(G) \leq 2|M|$, because for each edge in M we have to cover at least one of its endpoints and if we take both endpoints for every edge we will cover every edge by maximality of M. A maximal matching can be constructed greedily in linear time by an algorithm below.

We will use the following modification of this greedy algorithm to construct a Vertex Cover C. We will pick a sample of vertices S of size $|S| = O(1/\epsilon^2)$ as before. For every node in S we will check, whether there is an edge from a minimal matching M adjacent to it. If there exists such an edge, then a vertex will add itself to C.

input : G(V,E)

$M = \emptyset$
for *every edge* $(u, v) \in E$ **do**
  If neither u nor v are matched, add (u, v) to M
end
Output M

Algorithm 2: Greedy-Vertex-Cover(G).

To implement this algorithm we need to implement an oracle, which for a given edge decides, whether this edge is in M. For this we will use a minimal matching M, which is constructed by a greedy algorithm above with edges ordered uniformly at random. To select random ordering of the edges we will sample a random variable $r_e \in [0, 1]$ for each edge in E and consider the ordering of edges in increasing order of r_e's (we assume all r_e's are distinct.)

The following invariant in maintained for the set M constructed by the greedy algorithm with the ordering we consider: an edge $e = (u, v)$ is in M iff none of the neighboring edges e' of u and v, such that $r_{e'} \leq r_e$ are in M.

This is why to simulate the greedy algorithm for the ordering we consider when we need to decide, whether $e \in E$ is in M we use the following recursive procedure: for each e' adjacent to E we run the decision procedure recursively iff $r_{e'} < r_e$. Then we take an edge in a matching M if and only if none if its neighbors are in the matching (see Figure 2, from lecture notes for a class by Ronitt Rubinfeld http://www.cs.tau.ac.il/ ronitt/COURSES/F09course/HANDOUTS/l3.pdf).

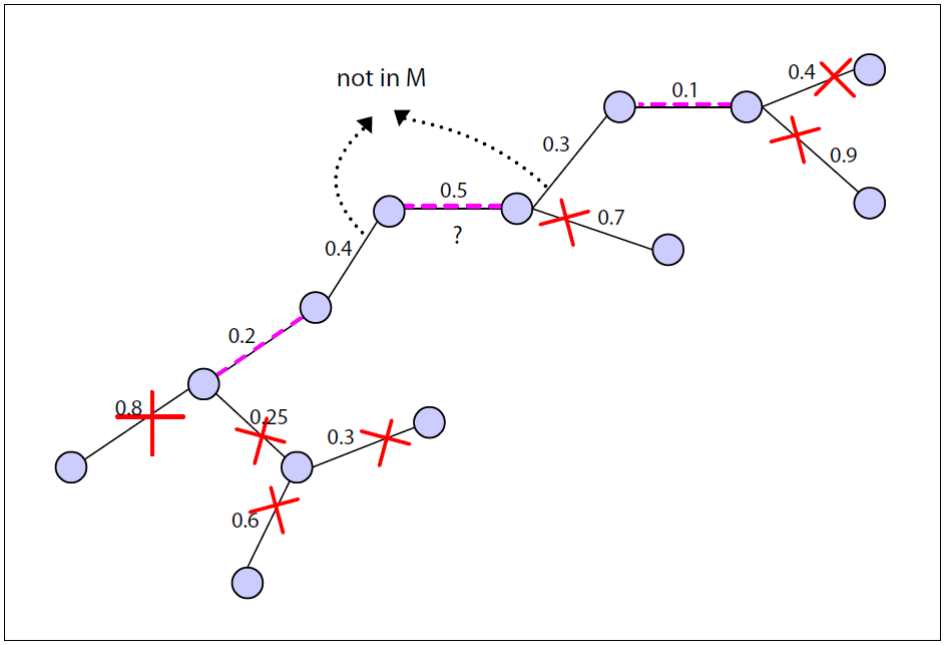

Figure 2: Improved oracle implementation

Note that algorithm simulates the greedy VERTEX COVER algorithm above with the edges ordered according to their values r_e. This is why the decisions that such an algorithm makes for every edge are exactly the same as the decisions of the greedy algorithm for this ordering of edges.

Claim 7. *For each edge $e \in E$ the expected number of edges queried to decide whether e is in M is $2^{O(d)}$.*

Proof. Fix some path of length k starting at e.

$$\Pr[\text{A fixed path of length } k \text{ is explored}] = \Pr[\text{Labels of edges on the path are decreasing}] = \frac{1}{k!}.$$

The number of paths of length k is at most d^k, so $\mathbb{E}[\text{number of edges explored}] \leq \sum_{k=0}^{\infty} \frac{d^k}{k!} \leq e^d$. □

References

[1] I. Dinur and S. Safra. The importance of being biased. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 33–42, New York, NY, USA, 2002. ACM.

[2] S. Khot and O. Regev. Vertex cover might be hard to approximate to within 2-ϵ;. In *Computational Complexity, 2003. Proceedings. 18th IEEE Annual Conference on*, pages 379 – 386, july 2003.

[3] K. Onak, D. Ron, M. Rosen, and R. Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 1123–1131. SIAM, 2012.

[4] M. Parnas and D. Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theor. Comput. Sci.*, 381(1-3):183–196, 2007.