Sublinear Time and Space Algorithms 2016B – Lecture 7
Sublinear-Time Algorithms for Sparse Graphs*

Robert Krauthgamer

1 Approximating Average Degree in a Graph

Problem definition:

Input: A graph represented (say) as the adjacency list for each vertex (or even just the degree of each vertex)

Goal: Compute the average degree (equiv. number of edges)

Concern: Seems to be impossible e.g. if all degrees ≤ 1, except possibly for a few vertices whose degree is about n.

Theorem 1 [Feige, 2004]: There is an algorithm that estimates the average degree d of a *connected* graph within factor $2+\varepsilon$ in time $O((\frac{1}{\varepsilon})^{O(1)}\sqrt{n/d_0})$, given a lower bound $d_0 \leq d$ and $\varepsilon \in (0,1)$.

We will prove the case of $d_0 = 1$ (i.e., suffices to know G is connected).

Algorithm:

1. Choose a set S by choosing at random $s = c\sqrt{n}/\varepsilon^{O(1)}$ vertices, and compute the average degree d_S of these vertices.

2. Repeat the above $8/\varepsilon$ times, and report the smallest seen d_S.

Analysis: We will need 2 claims.

Claim 1a: In each iteration, $\Pr[d_S < (\frac{1}{2} - \varepsilon)d] \leq \varepsilon/64$.

Claim 1b: In each iteration, $\Pr[d_S > (1+\varepsilon)d] \leq 1 - \varepsilon/2$.

Proof of theorem: Follows easily from the two claims, as seen in class.

Proof of Claim 1b: Follows from Markov's inequality, as seen in class.

Proof of Claim 1a: Was seen in class. Here we really used the fact the degrees form a graph.

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.

Exer: Explain how to extend the result to any $d_0 \geq 1$.

2 Maximum Matching

Problem definition:

Input: A graph $G = (V, E)$ of maximum degree D, represented as the adjacency list for each vertex.

Definition: A matching is a set of edges that are incident to distinct vertices.

Goal: Compute the maximum size of a matching in G.

Note: The matching is too large to report in sublinear time, we only estimate its cost using (α, β)-approximation, i.e., $OPT \leq ALG \leq \alpha\ OPT + \beta$.

Theorem 2 [Nguyen and Onak, 2008]: There is an algorithm that gives $(2, \varepsilon n)$ approximation to the maximum matching size in time $D^{O(D)}/\varepsilon^2$.

Main idea: It is well-known that maximal matching (note: maximal means with respect to containment) is a 2-approximation for maximum matching. We will fix one such matching almost implicitly, and then estimate its size by sampling.

Algorithm GreedyMatching:

1. Start with an empty matching M.

2. Scan the edges (in arbitrary order), and add each edge to M unless it is adjacent to an edge already in M.

Lemma 2a: The size of a maximal matching is at least half that of a maximum matching.

Proof: Exercise

Algorithm ApproxGreedyMatching: Choose (implicitly) a permutation of the edges via a random edge priority $p(e) \in [0, 1]$. Choose $s = O(D/\varepsilon^2)$ edges $e_1, \ldots, e_s$ uniformly at random from the Dn possibilities (note that each edge has two "chances" to be chosen, and some choices may lead to no edge, if the actual degree is smaller than D). Let X_i be an indicator for whether each edge e_i belongs to the maximal matching corresponding to p. Compute each X_i by exploring the neighborhood of e_i incrementally, and report $X = \frac{Dn}{2s} \sum_i X_i$. [Stop if altogether it required too many steps.]

Analysis:

Correctness: As seen in class, to determine whether $e_i \in M$, whp it suffices to explore up to radius $k = O(D)$.

Runtime: expectation is at most $O(s \cdot D^k) \leq D^{O(D)}/\varepsilon^2$. The probability to exceed this by much is small by Markov's inequality.

3 Vertex Cover in Planar Graphs via Local Partitioning

Problem definition:

Input: A graph $G = (V, E)$ on n vertices. We shall assume G is planar, has maximum degree $\leq d$, and is represented using adjacency list.

Definition: A vertex-cover is a susbet $V' \subset V$ that is incident to every edge.

Goal: Estimate $\mathrm{VC}(G)$ = the minimum size of a vertex-cover of G.

Theorem 3 [Hassidim, Kelner, Nguyen and Onak, 2009]: There is a randomized algorithm that, given a planar graph G with maximum degree $\leq d$ and $\varepsilon > 0$, estimates (whp) $\mathrm{VC}(G)$ within additive εn and runs in time $T(\varepsilon, d)$ (independent of n).

Main idea: Fix "implicitly" some near-optimal solution. Then estimate it's size by checking for $s = O(1/\varepsilon^2)$ random vertices whether they belong to that solution.

Initial analysis: Let SOL be the implicit solution computed by the algorithm, let X_i for $i = 1, \ldots, s = O(1/\varepsilon^2)$ be an indicator for whether the i-th vertex chosen belongs to SOL. The algorithm outputs $\frac{n}{s} \sum_i X_i$. We will need to prove:

$$|\mathrm{SOL} - \mathrm{VC}(G)| \leq \varepsilon n$$
$$\Pr[|\tfrac{n}{s} \textstyle\sum_i X_i - \mathrm{SOL}| \leq \varepsilon n] \geq 0.9$$

The last inequality follows immediately from Chebychev's inequality, since each $X_i = 1$ independently with probability SOL/n.

Planar Separator Theorem [Lipton and Tarjan, 1979]: In every planar graph $G = (V, E)$ there is a set S of $O(\sqrt{|V|})$ vertices such that in $G \setminus S$, every connected component has size at most $n/2$.

Remark: Extends to excluded-minor families.

Definition: We represent a partition of the graph vertices as $P : V \to 2^V$. It is called an (ε, k)-*partition* if every part $P(v)$ has size at most k, and at most $\varepsilon|V|$ edges go across between different parts.

Corollary 4: For every $\varepsilon, d > 0$ there is $k^* = k^*(\varepsilon, d)$ such that every planar G with max-degree $\leq d$ admits an (ε, k^*)-partition.

Exer: Prove this corollary.

Hint: Use the planar separator theorem recursively.

Our sublinear algorithm will not compute this partition directly, and instead will use local computation to compute another partition (with somewhat worse parameters).

Proof Sketch of Theorem 3: Given an (ε, k)-partition P of G, we define the solution SOL by taking some optimal solution in each part of P, and adding one endpoint for each cross-edge. Clearly, $\mathrm{VC}(G) \leq \mathrm{SOL} \leq \mathrm{VC}(G) + \varepsilon n$.

Thus, the main challenge is to implement a partition oracle, i.e., an "algorithm" that can compute

$P(v)$ for a queried vertex $v \in V$ in constant time. Note: P could be random, but should be "globally consistent" for (and independent of) the different queries v.

Algorithm Partition (used later as oracle):

Remark: It uses parameters k, ε' that will be set later (in the proof)

1. $P = \emptyset$
2. Iterative over the vertices in a random order $\pi_1, \ldots, \pi_n$
3. if π_i is still in the graph then
4. if current graph has a (k, ε')-isolated neighborhood of π_i
5. then $S =$ this neighborhood
6. else $S = \{\pi_i\}$
7. Add $\{S\}$ to P and remove S from the graph.

Definition: A (k, ε')-*isolated neighborhood* of $v \in V$ is a set $S \subset V$ that contains v and has size $|S| \le k$, such that the subgraph induced on S is connected, and the number of edges leaving S is $\mathrm{e_{out}}(S) \le \varepsilon' |S|$.

Lemma 3a: Fix $\varepsilon' > 0$. Then the probability that a random vertex in G does not have a $(k^*(\varepsilon'^2/2), \varepsilon')$-isolated neighborhood is at most ε'.

Proof of Lemma 3a: G admits an $(\varepsilon', k^*(\varepsilon', d))$-partition. Therefore, one can remove from it a set E' of $\le (\varepsilon'^2/2)|V|$ edges, such that in the resulting graph, every connected component has size $\le k^*(\varepsilon'^2/2, d)$. Denote the achieved partition by P. Then

$$\mathop{\mathbb{E}}_{v \in V}\left[\tfrac{\mathrm{e_{out}}(P(v))}{|P(v)|}\right] = \sum_{S \in P} \sum_{v \in S} \tfrac{1}{|V'|} \cdot \tfrac{\mathrm{e_{out}}(S)}{|S|} = \sum_{S \in P} \tfrac{|S|}{|V|} \cdot \tfrac{\mathrm{e_{out}}(S)}{|S|} = \tfrac{2|E'|}{|V|} \le \varepsilon'^2.$$

By Markov's inequality, a random vertex $v \in V'$ satisfies with probability $1 - \varepsilon'$ that $\frac{\mathrm{e_{out}}(P(v))}{|P(v)|} \le \varepsilon'$, in which case it has a $(k^*(\varepsilon'^2/2, d), \varepsilon')$-isolated neighborhood.

Lemma 3b: Fix $\varepsilon > 0$. Let $\varepsilon' = \varepsilon/(16d)$ and $k = k^*(\varepsilon'^2/2, d)$. The above Partition algorithm (oracle) computes whp an (ε, k)-partition. Moreover, if the oracle is asked q non-adaptive queries, then whp its query complexity into G (and also its runtime) is at most $q \cdot 2^{d^{O(k)}}$.

Proof of Lemma 3b: Every part is of size at most k by construction. Let X_i for $i = 1, \ldots, n$ be a random variable corresponding to π_i, the vertex considered in iteration i, as follows. Denote by S_i the set $S \in P$ that contains π_i (it is removed from the graph in iteration i or earlier) and define $X_i = \mathrm{e_{out}}'(S_i)/|S_i|$, where $\mathrm{e_{out}}'(S_i)$ is the number of edges at the time of removing S_i. Notice that each $S \in P$ "sets" $|S|$ variables X_i to the same value, thus $\sum_i X_i = \sum_{S \in P} \mathrm{e_{out}}'(S)$ is the number of cross-edges in P (each edge is counted once, because the graph changes with the iterations).

Fix i. Then π_i is a random vertex, and by Lemma 3a, with probability $\ge 1 - \varepsilon'$ it has a (k, ε')-isolated neighborhood in G (and thus also in every subgraph of G), which implies that $X_i \le \varepsilon'$ (both if π_i is removed in iteration i and if in an earlier iteration). With the remaining probability $\le \varepsilon'$, we can use $X_i \le d$ which always holds. Altogether,

$$\mathbb{E}[X_i] \le 1 \cdot \varepsilon' + \varepsilon' \cdot d \le 2\varepsilon' d.$$

$$\mathbb{E}[\sum_i X_i] \le 2\varepsilon' dn.$$

By Markov's inequality, with probability $\geq 7/8$, the number of cross-edges in P is at most $8(2\varepsilon' dn) = \varepsilon n$.

Local simulation: We generate the permutation on the fly by assigning each vertex v a random number $r(v) \in [0,1]$ (and remember previously used values). Before computing $P(v)$, we first compute (recursively) $P(w)$ for all vertices w within distance at most $2k$ from v that satisfy $r(w) < r(v)$. If $v \in P(w)$ for one of them, then $P(v) = P(w)$. Otherwise, we search for a (k, ε')-isolated neighborhood of v, keeping in mind that vertices in any $P(w)$ as above are no longer in the graph. The search for an optimal vertex cover in a part is done exhaustively.

Complexity: We effectively work in an auxiliary graph H, where we connect two vertices if their distance in G is at most $2k$. Thus, the maximum degree in H is at most $D = d^{2k}$. As seen earlier, this means the expected number of vertices inspected recursively is at most $D^{O(D)} = 2^{D^{O}(1)} = 2^{d^{O(k)}}$.