Sublinear Time and Space Algorithms 2018B – Lecture 12
Sublinear-Time Algorithms for Sparse Graphs*

Robert Krauthgamer

1 Maximum Matching

We completed the proof from last class, see updated notes for the previous lecture.

2 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 $\text{VC}(G)$ = the minimum size of a vertex-cover of G.

Theorem 1 [Hassidim, Kelner, Nguyen and Onak, 2009]: There is a randomized algorithm that, given $\varepsilon > 0$ and a planar graph G with maximum degree $\leq d$, estimates whp $\text{VC}(G)$ 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 sampling $s = O(1/\varepsilon^2)$ random vertices and checking 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:

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

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

*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.

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 3: 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. What k^* do you get?

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, $\text{VC}(G) \leq \text{SOL} \leq \text{VC}(G) + \varepsilon n$.

The remaining (and 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 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. iterate over the vertices in a random order $\pi_1, \ldots, \pi_n$
3.   if π_i is still in the graph then
4.   if π_i has a (k, ε')-isolated neighborhood in the current graph
5.   then $S =$ this neighborhood
6.   else $S = \{\pi_i\}$
7.   add $\{S\}$ to P and remove S from the graph
8. output P

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

Lemma 1a: Fix $\varepsilon' > 0$. With probability at least $1 - 2\varepsilon'$, a random vertex in G has a $(k^*(\varepsilon'^2, d), \varepsilon')$-isolated neighborhood.

Proof of Lemma 1a: Was seen in class, by considering the $(\varepsilon'^2, k^*(\varepsilon'^2, d))$-partition guaranteed by Corollary 3.

WE STOPPED HERE IN CLASS. In case we do not continue, below is the rest of the proof.

Lemma 1b: For every $\varepsilon > 0$, Algorithm Partition above with parameters $\varepsilon' = \varepsilon/(12d)$ and

$k = k^*(\varepsilon'^2, d)$ computes whp an (ε, k)-partition. Moreover, it can be implemented as a partition oracle (given a query vertex, it returns the part of that vertex), whose running time (and query complexity into G) to answer q non-adaptive queries is whp at most $q \cdot 2^{d^{O(k)}}$.

Proof of Lemma 1b: By construction, the output P is a partition, where every part has size at most k. To analyze the number of cross-edges in P, we define for each $i = 1, \ldots, n$ two random variables related to π_i, as follows. Let $S_i = P(\pi_i)$, i.e. the set $S \in P$ that contains π_i (note 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).

Now fix i. Since π_i is a random vertex, by Lemma 1a, with probability $\geq 1 - 2\varepsilon'$, it has a (k, ε')-isolated neighborhood in G, and thus also in every subgraph of G, in which case $X_i \leq \varepsilon'$ (both if π_i is removed in iteration i and if in an earlier iteration). With the remaining probability $\leq 2\varepsilon'$, we can bound $X_i \leq d$ which always holds. Altogether,

$$\mathbb{E}[X_i] \leq 1 \cdot \varepsilon' + 2\varepsilon' \cdot d \leq 3\varepsilon' d.$$

$$\mathbb{E}[\sum_i X_i] \leq 3\varepsilon' dn.$$

By Markov's inequality, with probability $\geq 3/4$, the number of cross-edges in P is at most $4(3\varepsilon' dn) = \varepsilon n$.

Implementation as an oracle: We generate the permutation π on the fly by assigning each vertex v a priority $r(v) \in [0, 1]$ (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, search (by brute-force) 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. Searching for an optimal vertex cover inside a part is done exhaustively.

Running time: 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 $D^{O(D)} = 2^{D^{O(1)}} = 2^{d^{O(k)}}$.