Sublinear Time and Space Algorithms 2026A – Lecture 12
Sublinear-Time Algorithms for Vertex Cover in Planar Graphs*

Robert Krauthgamer

1 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 lists.

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

Goal: Estimate $\mathrm{VC}(G)$ = 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 $\mathrm{VC}(G)$ 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 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 chosen vertex 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.

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.

Theorem 2: For every $\varepsilon, d > 0$ there is a polynomial $k^* = k^*(\varepsilon, d)$ such that every planar G with max-degree $\leq d$ admits an (ε, k^*)-partition.

*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.

It is proved using the famous Planar Separator Theorem (which we will not prove).

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 $|V|/2$.

Remark: It extends to excluded-minor families.

Exer: Prove Theorem 3 by using the planar separator theorem recursively. What k^* do you get?

Our sublinear algorithm will not compute this partition directly (even though the Planar Separator Theorem is algorithmic), and instead it will use a "local" algorithm to compute another partition (with somewhat worse parameters).

Proof Plan for Theorem 1: 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. The following lemma is immediate.

Lemma 1a: $\mathrm{VC}(G) \leq \mathrm{SOL} \leq \mathrm{VC}(G) + \varepsilon n$.

Proof: Since $\mathrm{VC}(\cdot)$ is monotone in adding edges,

$$\mathrm{SOL} \leq \mathrm{VC}(G \setminus \mathrm{cross}(P)) + \epsilon n \leq \mathrm{VC}(G) + \epsilon n.$$

The remaining (and main) challenge is to design an algorithm that can compute $P(v)$ for a queried vertex $v \in V$ in constant time. This is called a *partition oracle*.

Note: P could be random, but should be "globally consistent" for the different queries v.

Definition: An (ε', k')-*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 $\mathrm{e}_{\mathrm{out}}(S) \leq \varepsilon'|S|$.

The advantage of this notion is that it can be verified locally, in contrast to (ε, k)-partition which is global (depends on entire graph).

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

Proof of Lemma 1b: Was seen in class, by considering an $(\varepsilon'^2, k^*(\varepsilon'^2, d))$-partition that is guaranteed to exist by Theorem 2.

Algorithm Partition: We now present a sequential algorithm to construct a partition of G; later we will discuss how to implement it locally (oracle) in sublinear time. It uses parameters ε', k' that will be set later.

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 an (ε', k')-isolated neighborhood in the current graph
5. then $S =$ this neighborhood
6. else $S = \{\pi_i\}$

7. update $P \leftarrow P \cup \{S\}$ and remove S from the graph
8. output P

Lemma 1c: 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 containing 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 1c: By construction, the output P is a partition, and every part in it has size at most k.

Analyzing the number of cross-edges in P: for each $i = 1, \ldots, n$ define 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 a nontrivial iteration create one part $S \in P$, and thus "sets" $|S|$ variables X_i all 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 an (ε', k)-isolated neighborhood in the input G, and then $X_i \leq \varepsilon'$ (this is certainly true if π_i was removed at an earlier iteration). This argument applies also to the current graph, because a subgraph of G is still planar with degrees bounded by d (formally, we need to condition on the first $i-1$ iterations). With the remaining probability $\leq 2\varepsilon'$, we 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 $r(w) < r(v)$. (Note that a vertex w at distance $2k-2$ might affect v by causing removal of a vertex mid-way between v and w.) If $v \in P(w)$ for one of them, then $P(v) = P(w)$. Otherwise, search by brute-force for a (ε', k)-isolated neighborhood of v, keeping in mind that vertices in any $P(w)$ as above are no longer in the graph.

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

QED

Concluding Theorem 1: Lemma 1c designs a partition oracle, and thus completes (the proof of) the algorithm for vertex cover.