6.5240 Sublinear Time Algorithms October 3, 2022

Lecture 7

Lecturer: Ronitt Rubinfeld *Scribe: Lily Chung*

Today we will finish planarity testing, and talk a bit about testing properties of dense graphs. As usual we consider a graph $G = (V, E)$ with maximum degree d, number of vertices $|V| = n$, and number of edges $|E| = m$.

1 Planarity Testing

The algorithm presented in this lecture is from [2]. Note that while we will discuss planar graphs for brevity, these results also apply to other classes of graphs, such as any class of graphs with a forbidden minor. Recall the following definitions and results from last lecture:

Definition 1 *A graph G is (ϵ, k)-**hyperfinite** if it is possible to remove at most ϵn edges from G to produce a graph where every connected component has size at most k.*

Theorem 2 (Follows from Proposition 4.1 in [1]) *For all ϵ and δ, there exists a constant c so that every planar graph of degree at most d is $(\epsilon d, c/\epsilon^2)$-hyperfinite.*

This definition says that it is possible to break up planar graphs, but does not say how to do so. We need a partition oracle to tell us how to do the partitioning.

Definition 3 *A **partition oracle** is a function which, given a vertex v and a parameter ϵ, outputs the name of a part $P[v] \subseteq V$ such that $|P[v]| \leq k$ and $P[v]$ is connected. Furthermore, if G is planar, then with probability at least $9/10$, the number of edges connecting different parts*

$$|\{(u, v) \in E \mid P(u) \neq P(v)\}|$$

is at most $\epsilon dn/4$.

Note that a partition oracle works on all graphs, but the bound on the number of crossing edges holds only for planar graphs. Last lecture we showed that given a partition oracle, we can test if a graph is planar or ϵ-far from planar. The idea was to sample random parts and count the number of edges between them. There was some difficulty because of the fact that it wasn't possible to sample random parts, but only to sample random vertices.

In this lecture we will show how to implement a partition oracle.

1.1 Implementing a Partition Oracle

We will first describe the partitioning process as a global algorithm, and then show how it can be simulated locally. We start with a definition:

Definition 4 *An induced subgraph S of G is a (k, δ)-**isolated neighborhood** of a vertex v if*

- *$v \in S$*
- *S is connected*
- *$|S| \leq k$*
- *The number of edges between S and $V \setminus S$ is at most $\delta|S|$.*

Algorithm 1 Global Partitioner

```
1: $\pi_1 \dots \pi_n \leftarrow$ a random permutation on $V$
2: $P \leftarrow \emptyset$
3: for $i$ from 1 to $n$ do
4:     if $\pi_i$ is still in the graph then
5:         if $\pi_i$ has a $(k, \delta)$-isolated neighborhood then
6:             $S_i \leftarrow$ this neighborhood
7:         else
8:             $S_i \leftarrow \{\pi_i\}$                                    ▷ Bad case
9:         end if
10:        $P \leftarrow P \cup \{S_i\}$
11:        Remove $S_i$ and all incident edges from the graph.
12:    end if
13: end for
14: return $P$
```

The idea of the partitioning process is to carve away a single isolated neighborhood at a time from the graph, removing it. After each removal the graph remains planar. Each isolated neighborhood S_i will become a part of the partition and the total number of edges between different parts will be at most $\sum_i \delta |S_i| = \delta n$, which is what we want. The problem is that not every node has an isolated neighborhood, but luckily only a small number do not. The global partitioning algorithm based on this intuition is given in Algorithm 1.

It is clear from definitions that this algorithm satisfies the first two properties of a partition oracle. We must now show that if G is planar, then P has a small number of crossing edges.

It follows from similar reasoning to before that the contribution of crossing edges from the cases where π_i has a (k, δ)-isolated neighborhood is at most δn. The contribution from the other cases is at most dr, where r is the number of nodes which do not have (k, δ)-isolated neighborhoods. Thus we wish to bound r.

Lemma 5 *Let $G' = (V', E')$ be a subgraph of a planar graph G with $|V'| \geq \delta n$. Then the number of vertices in V' which do not have (k, δ)-isolated neighborhoods is at most $\epsilon|V'|/30$, where $\delta = \epsilon/30$ and $k = \Theta(1/\epsilon^3)$.*

Note that the parameters δ and k have no dependence on n. We won't prove this lemma exactly but we will give the idea:

Proof Idea Since G' is planar, it is hyperfinite. Thus there exists a partition such that the overall number of crossing edges is small. Therefore most of the parts must in fact be isolated neighborhoods by Markov's inequality. ■

This lemma gives us the bound on r necessary to show that the above algorithm is a partition oracle. We will elide the precise details and move on to showing that this global algorithm can be simulated locally.

1.2 Local Simulation of Global Partitioner

The local simulation is given in Algorithm 2. We assume we are given an oracle for π, a function which randomly assigns a rank to each vertex of the graph. Note that we need to query the ball of size $2k$ instead of k because on line 11 we need to ignore vertices which already belong to other parts.

Since we can make further queries during the recursive call, we must bound the cost of the recursion. A similar analysis (which we omit here) to that in the lecture on maximal matching shows that the

Algorithm 2 Local Partitioner

```
1:  On input $v$:
2:  Query the ball $B_{2k}(v)$ of radius $2k$ around $v$.
3:  for $w \in B_{2k}(v)$ do
4:      if $\pi(w) < \pi(v)$ then
5:          Recursively compute P[w]        ▷ This recursion may induce further queries outside $B_{2k}(v)$.
6:      end if
7:  end for
8:  if there exists $w$ with $\pi(w) < \pi(v)$ and $v \in P[w]$ then
9:      return $P[v] = P[w]$
10: else
11:     Use brute force within $B_k(v)$ to find a $(k, \delta)$-isolated neighborhood of $v$, ignoring all vertices
    which already belong to some $P[w]$ with $\pi(w) < \pi(v)$.
12:     if such an isolated neighborhood exists then
13:         return $P[v] =$ the isolated neighborhood
14:     else
15:         return $P[v] = \{v\}$
16:     end if
17: end if
```

overall cost is $2^{d^{O(k)}}$ queries for $k = \tilde{O}(1/\epsilon^3)$. Note that the query complexity does not depend on n. It is known that this query complexity can be improved to $d^{O((\log 1/\epsilon)^2)}$.

2 Property Testing in Dense Graphs

We now change subjects and consider the problem of testing properties in dense graphs, where the graph is given in the adjacency-matrix representation and we are not given a bound on the maximum degree. Recall that the adjacency matrix A of a graph $G = (V, E)$ is an $n \times n$ matrix whose rows and columns are indexed by the vertices V. It is defined such that A_{uv} is 1 if $(u, v) \in E$ and 0 otherwise. We assume we are given query access to the cells of A, but we do not have degree queries or next-neighbor queries.

In dense graphs, we define G to be **ϵ-far** from a property P if greater than ϵn^2 entries of A would need to change in order to make G into a graph with property P. A perhaps surprising consequence of this definition is that, for instance, all sufficiently-large graphs are ϵ-close to connected.

In the next lecture, we will show how to distinguish bipartite graphs from graphs which are ϵ-far from bipartite. Recall that a graph is **bipartite** if there exists a coloring of its vertices with two colors such that no edge is monochromatic. Equivalently, a bipartite graph is one whose vertices can be partitioned into two parts V_1 and V_2 such that no edge has both its endpoints in a single part. Given a (possibly improper) coloring of a graph, we say that a monochromatic edge is a **violating edge**. Thus a graph is ϵ-far from bipartite if and only if any 2-coloring of the graph has more than ϵn^2 violating edges.

The main idea for testing bipartiteness will be to pick a random sample of nodes, and check if the induced subgraph on these nodes is bipartite. If it is not, then the original graph cannot be bipartite.

References

[1] Noga Alon, Paul Seymour, and Robin Thomas. "A Separator Theorem for Graphs with an Excluded Minor and its Applications". In: *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*. 1990, pp. 293–299.

[2] Avinatan Hassidim et al. “Local Graph Partitions for Approximation and Testing”. In: Oct. 2009, pp. 22–31. DOI: 10.1109/FOCS.2009.77.