6.5240 Sublinear Time Algorithms September, 2022

Lecture 6

Lecturer: Ronitt Rubinfeld *Scribe: Farrell Eldrian Wu*

Contents

In this lecture, we discuss a sublinear algorithm for planarity testing, extendable to other minor-closed properties, through reduction to the hyperfiniteness property and the use of a partition oracle. We primarily follow Section 3 of the paper "Local Graph Partitions for Approximation and Testing" by Hassidim, Kelner, Nguyen, and Onak (2009) [3], while the implementation of the partition oracle, covered in the following lecture, is in Section 4 of the same paper.

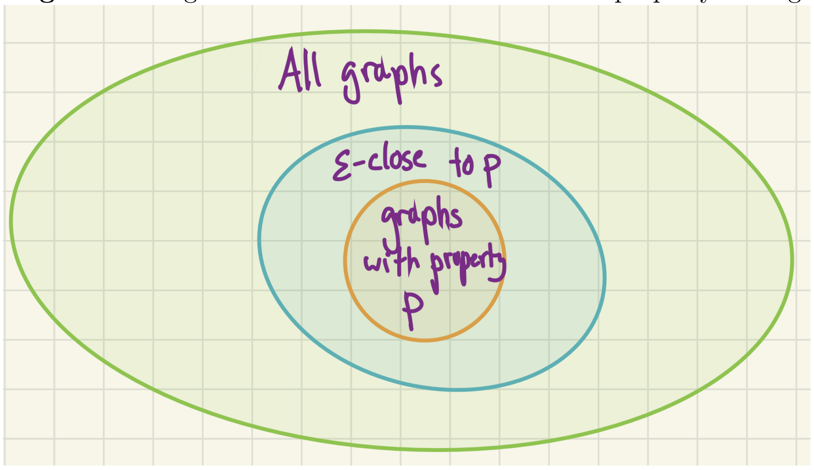
1 Introduction to Property Testing and Planarity

In this section, we formalize the paradigm of testing a property of a graph in sublinear time. We base our discussion around an algorithm for testing planarity. While the focus of this lecture is on graphs, in particular, bounded degree graphs, the concept of property testing and the proof methods used can be applied to many other objects, such as strings and functions.

1.1 Basics of Property Testing

Previously, our focus was on *estimation* problems, which involve approximating some parameter or quantity of graphs, such as the number of edges or connected components. We did so in a framework that involves estimation bounds, probability guarantees, and a parameterized measure of distance to the desired property.

Figure 1: Diagram of ϵ-closeness in the context of property testing



Now, we consider *decision* problems, where we attempt to assess whether some property of a given object holds or not, for example, whether a graph is planar. In our terminology, the algorithm either *accepts* or *rejects* the input. As these problems are "0 or 1", we need to define a coherent notion of a "grey area" for similar principles in constructing approximation algorithms to work. For example, we have to define what it means to say a graph is "approximately planar". The framework we define is called *property testing*.

Consider the three ovals in Figure 1. The innermost circle is the set of graphs with a particular property P that we desire. A completely precise test as to whether a graph has property P or not can likely take linear time, as there can be a single vertex or edge that switches the status of the graph, and finding this single tipping point will take $O(n)$ time.

If we wish to have a sublinear-time algorithm, we therefore need a weaker version of the condition. Property testing is a framework that involves such a weakening of a yes/no question, by focusing on distinguishing graphs that have property P from graphs that are "far enough" from having this property, where the definition of "far enough" depends on the context. The set of graphs in between, which neither have property P or are far enough from having the property is called an ϵ-close set, graphs in which we do not guarantee anything about the output of the algorithm.

Definition 1 (Property Testing) *A property testing algorithm for a property P with parameter ϵ is a sublinear time algorithm that satisfies the following conditions:*

- *If G has property P, then accept (pass).*
- *If G is ϵ-far from having property P, then reject (fail).*
- *If G is ϵ-close, then can either accept or reject (no guarantee of behavior).*

In this lecture, we focus on graphs with maximum degree d represented in adjacency list format, for which the definition of ϵ-close is given below. A graph is ϵ-far if it is not ϵ-close.

Definition 2 (ϵ-closeness for bounded degree graphs) *A graph G with maximum degree d is ϵ-close to having a property P if it is possible to remove at most ϵdn edges from G to produce some graph G' having property P.*

1.2 Planarity Definitions and Characterizations

We first review the definition of a planar graph.

Definition 3 (Planar Graph) *A graph is planar if it can be drawn on the plane such that its edges intersect only at their endpoints.*

Two well-known graphs that are not planar are the clique on 5 vertices (denoted as K_5) and the complete bipartite graphs with 3 vertices on each side (denoted as $K_{3,3}$.) In general, for positive integers m and n, we let K_n be the clique on n nodes and $K_{m,n}$ be the complete bipartite graph with m nodes on one side and n nodes on the other.

A characterization of planarity that allows for easier algorithmic handling is given by the theorem below that relates planarity to forbidden minors, defined below.

Definition 4 (Graph Minor) *A minor of a graph G is a graph constructed by recursively merging edges into vertices, followed by deleting vertices and/or edges. Each merge procedure constructs a supernode that retains the union of connections from the vertices that comprise it.*

Theorem 5 (Kuratowski, Planarity $\iff$ Forbidden K_5 and $K_{3,3}$) *A graph G is planar if and only if it does not contain K_5 or $K_{3,3}$ as a minor.*

For our purposes, such reductions are less useful because we will only use exact tests for constant-sized problems, which will not affect the overall sublinearity of our algorithm. It may, however, be useful to view planarity as a special case of forbidden minors, where specifically, K_5 and $K_{3,3}$ are forbidden. There are other classes of graphs with other forbidden minors, such as disallowing a 4-cycle of vertices.

Therefore, throughout this lecture, keep in mind that the methods we discuss for planarity testing, especially hyperfiniteness and the construction of a partition oracle, are also applicable to other forbidden minor properties.

2 Hyperfiniteness and the Partition Oracle

In this section, we introduce hyperfiniteness, a parameterized property that follows from planarity. More generally, a version of this property, with appropriate parameters, holds for any graph belonging in a particular class of graphs characterized by forbidden minors.[1] This property then motivates the design of the partition oracle, a tool where the resulting partitions are used to decompose the graph into constant-sized smaller graphs. The partition oracle facilitates sublinear testing of planarity.

2.1 The Hyperfiniteness Property

The hyperfiniteness property formally motivates the intuition that it is easy to "break apart" a planar graph by removing not too many of its edges. The formulation of such partitions is a common tool to construct sublinear time algorithms to test for forbidden minor classes of graphs.

Definition 6 (Hyperfiniteness) *A graph G with n vertices is (ϵ, k)-hyperfinite if it is possible to remove at most ϵn edges to result in a graph where all connected components are of size at most k. Here, k can be a function of ϵ.*

Next, the theorem below formally relates planarity and hyperfiniteness. The proof is beyond our scope.[2]

Theorem 7 (Planarity $\rightarrow$ Hyperfiniteness) *For all ϵ and d, there exists a constant c such that every planar graph of maximum degree at most d is $(\epsilon d, c/\epsilon^2)$-hyperfinite.*

Intuitively, the theorem states that removing an ϵ fraction of the edges of a planar graph G, we can break down the graph into connected components of size at most c/ϵ^2. In such graphs, we can get *small* connected components by only removing a *small* fraction of all edges in G. Furthermore, the contrapositive of the theorem implies that if there is no such partition, then G is not planar.

2.2 Consequences of Hyperfiniteness

We now examine consequences of hyperfiniteness. Suppose we apply the definition of hyperfiniteness with parameter $\frac{\epsilon d}{2}$, where ϵ is the parameter in the original testing problem. Denote G' to be the graph after the partition, i.e. G minus the edges between partitions, which number at most $\epsilon dn/2$. This is illustrated in Figure 2.

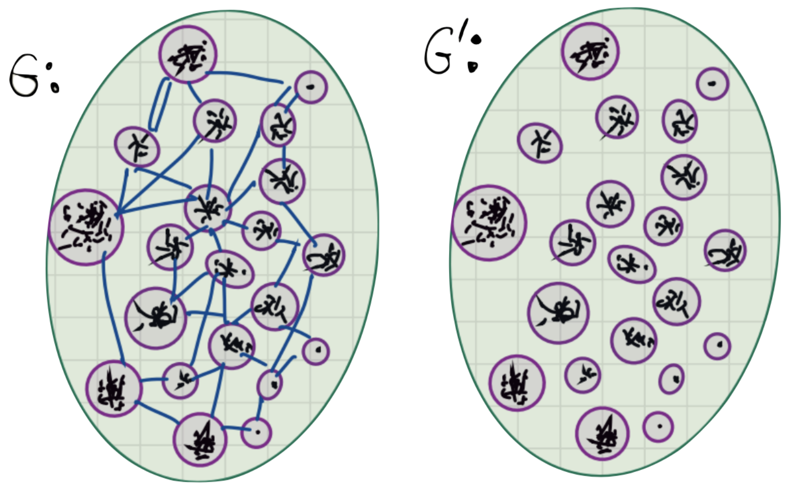
Returning to planarity, we note the following two relationships between G and G'. First, if G is planar, then G' is also planar, as merely removing edges from a planar graph keeps it planar. Second, G' is very similar to G, as they differ by at most $\epsilon dn/2$ edges, as the only difference is that G' drops the edges that cross partitions. Hence, if G is ϵ-far from planar, then G' is $\epsilon/2$-far from planar.

Looking ahead towards testing, in a graph that is ϵ-far from being planar, at least ϵdn edges must be removed to make it planar. We removed at most $\epsilon dn/2$ edges, so the resulting graph still needs at

[1]The converse, however, does not hold. A hyperfinite graph does not necessarily forbid a minor.

[2]If you're interested in the proof, see Lemma 2 and Corollary 3 in [3], noting that planarity is a special case of minor exclusion. A more in-depth discussion about such "separating" theorems for excluded minors can be found in [1].

Figure 2: Depiction of partitions and crossing edges in G and G'



least $\epsilon dn/2$ edges removed to make it planar. Therefore, a lot of the small components, after the cross-partition edges are removed, are non-planar. As all connected components in G' are of size $O(1/\epsilon^2)$, we examining a single component and testing it for planarity can be completed in a time complexity independent of n.

2.3 Partition Oracle for Hyperfiniteness

We are now ready to define the partition oracle, which allows us to query the partition assignment of each vertex. The implementation details of this partition oracle will be covered in the next lecture.

Definition 8 (Partition Oracle (parametrized by (ϵ, k))) *A partition oracle is a function P that takes a vertex v as input and returns $P[v]$, the label of v's partition. The partition must satisfy the following conditions:*

- *(Partitions small and connected) For all vertices $v \in V$, its partition is of size at most k, and the induced subgraph from the set of vertices in this partition is connected.* [3]
- *(Planar graph $\rightarrow$ small number of partition-crossing edges) If G is planar, then with probability of at least $9/10$, $|\{(u,v) \in E \mid P[u] \neq P[v]\}| \leq \epsilon dn/4$.*

Note: the oracle does not have to decide "in advance" which partition to use, as long as the outputs are consistent with one another.

We note a couple observations on the behavior of the partition oracle, considering the cases where G is planar or not planar. If G is planar, the second property of the oracle greatly restricts is behavior. On the other hand, if G is ϵ-far from planar, the first property can provide bounds to help us characterize and locate a violation of planarity.

As a side note, the definition of the partition oracle does not imply that there is a unique partition for every planar graph. There can be many partitions that satisfy the given conditions.

[3]There is no requirement, however, for a partition to be a full connected component.

3 Sublinear Algorithm for Testing Planarity

From the definition of testing, we can set a target for our planarity testing algorithm on a graph G to behave as follows:

- If G is planar, then $\Pr[\text{Pass}] \geq \frac{2}{3}$.
- If G is ϵ-far from planar, then $\Pr[\text{Fail}] \geq \frac{2}{3}$.

3.1 Failure modes of a non-planar graph

The algorithm is motivated by considering what happens when the graph G is not planar. We consider separate cases as to whether the partition oracle fails or succeeds.

1. (Simple case: partition oracle fails) This case is simple, as the partition oracle failing automatically implies that the graph is not planar. In this case, the partition oracle returns a partition with too many cross-partition edges, or there is a partition that is too large.
2. (More complex case: partition oracle succeeds) In this case, we have to find another way to test that the graph is not planar. In this case, we look for some small $O(1/\epsilon^2)$-sized connected component that is itself non-planar, the existence of which we show from the properties of the partition oracle.

The first "failure mode" can be seen as a *global* condition that considers a property of the graph as a whole; in contrast, the second can be seen as a *local* property of a particular vertex and its partition.

3.2 Testing Algorithm

We base this procedure off a particular partition given by the partition oracle with parameters $(\epsilon/2, 4c/\epsilon^2)$. The algorithm in this section tests for the above two "failure modes" in succession.

3.2.1 Stage 1: Test that there are not too many crossing edges

In this stage, denote C as $\{(u, v)|P(u) \neq P(v)\}$, the set of partition-crossing edges. We use an **estimate** $\hat{f}$ of $|C|$, such that its additive error is at most $\epsilon dn/8$ with probability of failure δ that is at most $1/10$. From the estimate, if $\hat{f} \geq 3\epsilon dn/8$, then output "not planar" and halt. Otherwise, proceed to Stage 2.

$\hat{f}$ is obtained by sampling $24/\epsilon^2 = O(1/\epsilon^2)$ edges uniformly at random with replacement, taking the sample average, and then multiplying by the number of edges. The additive error corresponds to a proportion $t = (\epsilon dn/8)/(dn/2) = \epsilon/4$ of all edges, so the Hoeffding bound [4] of $2e^{-2nt^2}$ for Bernoulli random variables gives an upper bound of $2e^{-3} < 1/10$. [5].

The query complexity of this stage is $O(1/\epsilon^2)$, as this is the number of oracle queries.

3.2.2 Stage 2: Test random partitions for planarity

In this stage choose a set S of $O(1/\epsilon)$ random vertices. For each vertex $s \in S$, consider the partition $P[s]$ that contains s. If G is planar, according to the properties of the partition oracle, the vertices in $P[s]$ form a planar connected component of size at most $4c/\epsilon^2$. We then test the vertices one by one until a vertex is found that either has a connected component larger than $4c/\epsilon^2$ vertices or is not planar. If a vertex that violates is found, then reject, otherwise, accept.

[4] See this set of lecture notes for a discussion. Take the second equation with $a = 0$ and $b = 1$, the probability of having an additive error greater than $\epsilon dn/8$ is at most $2e^{-3}$.

[5] In implementing this algorithm, we have to define the process of sampling an edge uniformly at random. If we simply sample a vertex uniformly at random and then sample an edge connected to this vertex again uniformly at random, the sampling will be biased towards edges connected to low-degree vertices. Therefore, we perform a rejection sampling, where after we sample a vertex v and an incident edge e, we only accept this choice with probability $\deg(v)/d$. This yields a constant probability of $2/nd$ for each edge to be picked, possibly with some probability of no edge being picked.

This test is implemented by computing each partition's size via Breadth First Search (BFS) going outwards from the vertex and only considering the edges that lead to another vertex in the same partition. Meanwhile, planarity for the induced subgraph corresponding to a particular partition can be tested by any non-approximate linear-time algorithm.

The query complexity of this stage is determined by the number of calls needed for the BFS steps. For each cell, there are $O(1/\epsilon^2)$ vertices in its connected component and at most d edges connected to each vertex, which yields $O(d/\epsilon^2)$ queries per starting vertex. As there are $O(1/\epsilon)$ vertices in the testing set, this gives $O(d/\epsilon^3)$ total calls.

3.3 Analysis and Discussion

The above discussion provides an algorithm with a query complexity of $O(d/\epsilon^2)$, which is sublinear in n. Now, we show that the testing algorithm above satisfies the desired requirements; that it will output accept when G is planar and reject when G is ϵ-far from planar, both with probabilities of at least $2/3$.

3.3.1 Criteria 1: G is planar $\rightarrow$ accept with probability at least $2/3$

In this case, the partition oracle will return a partition with at most $\epsilon dn/4$ partition-crossing edges, which is also $\text{Exp}[\hat{f}]$ as estimator based on a sample mean is unbiased. As $\hat{f}$ has an additive error of at most $\epsilon dn/8$ with probability at least 9/10. Therefore, with probability at least 9/10,

$$\hat{f} \leq \text{Exp}[\hat{f}] + \epsilon dn/8 \leq \epsilon dn/4 + \epsilon dn/8 = 3\epsilon dn/8,$$

and the algorithm proceeds to stage 2. The algorithm will then always pass Stage 2 because the partition oracle will return a partitions each of size at most $k = 4c/\epsilon^2$, all of which induce a planar subgraph.

3.3.2 Criteria 2: G is ϵ-far from planar $\rightarrow$ reject with probability at least $2/3$

This case is more interesting because it is still possible that the partition oracle returns a valid partition. We thus derive different contradictions based on the number of partition-crossing edges.[6]

The simpler case is where $|C| > \epsilon dn/2$, meaning that there are too many paritition-crossing edges for a valid partition. In this case, the sampling procedure will detect it. To prove this, we use the same Hoeffding bound as in the previous case to get the inequality,

$$f \geq \text{Exp}[\hat{f}] - \epsilon dn/8 > \epsilon dn/2 - \epsilon dn/8 = 3\epsilon dn/8$$

with probability at least 9/10, so we will indeed reject with this desired frequency.

The other possibility is that $|C| \leq \epsilon dn/2$, which means that we have a valid paritition without too many crossing edges. Then, we could go back to the ϵ-far definition and note that we still have to remove many edges, besides the partition-crossing edges, to make G planar. More concretely, denote G' to be G with the edges in C removed. Then as G' is $\epsilon/2$ close to G and G is ϵ-far from being planar, G' must still be $\epsilon/2$-far from being planar. This means that at least $\epsilon dn/2$ edges must be removed for planarity.

Consider a minimal set of edges to be removed from G in order to make it planar. Then any edge in this set is part of a non-planar connected component of G'. As this minimal set contains at least $\epsilon dn/2$ edges, there must be at least ϵn vertices adjacent to at least one edge of this set. [7] Then any of these vertices will be in a non-planar connected components. Thus, considering the whole graph, we can detect non-planarity once any one of this ϵ fraction of the vertices is selected, meaning that a set of $O(1/\epsilon)$ suffices. [8]

[6]An astute observer might notice that our work for this criteria involves nearly all of the characteristics discussed in the algorithm, except for the maximum size of a connected component. We might naturally have expected this condition to cause an issue here, but actually, the condition in fact serves to reduce the worst case runtime as it avoids the case of overlapping and time-consuming BFS's from different nodes.

[7]This can be derived from a standard double-counting argument, as each edge is incident to exactly two vertices.

[8]This follows from the inequality $(1-\epsilon)^{a\epsilon} \leq e^{-a}$. If the sampling is done without replacement, then the inequality only becomes stronger. This part is left as an exercise.

4 Further Reading

For those interested in further reading, Sections 2 and 3 from a 2011 overview paper on Sublinear Time Algorithms by Rubinfeld and Shapira [7] provide a more thorough discussion on the frameworks behind complexity testing; Section 5.2 in the same paper reviews the literature and state of research for bounded degree graphs. You can find a recently improved algorithm with an improved partition oracle in [4].

On the foundations and techniques used in this lecture, Background information about graph planarity, minor-closed properties, and forbidden minors can be found in [8]. More discussion on partition oracles can be seen in [5] as well as in Sections 2.5 and 2.6 of [6]. Finally, similar methods for testing minor-closed properties are discussed in [2].

References

[1] N. Alon, P. Seymour, and R. Thomas. "A Separator Theorem for Graphs with an Excluded Minor and Its Applications". In: *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing.* STOC '90. Baltimore, Maryland, USA: Association for Computing Machinery, 1990, pp. 293–299. ISBN: 0897913612. DOI: 10.1145/100216.100254. URL: https://doi.org/10.1145/100216.100254.

[2] Itai Benjamini, Oded Schramm, and Asaf Shapira. "Every Minor-Closed Property of Sparse Graphs is Testable". In: *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing.* STOC '08. Victoria, British Columbia, Canada: Association for Computing Machinery, 2008, pp. 393–402. ISBN: 9781605580470. DOI: 10.1145/1374376.1374433. URL: https://doi.org/10.1145/1374376.1374433.

[3] Avinatan Hassidim et al. "Local Graph Partitions for Approximation and Testing". In: *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA.* IEEE Computer Society, 2009, pp. 22–31. DOI: 10.1109/FOCS.2009.77. URL: https://doi.org/10.1109/FOCS.2009.77.

[4] Reut Levi and Dana Ron. *A Quasi-Polynomial Time Partition Oracle for Graphs with an Excluded Minor.* 2013. DOI: 10.48550/ARXIV.1302.3417. URL: https://arxiv.org/abs/1302.3417.

[5] Huy N. Nguyen and Krzysztof Onak. "Constant-Time Approximation Algorithms via Local Improvements". In: *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science.* FOCS '08. USA: IEEE Computer Society, 2008, pp. 327–336. ISBN: 9780769534367. DOI: 10.1109/FOCS.2008.81. URL: https://doi.org/10.1109/FOCS.2008.81.

[6] Krzysztof Onak. *New sublinear methods in the struggle against classical problems.* 2010. URL: https://dspace.mit.edu/handle/1721.1/62428.

[7] Ronitt Rubinfeld and Asaf Shapira. "Sublinear Time Algorithms". In: *Electron. Colloquium Comput. Complex.* TR11-013 (2011). ECCC: TR11-013. URL: https://eccc.weizmann.ac.il/report/2011/013.

[8] Dan Weiner. *Forbidden Minors and Minor-closed Graph Properties.* 2006. URL: https://math.uchicago.edu/~may/VIGRE/VIGRE2006/PAPERS/Weiner.pdf.