CS762: Graph-Theoretic Algorithms
Lecture 29: Planarity Testing
March 25, 2002

Scribe: Therese Biedl

Abstract

In the previous lecture, we studied the independent set problem in planar graphs and gave an ϵ-approximation scheme for it. We give a different ϵ-approximation scheme in this lecture and compare the two algorithsm. Then we give a (very brief) history of the problem of testing planarity.

1 Introduction

An independent set in a graph is a set of vertices without an edge between them. The independent set problem is the problem of, given a graph, finding the maximum cardinality independent set. As we saw previously, this problem is NP-hard, even if the given graph is planar. In the last class, we saw an ϵ-approximation scheme for this problem, i.e., a class of algorithms that gets arbitrarily close to the optimal answer. This scheme was based on the idea of converting a planar graph into a $(k-1)$-outer-planar graph without deleting too many vertices in k different ways. Since we can find the maximum independent set in a k-outer-planar graph in polynomial time, we can compute a maximum independent set for each of the resulting graphs, and one can show that the best of these independent sets will be good. See the previous lecture and [Bak94] for details of this algorithm.

Also in the last lecture we started another approach for an ϵ-approximation scheme for independent set in planar graphs. This one is based on weighted vertex-separators, and was presented in [LT79, LT80]. Recall that a (weighted) α-separator of a graph is a set of vertices S such that removing them leaves components that each have weight at most $\alpha \cdot w(V)$, where $w(V)$ is the total weight of the graph. The two following results are known:

Theorem 1 *[LT79] Every planar graph has a weighted 2/3-separator of size $\sqrt{8n}$. It can be found in $O(n)$ time.*

Theorem 2 *[LT80] For any $0 < \epsilon < 1$, every planar graph has a weighted ϵ-separator of size $16\sqrt{n/\epsilon}$. It can be found in $O(n \log n)$ time, where the constant inside "O" does not depend on ϵ.*

Using the latter theorem, it is quite easy to develop another ϵ-approximation scheme, which will be covered in Section 2.

In the second half of the lecture, we turned to the planarity testing problem, i.e., given a graph, find out whether it is planar. Clearly this is a very important problem, because none of the previous results for planar graphs can be applied unless we can find out first whether a given graph is planar. Unfortunately, due to lack of time just about no details of these algorithms can be given, but lots of references will be provided.

2 Another approximation algorithm

So we first turn to giving another approximation algorithm for independent set in planar graphs. This was first presented in [LT80].

Let G be an arbitrary planar graph with n vertices, and let $k(n)$ be a function. (We will pick $k(n)$ suitably later.) Assign uniform weights to the vertices (i.e., $w(v) = \frac{1}{n}$). Set $\epsilon = k(n)/n$. Compute an ϵ-separator S for G; we know that there exists such a separator of size $16\sqrt{n/\epsilon} = 16\sqrt{n^2/k(n)} = 16n/\sqrt{k(n)}$. This takes $O(n \log n)$ time.

Now consider the graph obtained by deleting the vertices in S. It has a number of connected components, and each connected component C has weight at most ϵ, thus

$$w(C) = \frac{1}{n}|C| \leq \epsilon = k(n)/n,$$

and therefore $|C| \leq k(n)$. For each such component, compute a maximum independent set with brute force by trying all possible subsets, and returning the largest that is an independent set. This can be done in $O(2^{|C|}|C|) = O(2^{k(n)}|C|)$ time per component C, and therefore takes $O(2^{k(n)}n)$ time for all componeents together, since all componeents together contain at most n vertices.

Let I_C be the maximum independent set of component C, and let I be the union of all these independent sets. Since there is no edge between two components, I is an independent set of the original graph G, and we claim that in fact it is a pretty good independent set.

To see this, let I^* be a maximum independent set of graph G. For each component C, $I^* \cap C$ is an independent set within this component, and therefore $|I^* \cap C| \leq |I_C|$, since I_C was a maximum independent set in the component. In essence therefore, because we did not delete very many vertices in S to obtain the components, I^* cannot be much bigger than I. The precise analysis is obtained through tedious set manipulations as follows:

$$\begin{aligned}
|I^*| &= |\,(I^* \cap S) \cup \bigcup_{C \text{ comp.}} I^* \cap C\,| \\
&= |I^* \cap S| + \sum_{C \text{ comp.}} |I^* \cap C| \quad \text{since components are disjoint} \\
&\leq |S| + \sum_{C \text{ comp.}} |I_C| \\
&\leq 16n/\sqrt{k(n)} + |I|.
\end{aligned}$$

We know that every planar graph is 4-colorable, and therefore the largest color class has size at least $n/4$. Each color class is an independent set, so clearly $|I^*| \geq n/4$. Using this, we obtain:

$$\begin{aligned}
|I| &\geq |I^*| - 16n/\sqrt{k(n)} \\
&\geq |I^*| - |I^*|\frac{4}{n}16n/\sqrt{k(n)} \\
&= |I^*|(1 - 64/\sqrt{k(n)})
\end{aligned}$$

which shows that $|I|$ can not be much smaller than $|I^*|$, as desired.

The precise bound depends on our choice of $k(n)$, and to see what good choices are, we analyze the time complexity. From what was said above, finding I takes $O(n \log + 2^{k(n)}n)$ time. Clearly therefore $k(n)$ should be at most logarithmic, for otherwise this will be exponential. Setting $k(n) = \log\log n$, for example, we obtain an $O(n \log n)$ algorithm to find an independent set of size at least $(1 - 64/\sqrt{\log\log n})$ times the optimum.

Now, how does this compare to Baker's algorithm? In general, not well. Note that for the approximation factor to be at least positive, we must have $64 \geq \sqrt{\log\log n}$, or $n \geq 2^{2^{64^2}}$, a truly gigantic number. On the other hand, Baker's algorithm (applied with $k = \log\log n/6$, for example) gives an algorithm that finds an independent set of size at least $(1 - 6/\log\log n)$ the optimum in $o(n \log n)$ time. This becomes a non-trivial bound for $n = 2^{2^6}$, a much more managable number.

Thus, the approximation algorithm by Lipton and Tarjan is certainly not recommendable for planar graphs. However, there is nevertheless merit in it for the following reason: Note that Baker's algorithm strongly relies on the fact that we have a planar graph, because only then can we obtain k-outer-planar graphs by removing few vertices from the graph. On the other hand, the algorithm by Lipton and Tarjan really does not use planarity, beyond the fact that we have a small ϵ-separator (or as a matter of fact, that we have a small 2/3-separator – Theorem 2 follows immediately from Theorem 1). Thus this technique can be generalized to any graph that is known to have small separators.

3 Planarity Testing

We now turn to an entirely different topic, which is how easy it is to test whether a given graph is planar. We provide only a brief history, and mostly refer to the references given below for details.

- The history of planarity testing starts in 1961 with the paper by Auslander and Parter [AP61]. They were the first to show that planarity testing is in fact polynomial, though the time complexity of this algorithm is $O(n^3)$, hence not particularly fast.
- During the next decade, various attempts were made to decrease the time complexity. One notable one among those is the algorithm by Demoucron, Malgrange and Partuiset [DMP64], which is probably one of the simplest planarity testing algorithms known, but unfortunately its time complexity is still relatively high at $O(n^2)$.
- One algorithm that proves to be very useful later was presented in 1966 by Lempel, Even and Cederbaum [LEC67]. Just to give an idea of how planarity testing can work in general, we give the briefest of sketch of how this algorithm works:
  - Define an st-ordering of a graph to be a vertex ordering $v_1, \ldots, v_n$ such that (v_1, v_n) is an edge, and for every $1 < i < n$, vertex v_i has at least one predecessor and at least one successor.
  - It is easy to show that such an st-ordering exists whenever the graph is biconnected. Therefore, we will only be testing planarity for biconnected graphs. This is not a restriction because a graph is planar if and only if all biconnected components of the graph are planar.
  - Studying how an st-ordering works on a planar graph, one can observe that if we choose the outer-face as one of the faces incident to edge (v_1, v_n), then for every $i > 1$, vertex v_i must be in the outer-face of the graph defined by $v_1, \ldots, v_{i-1}$. In this sense, the st-ordering acts like the canonical ordering for triangulated graphs. (In fact, the canonical ordering is a special case of an st-ordering, but the definition of a canonical requires a planar embedding to be given, whereas the st-ordering does not.

  - The idea is thus to compute an *st*-ordering, and then building up a planar embedding of the graph by adding vertices one-by-one, maintaining all possible embeddings of the graph in a suitable data structure. At each step, we must add the new vertex in the outer-face, which in particular means that all predecessors of it must be consecutive on the outer-face of the graph that we had so far. If we ever fail of achieving that, then the graph was not planar.

- In 1974, a significant breakthrough was achieved in that Hopcroft and Tarjan presented the algorithm to test planarity in linear time [HT74]. However, this algorithm was not entirely satisfactory on various accounts. First, it is relatively hard to understand, and even harder to implement so that its running time actually is linear time. Second, the algoritm only returns whether a graph is planar, but it does not actually find the planar embedding.

- In 1976, two independent results showed together that the algorithm by Lempel, Even and Cederbaum mentioned above actually also can be implemented in $O(n)$ time. First, Even and Tarjan [ET76] showed that an *st*-ordering of a biconnected graph can be found in linear time. Then, Booth and Lueker introduced the PQ-tree [BL76], and showed that among other things, it can be used to test the consecutiveness of predecessors in overall linear time.

  Unfortunately, this still is not an entirely satisfying algorithm. The concept of the algorithm is simple enough, and finding the *st*-orderings in linear time is not difficult either, but the details of the linear-time implementation of PQ-trees are relatively complicated. Also, this algorithm yet again does not yield the planar embedding in linear time (though it can be obtained in $O(n^2)$ time.)

- Nothing happened for a decade. Then Chiba et al. [CNAO85] showed how to find the embedding during the Lempel/Even/Cederbaum algorithm in linear time. Then nothing happened for a decade. Then interest in planarity testing algorithms was revived, probably because people started to build libraries of graph algorithms and data structures, and in particular, packages that did graph drawing, especially for planar graphs. In 1996, Mehlhorn and Mutzel implemented Hopcroft and Tarjan's algorithm as part of the LEDA program package, and in the process, clarified the algorithm and explained how to find the planar embedding in linear time as well [MM96].

- Also in 1996, Di Battista and Tamassia developed an on-line planarity testing algorithm [BT96]. Here, the question is to decide whether the current graph is planar, under a sequence of changes to the graph that may or may not destroy or add planarity.

- In 1999, Boyer and Myrvold presented a paper at SODA [BM99] in which they gave an entirely new approach to planarity testing, based on doing a depth-first search with appropriate bookkeeping. (As a matter of fact, this approach wasn't so entirely new: in apparently independent work, Hsu and Shih [SH92] have developed an algorithm that appears to be the same one, though the details of this paper are not entirely clear).

  The conference paper by Boyer and Myrvold leaves out some details, especially as far as the linear time complexity of implementing the algorithm is concerned, and a journal version does not appear to be in sight yet. So there are still some doubts about this algorithm. In this scribe's opinion, there are no doubts that the algorithm works correctly, and it seems likely that it can be implemented in linear time, though this may be less obvious than the authors make it seem in the paper.

In summary, a truly simple linear-time planarity testing algorithm that could be explained in, say, a lecture at most is still waiting to be found. At this point in time, the best approach to planarity testing may be to either give up on linear time (and implement the algorithm by Demoucron et al.) or to use an exiting program package (for example LEDA) that has a planarity testing algorithm built in.

4 Testing outer-planar and k-outer-planar graphs

With the history of planarity testing now in place, we should also speak about how easy it is to test whether a graph is outer-planar and k-outer-planar.

Testing whether a graph is outer-planar is very easy. Recall that a graph is outer-planar if and only if it has a planar drawing such that all vertices are on the outer-face. Assume for the moment we have such a drawing. Then we can add a new vertex inside the outer-face and connect it to all existing vertices without destroying planarity. We call the new vertex the *universal vertex* because it is connected to all other vertices. So if G is outer-planar, then adding a universal vertex to G gives a planar graph. Now on the other hand, if G plus a universal vertex is planar, then clearly G is outer-planar, simply by declaring the outer-face to be the face that results from deleting the universal vertex.

So in summa, to test whether a given graph G is outer-planar, we add a universal vertex to G and run a planarity test. This takes linear time.

Testing whether a graph is k-outer-planar is more difficult. First, recall that the definition of k-outer-planar states that among all possible planar embeddings, there must be one that has at most k layers. For a given planar embedding and outer-face, this is very easy to test – just compute all layers and count how many there are. But if the planar embedding is not given, then we effectively must try all planar embeddings. This is not as daunting as it may sound, because there exist good data structures to store all planar embeddings, by only storing the places where a planar embedding can be changed. In particular, testing whether a graph is k-outerplanar can be done in $O(k^3n^2)$ time [BM90].

References

[AP61] L. Auslander and S. V. Parter. On imbedding graphs in the sphere. *Journal of Mathematics and Mechanics*, 10(3):517–523, 1961.

[Bak94] B. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the Association for Computing Machinery*, 41(1):153–180, 1994.

[BL76] K. Booth and G. Lueker. Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms. *Journal of Computing and System Sciences*, 13:335–379, 1976.

[BM90] D. Bienstock and C. Monma. On the comlexity of embedding planar graphs to minimize certain distance measures. *Algorithmica*, 5(1):93–109, 1990.

[BM99] J. Boyer and W. Myrvold. Stop minding your P's and Q's: A simplified $O(n)$ planar embedding algorithm. In *SIAM-ACM Symposium on Discrete Algorithms*, pages 140–146, 1999.

[BT96] G. Di Battista and R. Tamassia. On-line planarity testing. *SIAM J. Computing*, 25(5), 1996.

[CNAO85] N. Chiba, T. Nishizeki, S. Abe, and T. Ozawa. A linear algorithm for embedding planar graphs using pq-trees. *J. Comput. System Sci.*, 30:54–76, 1985.

[DMP64] G. Demoucron, Y. Malgrange, and R. Pertuiset. Graphes planaires: reconnaissance et construction de représentations planaires topologiques. *Rev. Francaise Recherche Opérationnelle*, 8:33–47, 1964.

[ET76] S. Even and R. E. Tarjan. Computing an *st*-numbering. *Theoretical Computer Science*, 2:436–441, 1976.

[HT74] J. E. Hopcroft and R. E. Tarjan. Efficient planarity testing. *J. Association Computing Machinery*, 21(4):549–568, 1974.

[LEC67] A. Lempel, S. Even, and I. Cederbaum. An algorithm for planarity testing of graphs. In *Theory of Graphs, International Symposium Rome 1966*, pages 215–232. Gordon and Breach, 1967.

[LT79] R. Lipton and R. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36(2):177–189, 1979.

[LT80] R. Lipton and R. Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980.

[MM96] K. Mehlhorn and P. Mutzel. On the embedding phase of the Hopcroft and Tarjan planarity testing algorithm. *Algorithmica*, 16:233–242, 1996.

[SH92] W.-K. Shih and W.-L. Hsu. A simple test for planar graphs. In *Proceedings of the Sixth Workshop on Discrete Mathematics and Theory of Computation*, pages 35–42, 1992.