CS762: Graph-Theoretic Algorithms
Lecture 25: Problems on planar graphs
March 15, 2002

Scribe: Therese Biedl

Abstract

In previous lectures, we introduced the canonical ordering for triangulated planar graphs and gave some applications. In this lecture, we examine in detail the application for which it was originally introduced, namely, straight-line grid drawings of planar graphs. Then we turn to an entirely different topics, which is problems on planar graphs, and whether they are still NP-hard.

1 Introduction

In previous lectures, we defined the canonical ordering [FPP90], which exists for every planar triangulated graph. The formal definition has a lot of extra conditions, but essentially, we need only the following parts. A canonical ordering is a vertex ordering $\{v_1, \ldots, v_n\}$ such that v_1, v_2 is an edge on the outer-face, and for every $i \geq 3$, vertex v_i is in the outer-face of the graph induced by $v_1, \ldots, v_{i-1}$, and has at least two neighbors in this graph.

When doing something with a canonical ordering, we therefore only have to show two parts: The base case (how to handle the first two vertices, which are an edge (v_1, v_2), or sometimes it is easier to show how to handle the first three vertices, which are a triangle $\{v_1, v_2, v_3\}$), and the step (how to handle adding a new vertex with at least two incident edges in the outer-face.) See Figure 1 for an illustration.

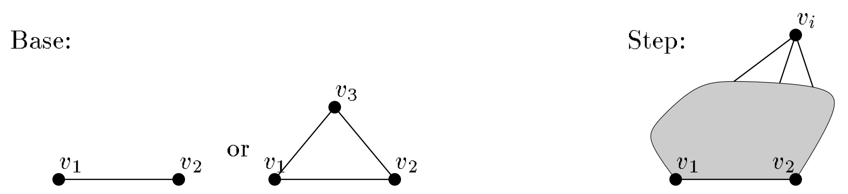

Figure 1: The base case and the step for the canonical ordering.

We saw previously how to use the canonical ordering to show that all planar graphs have arboricity at most 3. We also saw an algorithm that used a canonical ordering to create visibility representations of planar graphs. In this lecture, we will give one last application (and in fact, is the one that inspired the concept of a canonical ordering), which are straight-line grid drawings of planar graphs.

We then turn to an entirely different topic. Recall that there are a number of problems, for example Hamiltonian Cycle, Vertex Cover, Independent Set, Max Cut, Clique, Coloring, etc., which are known to be NP-hard on arbitrary graphs. Are these problems NP-hard even when the input graph is planar? The answer to this depends on the problem. Most problems are indeed still NP-hard. To show this, we define a variant of 3-SAT which operates on (in some sense) a planar graph; this so-called Planar-3-SAT problem can be shown to be NP-hard. From the NP-hardness of this problem follows the NP-hardness of most of the above problems. Exceptions are Maximum Cut and Clique, which both become polynomial time solvable on planar graphs.

2 Straight-line grid drawings of planar graphs

Recall that a planar drawing of a graph is a drawing in the plane such that no two edges cross. A *planar straight-line drawing* is a special case of a planar drawing: all edges must be drawn as straight-line segments between their endpoints. Thus, vertices are assigned to points in the plane, and each edge is represented by the straight-line segment between its endpoints. No two edges should cross. Also, no edge should be going "through" a non-incident vertex. In a more formal description, the interior of the straight-line segment of every edge should not intersect any other element of the drawing.

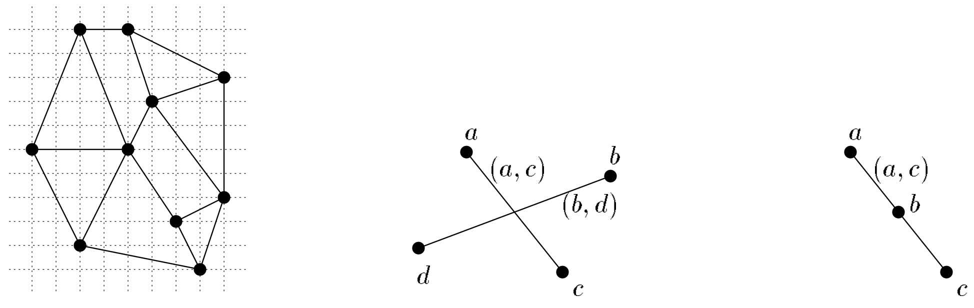

Figure 2: A planar straight-line drawing (in fact, a planar straight-line grid-drawing), and the two forbidden configurations for a planar straight-line drawing.

While by definition every planar graph has a planar drawing, it is not clear at all whether every planar graph has a planar straight-line drawing. However, this indeed is the case, and this result was among the first results in the area of graph drawing. It was proved independently by a number of people, among them Wagner [Wag36], Fáry [Fár48] and Stein [Ste51].

Theorem 1 *Every planar simple graph has a planar straight-line drawing.*

The proof that we give here is none of the above proofs, but instead uses the canonical ordering. It is based loosely on the work by de Fraysseix, Pach and Pollack [FPP90] for *planar straight-line grid-drawings.* In such drawings, we want a planar straight-line drawing such that additionally all vertices have integral coordinates and such that the area of the used grid is small.

So assume that $\{v_1, \ldots, v_n\}$ is a canonical ordering of a planar triangulated graph. (Recall that we can make every planar graph triangulated by adding edges, so if we can show that every triangulated graph has a planar straight-line drawing, then every planar graph has a straight-line drawing as well.) For $i = 3, \ldots, n$, we will build a straight-line drawing of the graph induced by $v_1, \ldots, v_i$ using induction. To be able to add v_i efficiently, we will keep the following invariant:

Invariant 1 *Let G_i be the graph induced by $v_1, \ldots, v_i$, and assume that $v_1 = c_1, c_2, \ldots, c_\ell = v_2$ are the vertices on the outer-face of G_i, in clockwise order. Then G_i has a straight-line drawing such that $x(c_1) < x(c_2) < \ldots < x(c_\ell)$, where $x(v)$ denotes the x-coordinate or vertex v.*

This invariant clearly holds for $i = 3$, see for example the picture in Figure 1. Thus assume that we have such a drawing for G_{i-1}; we will show how to add v_i to it. Assume that vertex v_i has neighbors $c_a, \ldots, c_b$ on the outer-face, with $a < b$. Let $x(v_i)$ be some value with $x(c_a) < x(v_i) < x(c_b)$; if we place v_i with this x-coordinate then the invariant will be satisfied.

We must be a little careful in choosing the y-coordinate for v_i to avoid introducing crossings or overlap. But it is clear that some suitable y-coordinate must exist. For each of $c_a, \ldots, c_b$ (which are the neighbors of v_i) can see upwards towards infinity by the invariant. This means that at infinity, the upward lines from these points intersect the line $x = x(v_i)$, and thus at $y = \infty$, the line from c_i to $(x(v_i), y)$ does not intersect any other element of the drawing. But in fact, since all lines in the drawing have finite slope, already for some finite value of y, the line from c_i to $(x(v_i), y)$ does not intersect any other element of the drawing.[1] Setting $y(v_i)$ to be the maximum over all these restrictions on y implied by the neighbors of v_i will yield a feasible y-coordinate. See Figure 3 for an illustration.

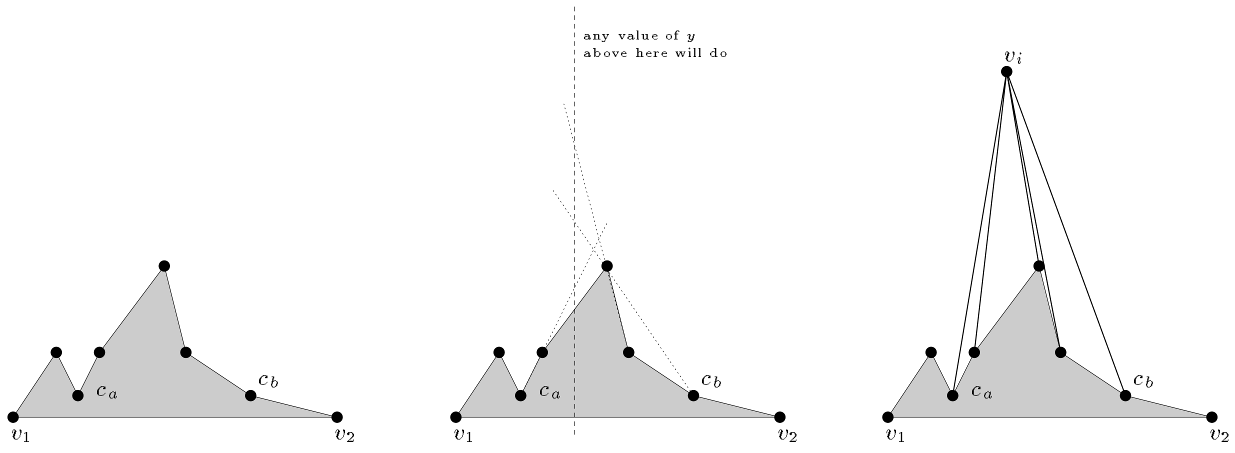

Figure 3: Adding to the straight-line drawing.

Since we can repeatedly add vertices until we have the whole graph, this proves that every planar graph has a straight-line drawing. But in fact, with a slight strengthening of the invariant, we can achieve a grid-drawing. If we demand that all slopes of edges on the outerface are either $+1$ or -1 (with the exception of the edge (v_1, v_2)), then it is easy to see that the edges (c_a, v_i) and (v_i, c_b) only need slopes $+1$ and -1, respectively. However, to avoid overlap, we then must start to modify the drawing of G_{i-1} and "shift" some vertices; see [FPP90] for details. In total, this then leads to a width of $2n - 4$ and a height of $n - 2$; in particular the area of the resulting drawing is $O(n^2)$.

3 NP-hard problems on planar graphs

Now we turn to the other, quite unrelated, topic of this class, which concerns what problems remain NP-hard when restricted to planar graphs. We will first show that 3-coloring (a problem that we showed to be NP-hard in one of the very first lectures) remains NP-hard in planar graphs. Then we talk about Planar 3-SAT and its implications.

[1]The precise value of y that we can use depends on the other coordinates and is of no importance for the general proof; for detailed proofs that also involve bounds on the grid size, we use stronger invariants that allow for specific values of y.

3.1 Coloring planar graphs

Recall that a coloring of a graph G with k colors is an assignment of labels (or colors) in $\{1, \ldots, k\}$ to the vertices such that for every edge the two endpoints have different colors. The k-coloring problem is the problem where we are given a graph and we want to know whether it has a coloring with k colors.

It is quite easy to see that every planar graph has a 6-coloring. First, multiple edges and loops do not change the existence of a coloring, so we can remove those and only study simple planar graphs. Now, every simple planar graph has a vertex of degree at most 5. By computing a minimum-degree order (i.e., a vertex ordering where each vertex has the minimum degree in the graph induced by the remaining vertices), we obtain an ordering such that $\text{outdeg}(v) \leq 5$ for all vertices. By running the greedy algorithm for coloring with the reverse of this ordering, we obtain a coloring with $\max\{\text{outdeg}(v) + 1\} \leq 6$ colors. In fact, this algorithm can easily be implemented in linear time.

It is not too hard to see that in fact 5 colors suffice. Whenever in the above approach we have a vertex that has $\text{outdeg}(v)$, we need to modify the greedy-algorithm slightly to contract two of its non-adjacent neighbors, and we will then be able to use only 5 colors throughout. See [CNS81] for more on this argument, as well as a linear time algorithm to find a 5-coloring of a planar graph.

It was long since conjectured that in fact 4 colors suffice for every planar graphs, see for example Aigner's book [Aig84] for a long history of misguided attempts to prove this result. Only in 1977 was this problem finally solved by Appel and Haken in the affirmative ([AH77] and [AHK77]); an updated and somewhat simplified (though still very complicated) proof was given by Robertson et al. [RSST97]. An algorithm to find such a coloring takes $O(n^2)$ time.

So for planar graphs, the 4-coloring problem is clearly polynomial, because the answer is always "yes". This makes it even more surprising that the 3-coloring problem actually remains NP-hard for planar graphs! So while every planar graph can be colored with 4 colors, deciding whether we can do with one fewer color is NP-hard. (On the other hand, deciding whether we can do with two fewer colors is again polynomial, because that is testing whether the graph is bipartite.)

3.2 3-coloring is NP-hard

So we will now show that 3-coloring is NP-hard, even in planar graphs. To do this, we need a "crossing-gadget", i.e., a small subgraph that will be used to replace a crossing in a drawing of a graph with a planar graph that otherwise does not change the problem. Our gadget is shown in Figure 4. It has four special vertices, labeled a, a', b and b'.

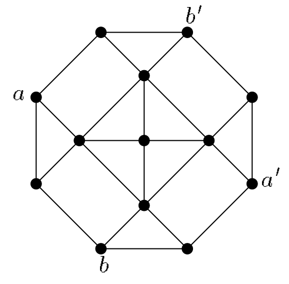

Figure 4: The crossing-gadget for 3-coloring.

Doing sufficiently tedious case-analysis of all possible 3-colorings of this gadget, it is not very hard to show the following results:

- In any 3-coloring of the gadget, $c(a) = c(a')$ and $c(b) = c(b')$, where $c(v)$ denotes the color assigned to vertex v.
- There exists a 3-coloring of the gadget where $c(a) = c(b)$.
- There exists a 3-coloring of the gadget where $c(a) \neq c(b)$.

Now we are ready for the formal proof that 3-coloring is NP-hard for planar graphs. The proof is by reduction from 3-coloring in general graphs. So let G be an instance of general 3-coloring; G is not necessarily planar. Create a simple drawing (with crossings) of G by drawing the vertices on a horizontal line in an arbitrary order, and drawing edges as partial circles.[2]

Now create a new graph G' as follows. Consider the crossing s with the smallest x-coordinates of all crossings. Say the crossing edges are edges (a, c) and (b, d). Since all edges are drawn monotonically increasing in x-direction, and since we picked the leftmost crossing, there can be no more crossings between s and the left endpoint of each of the edges; assume that these endpoints are a and b, respectively. We obtain G' by inserting the crossing gadget at s, identifying the two vertices labelled a, identifying the two vertices labelled b, and changes edges (a, c) and (b, d) to become (a', c) and (b', d) instead. See Figure 5 for an example.

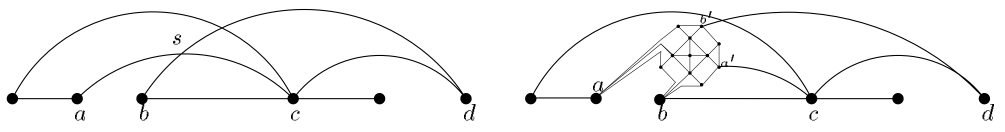

Figure 5: Replacing a crossing by the crossing gadget.

Claim 1 *G is 3-colorable if and only if G' is 3-colorable.*

Proof: Assume first that we have a 3-coloring of G. To obtain a 3-coloring of G', set $c(a') = c(a)$ and $c(b') = c(b)$, and 3-color the rest of the gadget appropriately (we know that this is feasible regardless of whether $c(a) = c(b)$ or not).

For the other direction, assume we have a 3-coloring of G'. Then we must have $c(a) = c(a')$ and $c(b) = c(b')$. By simply taking the same coloring for G, we obtain a 3-coloring for G. For the only edges where there might possibly be a violation of the coloring rules are (a, c) and (b, c), but edges (a', c) had differently colored endpoints in G', and $c(a) = c(a')$, so (a, c) has differently colored endpoints in G. Similarly (b, d) has differently colored endpoints in G, so this is indeed a legal 3-coloring of G. □

Now we are almost done. Note that G' has one fewer crossing that G and all edges that are involved in crossings are still drawn strictly monotone. By repeating the argument again and again, we obtain a graph G^* that is planar and that has a 3-coloring if and only if G has one. Also, since there were $O(n^2)$ crossings, the size of G^* is polynomial in the size of G. Therefore, there is a planar graph that is 3-colorable if and only if G is 3-colorable, and 3-coloring reduces to 3-coloring in planar graphs. Therefore 3-coloring is NP-hard even in planar graphs.

[2]The type of drawing here is really quite irrelevant; the only thing that matters is that we guarantee that there always is a crossing "at" two endpoints of the crossing edges.

3.3 Planar 3-SAT

The idea of replacing each crossing in a drawing with some small "gadget" that doesn't make the problem any easier can be applied to a number of other problems as well, proving that they are no easier in planar graphs than in general graphs. We will give one more example here (without details of the proof), which will also make it understandable why so many problems are still NP-hard in planar graphs.

Recall that 3-SAT is the problem where we are given n boolean variables $x_1, \dots, x_n$ and m clauses $c_1, \dots, c_m$, where each clause consists of at most three literals. We want to know whether there is an assignment of values TRUE and FALSE to the variables such that all clauses are satisfied.

This problem is not a graph problem, although a lot of graph problems can be shown to be NP-hard using 3-SAT. However, one can define a graph out of an instance of 3-SAT as follows:

- Create one vertex for every literal x_i and $\overline{x_i}$.
- Create one vertex for every clause c_j.
- Create an edge (ℓ_i, c_j) if and only if clause c_j contains literal ℓ_i.
- Create an edge $(x_i, \overline{x_i})$ for every variable.
- Create a cycle $x_1 - x_2 - x_3 - \dots - x_n - x_1$ of edges.

One should note that there are various definitions of the graph defined by 3-SAT out in the literature. For example, in some papers there is only one vertex per variable, and it represents both x_i and $\overline{x_i}$. In other variants, the last type of edges is not required. We use here the most general variant possible; since this already implies NP-hardness if the corresponding graph is planar, then for the other variants the problem is NP-hard as well.

So formally, an instance of Planar 3-SAT is an instance of 3-SAT for which (after suitable renumbering of the variables, if needed) the associated graph is planar. With an idea similar as to the one for 3-coloring (again using a crossing gadget), one can show the following:

Theorem 2 *3-SAT reduces to Planar 3-SAT, and in particular Planar 3-SAT is NP-hard.*

We will not even give the crossing gadget here; it is relatively complicated to draw and even more complicated to explain its correctness. See [Lic82] for details of the gadget and the formal proof of this theorem.

From this theorem, it is now straightforward to prove that VertexCover, IndependentSet and HamiltonianCycle are all NP-hard in planar graphs. Simply by re-inspecting the reductions from 3-SAT to these problems, it is not hard to see that the graph we are constructing is in fact identical (or very easily transformed into) the graph of 3-SAT. Hence by starting with an instance of planar 3-SAT instead, we construct an instance of VertexCover/IndependentSet/HamiltonianCycle for which the graph is planar, proving that the problem is NP-hard for planar graphs as well.

4 Problems that are polynomial in planar graphs

However, some problems that are ordinarily NP-hard become polynomial-time solvable for planar graphs, which is what we will study now.

4.1 Clique

Recall that Clique is the problem of finding the largest set of vertices that form a clique, i.e., have all possible edges between them. In general, this problem is NP-hard. However, for planar graphs it is easily solvable in polynomial time. Recall that K_5 is not planar, so no planar graph can have a clique of size 5 or larger. Finding the maximum clique is therefore a simple matter of testing all subsets of size 4 or less for whether they are a clique, and returning the largest set that is. This can be done in $O(n^4)$ time since there are only $O(n^4)$ such sets. (In fact, with a more sophisticated approach this can be done in linear time [PY81].)

4.2 Maximum Cut

Recall the maximum cut problem: Given a graph, we want to partition the vertices into sets A and B such that as many edges as possible are in the cut (A, B). In a different formulation, we want to find the minimum number of edges that must be deleted such that the graph becomes bipartite. It was a homework to show that this problem is NP-hard; this can be done with a reduction from NAE-3SAT that is quite similar to the one for 3-coloring.

However, amazingly so, the maximum cut problem becomes polynomial in planar graphs. We do not have time to give the detailed proof here, it can be found in Hadlock's paper ([Had75], see also [AI77]). The crucial idea is the observation that for planar graphs we have a dual graph, which helps in this case. Deleting the minimum number of edges to make the graph bipartite corresponds to contracting the minimum number of edges in the dual graph to make the dual graph Eulerian. The latter problem can be solved efficiently (it can be seen to be a variant of the Chinese Postman Problem) by computing a minimum-weight perfect matching between the vertices of odd degree.

4.3 Back to Satisfiability

The previous result that Maximum Cut is polynomial time solvable should give rise to some thinking. Where do the reductions that we had fail? Inspecting the reduction from NAE-3SAT again, one sees that the graph introduced during that reduction is exactly the graph of a satisfiability problem. So what does that tell us? Since NAE-3SAT reduces to Maximum Cut, this means that NAE-3SAT is polynomial time solvable for planar graph as well!

Theorem 3 *NAE-3SAT is polynomial time solvable if the graph defined by the satisfiability instance is planar.*

This theorem seems to have been pointed out explicitly for the first time in [KT00].

Another puzzling question is whether it wouldn't have been easier to show that 3-coloring is NP-hard in planar graphs by using the reduction from 3-SAT to 3-coloring that is known to exist. However, this approach does not work. During this reduction, we introduce an extra vertex that is connected to all variable-vertices, and nothing guarantees that this maintains planarity. (In fact, the special case of 3-SAT instances for which this graph is planar becomes polynomial time solvable, see [KK93].)

References

[AH77] K. Appel and W. Haken. Every planar map is four colorable. I. Discharging . *Illinois J. Math.*, 21(3):429–490, 1977.

[AHK77] K. Appel, W. Haken, and J. Koch. Every planar map is four colorable. II. Reducibili ty. *Illinois J. Math.*, 21(3):491–567, 1977.

[AI77] K. Aoshima and M. Iri. Comments on F. Hadlock's paper: "Finding a maximum c ut of a planar graph in polynomial time". *SIAM J. Computing*, 6(1):86–87, 1977.

[Aig84] M. Aigner. *Graphentheorie: Eine Entwicklung aus dem 4-Farben Prob lem.* Teubner Studienbücher, 1984. English translation: *Graph Theory: A Development from the 4-Color Problem*, 1987.

[CNS81] N. Chiba, T. Nishizeki, and N. Saito. A linear 5-coloring algorithm of planar graphs. *J. Algorithms*, 2:317–327, 1981.

[Fár48] I. Fáry. On straight line representation of planar graphs. *Acta. Sci. Math. Szeged*, 11:229–233, 1948.

[FPP90] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990.

[Had75] F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM J. Computing*, 4:221–225, 1975.

[KK93] J. Kratochvil and M. Křivanek. Satisfiability of co-nested formulas. *Acta Informatica*, 30(4):397–403, 1993.

[KT00] J. Kratochvil and Zs. Tuza. On the complexity of bicoloring clique hypergraphs of graphs. In *11th Annual Symposium on Discrete Algorithms (SODA 2000)*, pages 40–41. ACM, 2000.

[Lic82] David Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[PY81] C. Papadimitriou and M Yannakakis. The clique problem for planar graphs. *Information Processing Letters*, 13:131–133, 1981.

[RSST97] N. Robertson, D. Sanders, P. Seymour, and R. Thomas. The four-colour theorem. *J. Combin. Theory Ser. B*, 70(1):2–44, 1997.

[Ste51] S. Stein. Convex maps. In *Amer. Math. Soc.*, volume 2, pages 464–466, 1951.

[Wag36] K. Wagner. Bemerkungen zum Vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.