More on Isomorphism of Planar Graphs

We give $O(|V|^2)$ algorithms to solve labelled isomorphism on 2-connected planar graphs, connected planar graphs, and planar graphs. Linear time algorithms of the same flavor exist. A first step is to test if G_1 and G_2 are planar and return with failure if not. Also, we test if $|V(G_1)|=|V(G_2)|$ and return with failure if not. Otherwise We let $n=|V(G_1)|$.

Given our algorithm for connected planar graphs, our algorithm for general planar graphs proceeds as follows. We use depth-first search to determine, in O(n) time the number of components of G_1 and G_2, compute the components of G_1 and G_2, and determine the size of each component. We can assume that for some k, both G_1 and G_2 have k components, and that we have labelled the components of G_i as $U^i_1,\ldots,U^i_k$, and for i in {1,2}, we have an array, the j^{th} element of which contains the size of U^i_j and a copy of this component. Furthermore usng this array, in $O(|V|^2)$ time we can determine for every s, the number of components of G_1 and G_2 which have s vertices. If for any s, this number differs we return the fact that the graphs are not isomorphic and terminate.

We build an auxiliary bipartite graph on vertices (1,1),(1,2),...,(1,k), (2,1), (2,2), …,(2,k) where (1,j) is isomorphic to (2,i) if U^1_j is isomorphic to U^2_i. The two graphs are isomorphic precisely if every component of this bipartite graph has the same number of vertices on both sides of the bipartition. To test if U^1_j is isomorphic to U^2_i we first test if they have the same number of vertices (which takes O(1) time using our array), and then apply our algorithm for connected planar graphs if they have the same number of vertices.

Now, if there are a components of size b in G_1 (and hence also G_2) then the a^2 calls to our algorithm for connected graphs for pairs of components of size b takes $O(a^2b^2)=O((ab)^2)$ time. Since the sum of squares is at most the square of the sum, we see that our algorithm runs in $O(n^2)$ time in total.

We consider next how to use our algorithm for solving labelled isomorphism on 2-connected planar graphs, to get an algorithm for connected planar graphs. We use depth-first search to determine, in O(n) time the number of blocks of G_1 and G_2, compute the block trees T_1 and T_2 of G_1 and G_2, and determine the size of each block. We can assume that for some k, both G_1 and G_2 have k blocks, and that for each block node of the block tree, we have recorded the number of vertices of the block. Furthermore, in $O(|V|^2)$ time we can determine for every s, the number of components of blocks G_1 and G_2 with s vertices. If for any s this number differs we return the fact that the graphs are not isomorphic and terminate.

Since the block tree is unique, for any isomorphism f between G_1 and G_2, there is an isomorphism g between T_1 and T_2 such that (i) for every cutvertex node s of T_1, g(s) is a cutvertex node of T_2 and f maps the cut corresponding to s to the cut

corresponding to g(s), and (ii) for every block node s of T_1, g(s) is a block node of T_2 and f maps the block corresponding to s to the block corresponding to g(s).
We can find for each block tree, either a unique separator of the tree or a unique edge of the tree both of whose endpoints are separators. We know that the g corresponding to an isomorphism f must map separators of T_1 to separators of T_2. Since there is at most one cutvertex separator and at most one block node separator in each block tree, we know how g must behave on the separators. So, we root T_1 at a separator r_1 and T_2 at the separator r_2 to which r_1 must be mapped. We compute the depth of each node of each block tree. We know that for the g corresponding to any isomorphism f, (i) for any cutvertex node s of T_1, g(s) must be a cutvertex node of T_2 at the same depth such that there is an isomorphism f_s from the subgraph of G_1 consisting of the blocks of G_1 which are descendants of s to the subgraph of G_2 consisting of the blocks of G_2 which are descendants of g(s) which maps the cutvertex corresponding to s to the cutvertex coreesponding to g(s), and (ii) for any block node s of T_1, g(s) must be a block node of T_2 at the same depth such that there is an isomorphism f_s from the subgraph of G_1 consisting of the blocks of G_1 which are descendants of s to the subgraph of G_2 consisting of the blocks of G_2 which are descendants of g(s) which maps the block corresponding to s to the block coreesponding to g(s), and if s is not the root of T_1 maps the cutvertex corresponding to the parent of s to the cutvertex corresponding to the parent of g(s).

We consider the nodes of the two trees in decreasing order of depth. We want to determine for every cutvertex node s of T_1 and cutvertex node t of T_2 at the same heigh as s, whether there is an isomorphism f_s satisfying (i) with g(s)=t. We want to determine for every block node s of T_1 and block node t of T_2 at the same height as s, whether there is an isomorphism f_s satisfying (i) with g(s)=t.

Having solved this problem for nodes whose depth is one more than the depth of s, We can label the nodes at this higher depth, so that if they are in different trees then they have the same label precisely if such an isomorphim exists.

Now, if s and t are cutnode trees, such an isomorphism exists, precisely if there is a bijection between their children where we can only map a node to a node with the same label.

If s and t are block nodes then such an isomorphism exists precisely if there is a labelled isomorphism between the block corresponding to s and the block corresponding to t, where (a) we label the cutnodes of the children of s and t, using a pair (the original label, and the labels our algorithm has provided) and (b) if s is not the root then we label the cutvertex corresponding to the parent of s with a pair whose second component is new special label which is also assigned as the second component of the label of the cutvertex corresponding to the parent of t.

We can now use our algorithm for labelled planar isomorphism for 2-connected graphs to solve labelled planar isomorphism on connected planar graphs in quadratic time.

We consider next how to use our algorithm for solving labelled isomorphism on 3-connected planar graphs, to get an algorithm for 2-connected planar graphs.

We use depth-first search to determine, in O(n) time the number of strong triconnected components of G_1 and G_2, compute the strong 2-cut trees of G_1 and G_2, and determine the size of each strong tri-connected component. We can assume that for some k, both G_1 and G_2 have k strong tri-connected components , and that for each strong connected tricomponent node of the strong 2-cut tree, we have recorded the size of the strong tri-connected component. Furthermore, in $O(|V|^2)$ time we can determine for every i, the number of components of blocks G_1 and G_2 with I vertices. If for any i, this number differs we return the fact that the graphs are not isomorphic and terminate.

We proceed in essentially the same way, except that (1) for 2-cut nodes, we need to determine which if any bijections between the 2-cuts extend to a bijection of the graph consisting of the triconnected components below, and (2) for non-root tri-connected component, we need to determine which if any bijections of the 2-cut corresponding to the parent of the node extend in this faction.

If s and t are cut nodes, then if we fix one of the two possible bijections between them, then we simply need to check if there is a bijection between the children, where paired children get the same label.
Otherwise, since there are only O(|E|)=O(|V|) possible unlabeled bijections between the tricomponent corresponding to s and that corresponding to t, we can check in quadratic time, which if any extend to the desired f_s and what bijection they generate on the 2-cuts corresponding to the parents of s and t.

Finding an edge in a 3-connected graph whose contraction leaves it 3-connected.

Consider a vertex x of minimum degree in a 3-connected graph G. For any neighbour *y* of G, any 2-cut of *G/xy* not containing the vertex into which we contracted *xy* is a 2-cut of G. So, if y is not in a 2-cut of G-x then G/xy is 3-connected.

If G-x is triconnected and the strong 2-cut tree has only one vertex then G-x has no 2-cuts so we can contract any edge from y. If G-x is a cycle of length at least four and the strong 2-cut tree has only one vertex then every vertex of G other than x must see x (since it has degree at least three in G) and has degree exactly 3 in G-x. But x has degree at least four which is a contradiction.

Otherwise, the strong 2-cut tree has at least two leaves. For any such leaf l, let G_l be the corresponding cycle or triconnected graph. Let yz be the edge of G_l

corresponding to the 2-cut of G-x which is the parent of l. Now, x sees a vertex w of G_l-y-z, as G is 3-connected. So, w is in a 2-cut and G_l is a cycle of length at least four or we could contract xw. Now, every vertex of G_l-y-z has degree at least three in G and hence sees x and has degree exactly three in G. But now, x sees two vertices of G_l-y-z, for every leaf k and hence at least four neighbours. This contradicts the fact that it is a minimal degree vertex.

Thus, in linear time we can find the desired edge whose contraction leaves the graph 3-connected. As discussed in class Kawarabayashi, Li, and Reed have shown that for some constant c>0, there is a linear time algorithm which given a 3-connected planar graph G, finds an induced matching M with c|V(G)| edges such that contracting any subset of its edges leaves a 3-connected graph.

Quadratic and Linear Planar Embedding Algorithms for 3-connected Graphs

In linear time, we find an edge xy such that G/xy is 3-connected. We do so iteratively , constructing a sequence of 3-connected graphs G_o=G,G_1,…,G_{n-1}. Such that G_i is obtained from G_{i-1} by contracting some edge x_iy_i. Since G_{n-1} is a vertex we can find a planar embedding of it in constant time. As discussed in class, given a planar embedding of G_{i+1}, in linear time we can either either find a planar embedding of G_i or a K_5 or $K_{3,3}$ subdivision which shows it is not planar, in linear time. Recursively applying this algorithm for i=n-2 down to 0, we have a quadratic time algorithm to either determine that a 3-connected graph is not planar or find a planar embedding of it.

To improve the time complexity of the algorithm, We find, In linear time, an induced matching M with at least c|V(G)| vertices such that contracting any subset of the edges yields a 3-connected graph. We let G_1 be obtained by contracting the edges of this matching. Iterating this process, we construct a sequence of 3-connected graphs G_o=G,G_1,…,G_l. Such that G_i is obtained from G_{i-1} by contracting some matching with at last c|V(G_i)| edges for which contracting any subset of its edges yields a 3-connected graph, and |V(G_l)| is at most some large constant B. . Since G_l has bounded size, we can find a planar embedding of it in constant time. As discussed in class, given a planar embedding of G_{i+1}, in linear time we can either find a planar embedding of G_i or a K_5 or $K_{3,3}$ subdivision which shows it is not planar. Recursively applying this algorithm for i=l-1 down to 0, we have a linear time algorithm to either determine that a 3-connected graph is not planar or find a planar embedding of it.