Planarity Testing Revisited

Samir Datta, Gautam Prakriya

Chennai Mathematical Institute
India
{sdatta,gautam}@cmi.ac.in

Abstract. Planarity Testing is the problem of determining whether a given graph is planar while planar embedding is the corresponding construction problem. The bounded space complexity of these problems has been determined to be exactly Logspace by Allender and Mahajan [AM00] with the aid of Reingold's result [Rei08]. Unfortunately, the algorithm is quite daunting and generalizing it to say, the bounded genus case seems a tall order.

In this work, we present a simple planar embedding algorithm running in logspace. We hope this algorithm will be more amenable to generalization. The algorithm is based on the fact that 3-connected planar graphs have a unique embedding, a variant of Tutte's criterion on conflict graphs of cycles and an explicit change of cycle basis.

We also present a logspace algorithm to find obstacles to planarity, viz. a Kuratowski minor, if the graph is non-planar. To the best of our knowledge this is the first logspace algorithm for this problem.

1   Introduction

Planarity Testing, the problem of determining whether a given graph is planar (i.e. the vertices and edges can be drawn on a plane with no edge intersections except at their endpoints) is a fundamental problem in algorithmic graph theory. Along with the problem of actually obtaining a planar embedding, it is a prerequisite for many an algorithm designed to work specifically for planar graphs.

Our focus is on the bounded space complexity of the planarity embedding problem because we know that many graph theoretic problems like reachability [BTV09], perfect matching [DKR10,DKT10], and even isomorphism [DLN08,DLN+09] have efficient bounded space algorithms when provided graphs embedded on the plane.

Almost a decade ago, building on previous work by Ramachandran and Reif [RR94], Allender and Mahajan [AM00,AM04], proved that Planarity Testing is contained in SL and is L-hard. With Reingold's result [Rei08] proving SL = L, this gave a tight complexity theoretic classification for the problem. This seemed to be the end of the story as far as the problem of Planarity Testing vis-a-vis the logspace world was concerned.

The only catch was, the algorithm described in the paper was quite complicated - in fact a simpler SL algorithm was listed as one of the open questions in [AM00]. We would be satisfied with a complicate algorithm if all we were concerned with was pigeonholing the complexity of the problem. Planarity testing, however, happens to be a fundamental task in Topological Graph theory and a first step towards problems like toroidicity testing and bounded genus testing (also listed as open problems in [AM00]). Therefore, if we are to make any progress towards a tighter classification of these problems, for which no efficient bounded space algorithm is known, especially, if making a non-blackbox use of a planarity algorithm, it is advisable to search for a less daunting algorithm. This work is the result of this search.

In addition to its simplicity, the interplay between the properties of 3-connected planar graphs lends a certain elegance to the algorithm, at least in our, necessarily, biased eyes.

We also give a logspace algorithm to identify a Kuratowski minor if the graph is non-planar.

2   Related Work

Here we survey the related work very briefly - see the paper by Allender and Mahajan [AM00] for a detailed survey. The algorithmic aspects of Planarity Testing have been studied since the inception of Computer Science. It is clear that as far as sequential computation is concerned, a linear algorithm is optimal. Such algorithms include the one by Hopcroft and Tarjan [HT74]. The next result concerns parallel models of computation. Ramachandran and Reif proposed a very complicated algorithm which worked in logarithmic time and performed almost linar work and can be interpreted as placing the problem in the complexity class AC^1. The final frontier was bounded space computation and initial sorties had already taken place in the early eighties. Reif [Rei84] proved that planarity testing for degree 3 graphs is in the SL-hierarchy while Ja'Ja' and Simon [JS82a,JS82b] proved that planarity testing is in the NL-hirerarchy. After Nisan and TaShma's result [NTS95] and the Immerman-Szelpcsenyi Theorem [Imm88,Sze88] these bounds become SL and NL respectively. The paper by Allender and Mahajan completed the campaign when they proved that Planarity Testing was SL-complete and Reingold's result [Rei08] set the final seal by proving $\mathsf{SL} = \mathsf{L}$.

3   Outline of Algorithms

3.1   Outline of Planar Embedding Algorithm

We now motivate and informally describe Algorithm 1. It is easy to to see that for a planar graph G and any of its cycles, for the conflict graph w.r.t. the cycle is bipartite - the bipartitions being given by the bridges that are placed inside the cycle and those that are placed outside it, in some planar embedding.

Conversely, Tutte [Tut58] has shown that a graph is planar iff the conflict graph with respect to *every* cycle is bipartite. But since a graph potentially has exponentially many cycles a direct application of this result does not yield even a polynomial time algorithm. Thus we ought to seek a small set of cycles such that restricting our attention to the bipartiteness of their conflict graphs suffices to yield a planarity test. The set of fundamental cycles w.r.t. some spanning tree seems to be a natural candidate. Unfortunately, there are non-planar graphs which have a spanning tree such that the conflict graph w.r.t. each of the fundamental cycles is bipartite. e.g. in a K_5 with a spanning tree being a star centered at any vertex, the conflict graph w.r.t. any fundamental cycle has a single bridge.

Suppose, however that we are able to find a deterministic algorithm that constructs a valid planar embedding whenever its input is planar graph and either fails when supplied with a non-planar graph or outputs a non-planar embedding. Now, since, its easy to check (via

verification of Euler's formula) that an *embedding* is planar, we can eliminate any non-planar graphs at this stage.

Thus it suffices to focus on finding an embedding algorithm that works correctly under the promise that the given graph is planar. We can without much loss of generality, strengthen the promise and assume that the graph is, in addition to being planar, also 3-connected. This is because finding the triconnected components is known to be in logspace (see Lemma 1 and so is patching together the given planar embeddings of triconnected graphs (see Lemma 2).

Concentrating on 3-connected planar graphs we observe that:

– The graph has a unique planar embedding.
– The conflict graphs w.r.t. any cycle is connected (see Section 5.2), which enables us to bipartition the bridges in a unique way so that bridges in a partition lie on one side of the cycle in question.
– Ideally we would like to pick a small set of cycles and determine which edges lie inside them and somehow piece together the combinatorial embedding from this information. A natural choice for such a set is the set of fundamental cycles w.r.t. an arbitrary spanning tree. But the following two problems crop up:

  • Though we have a bipartition of bridges for each fundamental cycle , it is not clear which partition is mapped inside and which outside.
  • How do we actually piece together a combinatorial embedding once we know this information?

– The first problem can be solved if we knew at least one face of the unique embedding, because then we can just think of the face as the external face and stipulate that every edge (not part of the face) is contained inside it. Thus for every fundamental cycle, the bridge containing all the edges of the external face would lie outside the cycle - this fixes which bridges lie inside and which outside. But Corollary 1 tells us that for the graphs in question there is always a fundamental cycle which is a face. This, combined with Fact 2 ensures that we can find such a face in logspace.
– We solve the second problem in Section 5.4, where we first show that finding a solution is equivalent to a change of the cycle basis from fundamental cycles to faces. Then we show an explicit way to perform this inversion in logspace.
– Finally, we know that a 3-connected graph is planar iff every edge lies on exactly two induced non-separating cycles (see Fact 2). Thus, we can use this criterion to check for planarity of G.

3.2   Outline of Algorithm to find Kuratowski Minors

In Section 6 we describe an algorithm (Algorithm 2) to find a Kuratowski (i.e. a K_5 or $K_{3,3}$) minor in a non-planar graph. The algorithm indentifies a cycle with a non-bipartite conflict graph. It then finds an induced odd cycle in this conflict graph. Finally, it contracts some of the bridges and vertices of attachment of these bridges to yield a Kuratowski minor.

4   Definitions and Preliminaries

We assume familiarity with basic complexity theory in general and bounded space classes in particular, as described in any standard text e.g [AB09]. We will also assume familiarity with graph theory as described in a text like [Die05,Wes00]. Below, we explicate all non-standard material we will have occasion to use.

Definition 1. *The bridges of a cycle C consist of:*

- *For every connected component X of $G \setminus C$, the induced graph $G[X \cup A_X]$ where $A_X \subseteq C$ are the vertices of C adjacent to some vertex of X (the so called points of attachment).*
- *The chords of C - here the endpoints of C are its points of attachment*

Definition 2. *Two bridges B_1, B_2 of a cycle C conflict iff either of the following conditions hold:*

- *a_i, a_i' are two points of attachment of B_i w.r.t. C for $i \in \{1, 2\}$ such that they occur in the order a_1, a_2, a_1', a_2' along the cycle C.*
- *B_1, B_2 have three common points of attachment w.r.t. cycle C.*

We can extend the definition of conflict to sets of bridges $\mathcal{B}_1, \mathcal{B}_2$ by the existence of two sets of conflicting vertices of attachment in the above sense, belonging to (not necessarily the same bridge of) $\mathcal{B}_1, \mathcal{B}_2$ respectively. The conflict graph $H_C(G)$ of a graph w.r.t. a cycle C is formed by taking the bridges of C as vertices and joining two vertices by an edge iff they conflict.

Definition 3. *Given a spanning tree T of a biconnected graph G, and an edge $e \in E(G) \setminus E(T)$, the graph $G \cup e$ contains a unique cycle $C(e)$ called the fundamental cycle of e. We say that a face of an embedded planar graph is fundamental if it is a fundamental cycle of some non-tree edge (with respect to some fixed spanning tree).*

The following is an easy consequence of Reingold's result [Rei08] since we can count the number of components in $(G \cup e) \setminus e'$. We single it out of a set of similar elementary graph computations summarized in Section 3.2 of [AM00] which can be done in logspace because it is of special significance for us.

Fact 1 *The list of edges in each fundamental cycle of G w.r.t. a spanning tree T can be obtained by a logspace transducer.*

Fact 2 *(Proposition 4.2.10 and Theorem 4.5.2 [Die05]) The faces of a 3-connected planar graph G are exactly the induced non-separating cycles of G. Further, a 3-connected graph is planar iff every edge lies on exactly two induced, non-separating cycles.*

Proposition 1. *Given a the cyclic order of vertices in every face of a biconnected embedded graph, it is possible to construct in logspace, a combinatorial embedding of the graph.*

4

Proof. Call an ordered triplet,T of vertices (u, v, w) an angle of an embedded graph if $(u, v), (v, w)$ are consecutive edges on some face of the embedding. Call two triplets $T_i = (u_i, v_i, w_i)$ for $i \in \{1, 2\}$ adjacent if $v_1 = v_2$ and either $w_1 = u_2$ or $u_1 = w_2$. Then it is clear that the undirected graph with angles as vertices and angle adjacencies as edges, forms a set of disjoint[1] cycles[2]. The ordering on vertices induced by the angles is exactly the combinatorial embedding. $\square$

5 Planarity Testing

5.1 Reduction to the Triconnected case

Lemma 1. *(Lemma 3.3 [DNTW09]) The triconnected components of a graph can be obtained in logpace*

It is a well known fact that if a graph is non-planar then one of it's 3-connected components is non-planar. [Wes00]

Lemma 2. *Given a combinatorial planar embedding of the triconnected components of a graph, it is possible to obtain the biconnected planar embedding of the graph in logspace.*

Proof. Consider the tri-connected component , separating pair tree. This tree can be constructed in logspace [DNTW09]. We will use the tree as an index to construct the embedding of the graph, given the embedding of it's 3 components. Given a vertex in a tri-connected component,we can obtain the clockwise order of the edges around the vertex using the given combinatorial embedding.

Given a node in the tree corresponding to separating pair u, v we arrange the tri-connected components around u in the order they appear in the tree and around v in the opposite order. This gives us an ordering of edges around each vertex in the graph. We repeat this procedure for each of the nodes corresponding to a separating-pair in the tree. This gives us an embedding of the graph. To move from biconnected graphs to general graphs, see [AM00]. $\square$

5.2 The Conflict Graph

Lemma 3. *Given a 3-connected planar graph G and an arbitrary cycle C in the graph the conflict graph $H_C(G)$ is bipartite and connected.*

Proof. It is easy to see that for an embedded planar graph G and any cycle C contained in it, the conflict graph, $H_C(G)$ is bipartite (with the bipartition being given by whether a bridge is mapped "inside" or "outside" the cycle). We now show that $H_C(G)$ is connected.

For every pair of points $x, y \in C$ there are two paths from x to y along C. We will call these paths $P_{x,y}$ and $P'_{x,y}$. Let $\mathcal{B}$ be a set of bridges that correspond to one of the connected components of $H_C(G)$.

Pick a pair u, v s.t

---

[1] since any angle has a unique middle vertex
[2] since every angle is adjacent to exactly two angles on a vertex

- u and v are attachment points of bridges in $\mathcal{B}$.
- u and v are not adjacent.
- The attachment points of all bridges in $\mathcal{B}$ lie in either $P_{u,v}$ or $P'_{u,v}$. (Assume WLOG that they lie in $P_{u,v}$).

We now show that one can always find such a pair u, v. Pick a bridge $B' \notin \mathcal{B}$. The attachment points of B' divide C into a number of segments .(atleast 3 since every bridge has 3 or more points of attachment.) Since no bridge in $\mathcal{B}$ conflicts with B' and the vertices corresponding to $\mathcal{B}$ in $H_C(G)$ form a connected component, it follows easily that all points of attachments of $\mathcal{B}$ must lie in one of the segments. This implies that there is a point in C(a point of attachment of B') that is not a point of attachment of any bridge in $\mathcal{B}$. Therefore we can find a pair u, v with the properties listed above.

Now, for every point $x \notin \{u, v\}$ on $P_{u,v}$, $\exists$ a bridge $B \in \mathcal{B}$ with points of attachment b_1, b_2 s.t b_1 precedes x and b_2 succeeds x in $P_{u,v}$. If not then $\forall B \in \mathcal{B}$ either all points of attachment of B lie between u and x or all of them lie between x and v. This implies that in $H_C(G)$ vertices corresponding to bridges in $\mathcal{B}$ u as a point of attachment are not connected to vertices corresponding to bridges with v as a point of attachment. This is a contradiction since $\mathcal{B}$ corresponds to a connected component in $H_C(G)$.

Now, if $\{u, v\}$ is not a separating pair then $\exists$ a bridge $B \notin \mathcal{B}$ with points of attachment in both $P_{u,v} \setminus \{u, v\}$ and $P_{u,v} \setminus \{u, v\}$. From the above it follows that B' conflicts with a bridge in $\mathcal{B}$ which is not possible. Therefore, $\{u, v\}$ is a separating pair. $\qquad\square$

5.3   Inside and Outside a Fundamental Cycle

Next we fix which bridges w.r.t. each of the fundamental cycles are mapped inside and which are mapped outside the concerned cycle. Basically we find one *face* of the grap and call it the external face. Thus every bridge containing this face is mapped outside any other cycle. This completely fixes the 2-coloring.

Proposition 2. *Any biconnected embedded planar graph has a fundamental* face *(i.e. a fundamental cycle which is also a face) w.r.t. each of its spanning trees.*

Proof. It is well known that the set of dual edges of the non-tree edges of a biconnected planar graph forms a tree (after reducing all multi-edges to a single edge) - the so called dual tree. This follows from observing that cycles in the primal correspond to cuts in the dual (Proposition 4.6.1 [Die05]) and therefore the set of edges dual to the non-tree edges are connected in the dual graph, the proof being completed by applying Euler's relation to verify that the number of such edges is exactly one less than the number of faces. It is easy to see that the face of the original graph corresponding to a leaf of the dual tree is a fundamental face. $\qquad\square$

A direct consequence of Fact 1, Proposition 2, Fact 2 is the following (since finding whether a cycle is induced and non-separating is a logspace predicate given the list of edges in the cycle on the input tape).

Corollary 1. *Every 3-connected planar graph has at least one fundamental face and this can be found by a Logspace transducer.*

Knowing which edges of the graph map to the region enclosed by each of the fundamental cycles, we proceed to construct a purported embedding of the given graph. If the graph is indeed planar we will obtain a valid planar embedding, else we will not be able to do so, giving an effective planarity test (which produces an embedding as a side-effect).

5.4 Obtaining a Planar Embedding

Here we exhibit a way to identify the edges in each face of the given 3-connected graph. Fact 2 then gives a way of effectively checking if the graph is plnar.

At the heart of the proof is a change of basis in the cycle space over $\mathbb{Z}_2$ Somewhat msurprisingly the solution to the equations obtained in the previous section tell us which faces sum up (over $\mathbb{Z}_2$) to give a particular fundamental cycle. Notice that we have only an "implicit" representation of the faces (see below) and while we are seeking for an explicit representation in terms of which edges constitute a face. Since fundamental cycles and internal faces both form a basis of the cycle space over $\mathbb{Z}_2$, we just need to invert the matrix that expresses the fundamental cycles as a linear combination of faces. Though seemingly this would place the problem in $\oplus\mathsf{L}$ we give an explicit way to perform this inversion which yields a L upper bound.

Returning to the "implicit" representation of faces, there is a natural bijection between non-tree edges and faces viz. one that maps a non-tree edge e to that face $f(e)$ adjacent to e which lies *inside* the fundamental cycle $C(e)$ (w.r.t. the external face $C(e_0)$). With faces labeled in this way, the solution of the preceding section tells us which faces are contained within the fundamental cycle $C(e)$.

Let us start by fixing some notation. For distinct non-tree edges e_1, e_2, define $e_1 \prec e_2$ iff in a 2-coloring of the conflict graph $H_{C(e_2)}(G)$, the colors of the vertices corresponding to bridges containing e_0, e_1 get different colors. Intuitively, this means that (if G is planar) $C(e_2)$ separates e_1 from e_0 in the unique planar embedding of G.

For each $e \in E(G) \setminus \{e_0\}$, let $P(e)$ denote:

$$\{e' \in E(G) \setminus (E(T) \cup \{e\}) | e' \prec e \wedge \nexists e'' : e' \prec e'' \prec e\},$$

Further, let $F(e)$ denote

$$\bigoplus_{e' \in P(e) \cup \{e\}} C(e')$$

Notice that for sets A, B the set $A \oplus B$ denotes the symmetric difference of the two sets and the notation above refers to an iteration of this operation.

If the graph is 3-connected planar then in its unique planar embedding with $C(e_0)$ as the external face, $P(e)$ consists of the set of non-tree edges that are enclosed by e but not by any e'' which is also enclosed by e. Intuitively, it clear that these are exactly the non-tree edges occurring on the face $f(e)$ (see description in the preceding paragraph). We will make this intuition precise and in fact, show the following:

Lemma 4. *For each non-tree edge $e \neq e_0$, the face $f(e)$ consists exactly of the edges in the set $F(e)$.*

Proof. Given a fundamental cycle C let $\phi(C)$ represent the number of faces that lie inside C and $\psi(C)$ represent the number of fundamental cycles that lie inside C including C. Notice that for fundamental cycles which are also faces, $\phi(C) = \psi(C) = 1$. In fact for every fundamental cycle C, as an easy consequence of Euler's formula it follows that $\phi(C) = \psi(C)$.

Let $e \in E(G) \setminus E(T)$. We will first show that $F(e)$ is a face. Since each $e' \in P(e)$ lies inside $C(e)$ and for every non-tree $e'' \notin P(e)$, such that $e'' \prec e$, there is exactly one $e' \in P(e)$ such that $e'' \prec e'$. There is one such e' because we know that every non-tree edge enclosed by $C(e)$ is either enclosed by some other fundamental cycle $C(e_1)$ or otherwise is in $P(e)$. Thus by induction on the "enclosure depth" of an edge we get an $e' \in P(e)$ which encloses it. The uniqueness follows from observing that if both $e \prec e' \wedge e \prec e'_1$ then either $e'_1 \prec e'$ or $e' \prec e'_1$. - this is due to planarity - therefore both e', e'_1 cannot be in $P(e)$. Thus we get,

$$\psi(C(e)) = 1 + \sum_{e' \in P(e)} \psi(C(e'))$$

therefore:

$$\phi(C(e)) = 1 + \sum_{e' \in P(e)} \phi(C(e'))$$

Now since every fundamental cycle can be written as a sum of internal faces of G and $F(e)$ is a sum of a set of fundamental cycles (where all computation is over $\mathbb{Z}_2$), it is also a sum of some internal faces. Since only faces contained within $C(e)$ figure in this sum, $F(e)$ must be a disjoint sum of some cycles enclosed by $C(e)$. We can be more specific, the sum:

$$\bigoplus_{e' \in P(e)} C(e')$$

includes exactly the faces inside $C(e)$ which are also faces inside some $C(e')$ for $e' \in P(e)$. Thus $F(e)$ includes exactly the faces contained inside $C(e)$ but not in any $C(e')$ for $e' \in P(e)$. Now, it is easy to see from the expression for $\phi(C(e))$ that there is exactly one such face, so it follows from the linear independence of the faces that this must be $f(e)$. $\square$

Thus, we can complete the proof of the following:

Theorem 1. *Given a graph G, constructing a planar embedding for G if it is planar and otherwise rejecting it, can be done in logspace.*

Proof. Given a graph we obtain its 3-connected components using Lemma 1. If each triconnected compoenent is planar, we will successfully obtain a planar embedding for each component and then obtain the proof with the aid of Lemma 2. If some triconnected component is not planar, we will either obtain an $F(e)$ which is not induced or is separating or we will obtain an edge lying on at least three $F(e)$'s and therefore reject.

```
    Input       : Graph G
    Promise     : G is 3-connected
    Output      : A planar embedding of G if planar and ∅ otherwise
 1  Compute a spanning tree T in G;
 2  if there is no fundamental face w.r.t. T then return ∅;
 3  Let C₀ = C(e₀) be a fundamental face;
 4  foreach fundamental cycle C(e) ≠ C₀ do
 5  │   Construct conflict graph H_{C(e)}(G);
 6  │   if H_{C(e)}(G) is not bipartite then return ∅;
 7  │   2-color H_{C(e)}(G);
 8  │   foreach non-tree edge e′ ≠ e do
    │   │   // Let B_{e′,e} be the bridge of C(e) containing e′
 9  │   │   if B_{e′,e} gets color different from B_{e₀,e} in H_{C(e)}(G) then let e′ ≺ e;
10  │   end
11  end
12  foreach fundamental cycle C(e) ≠ C₀ do
13  │   Let P(e) = {e′ ∈ E(G) \ (E(T) ∪ {e})|e′ ≺ e ∧ ∄e″ : e′ ≺ e″ ≺ e};
14  │   Let F(e) = ⊕_{e′∈P(e)∪{e}} C(e′);
15  end
16  Let F(e₀) = C₀;
17  foreach edge e ∈ E(G) do
18  │   if e doesn't lie in exactly two F(e′)'s then return ∅;
19  end
    // Follow the proof of Proposition 1 to construct an embedding of G
20  return planar embedding of G;
```

Algorithm 1: Planarity Testing and Embedding Algorithm for 3-connected graphs

6   Finding Kuratowski Subgraphs

We describe an an algorithm to obtain a Kuratowski subgraph given a cycle with a non-bipartite conflict graph. As a consequence, to obtain a Kuratowski subgraph, it is sufficient to find such a cycle in a subgraph G' of G. We do this in a series of lemmas:

Lemma 5. *Given a non-planar graph G, we can, in logspace, find a non-planar subgraph G' and a cycle $C \subseteq G'$ s.t. $H_C(G')$ is non-bipartite.*

Proof. Given access to a routine for planarity checking, and a non-planar graph G, we can find a minimal non-planar subgraph G' of g in the following sense. Order the edges of G arbitrarily as $e_1, \ldots, e_m$. Now consider the smallest i such that the graph G' formed by the union of the first i edges $e_1, \ldots, e_i$ is non-planar. Notice that this means that $G' \setminus e_i$ is necessarily planar.

Now, we show how one can find a cycle in this graph G' such that the conflict graph $H_C(G')$ is non-bipartite. Construct a planar embedding of $G' \setminus \{e\}$. The endpoints, say x, y, of e_i must lie on different faces of this embedding because G' is non-planar.

Find a path in the dual graph of the embedding between any two faces F_x, F_y incident respectively on x, y. (this path must avoid any other faces incident on x or y.) The symmetric difference of the faces in this path is a cycle C one of whose bridges is e_i.

We claim that $H_C(G')$ is non-bipartite. If it were bipartite then it would be possible to give an orientation to each of it's bridges such that conflicting bridges got opposing orientations.

```
    Input       : Graph G
    Promise     : G is non-planar
    Output      : A Kuratowski Minor
 1  Find a non-planar subgraph G' and a cycle C ⊆ G' s.t. H_C(G') is non-planar ;
 2  begin
 3  │   Fix an ordering of edges of G ;
 4  │   Find smallest edge e_i = (x, y) such that graph G' on {e_1, ..., e_i} is non-planar;
 5  │   Find planar embedding of G' \ e_i;
 6  │   Let F_x, F_y be two faces containing x, y respectively;
 7  │   Find a simple path between F_x, F_y in dual(G') which avoids all other faces containing x, y;
 8  │   Let C be the symmetric difference of the faces along this path;
    │   // H_C(G') is non-bipartite
 9  end
10  Find an induced odd cycle C̄ in H_C(G');
11  begin
12  │   Find a spanning tree of H_C(G') and 2-color it;
13  │   Pick a non-tree edge e with both endpoints of same color, so that all the chords of fundamental cycle
    │   C(e) have different colors;
14  │   Walk on the tree between endpoints of e and "short-circuit" as many tree-paths, by chords, as possible;
15  end
16  Find a cycle with 3 mutually conflicting bridges;
17  Contract the bridges to single vertices (if they originally contained a vertex) or edges;
18  Find a Kuratowski minor of this graph by brute force;
19  Expand back to a Kuratowski minor of the original graph;
```

Algorithm 2: Algorithm for finding Kuratowski Minors

But since the bridges of C are all planar, this would imply that G' is planar . Therefore, $H_C(G')$ is non-bipartite.

Lemma 6. *Given a non-planar graph G' and a cycle C witnessing this via the non-bipartiteness of $H_C(G')$, we can, in logspace, find an induced odd cycle $\bar{C}$ in $H_C(G')$.*

Proof. Since the conflict graph $H_C(G')$ is non-bipartite, it contains an odd cycle. We aim to find an induced odd cycle in this graph. For this consider a spanning tree of $H_C(G')$. 2-color the spanning tree. Notice that there must exist a non-tree edge between two vertices with the same color (else the graph would have been bipartite). Find a non-tree edge e such that all the chords of the fundamental cycle $C(e)$ get opposite colors - a simple induction on the number of chords that a fundamental cycle has, shows that such an edge must exist and locating it in logspace is easy.

   We will construct a chordless cycle from $C(e)$ by replacing some tree paths by chords of $C(e)$. To do this, let $e = (u_1, u_k)$ and let the vertices of the tree path from u_1 to u_k be $u_2, \ldots, u_{k-1}$ in order. Call the point u_i of a chord (u_i, u_j) for $i < j$ as the origin of the chord and u_j the end of the chord. Now start walking from u_1 to u_k along the tree path, outputting tree edges till the origin of a chord is encountered. Output the chord and move on to the end of the chord and repeat. It is easy to see that the edges output by this procedure along with the non-tree edge (u_1, u_k) form an induced cycle because either the origin or endpoint of any other chord of $C(e)$ is not on this cycle. Also, because the endpoints of all edges on this cycle are oppositely colored except for (u_1, u_k), this is an odd cycle.

At this point we have a cycle C in G with a non-bipartite conflict graph $H_C(G')$ and an induced odd cycle $\bar{C}$ in $H_C(G')$ witnessing this. Now we prove the following,

Lemma 7. *Given an induced odd cycle $\bar{C}$ in $H_C(G')$, we can, in logspace, find a Kuratowski subgraph.*

Proof. First, suppose that two conflicting bridges B_1 and B_2 in the odd cycle share 3 points of attachment a_1, a_2 and a_3. If either of the bridges(B_1 say.) has another point of attachment a_4 then, clearly the points of attachment a_1, a_3 of B_2 alternate with the points of attachment a_2, a_4 of B_1. Therefore, it is sufficient to deal with the case when the case where both B_1 and B_2 have just 3 points of attachment. In this case it is not difficult to see that any bridge that conflicts with B_1 also conflicts with B_2 and vice-versa. Hence $\bar{C}$ must be a 3-cycle. We deal with this case in lemma 8.

Next, consider the case where $\bar{C}$ is an odd cycle of size greater than 3 and the conflict of a bridge B in the odd cycle with any other bridge in the odd-cycle is witnessed by 2 points of attachment of B i.e we exclude the possibility that 2 bridges share 3 points of attachment.

It is easy to see that $\exists$ 2 points of attachment for every bridge that witness both it's conflicts. Thus by contracting all bridges (excluding points of attachment) to single points and removing edges of attachment, one can reduce all bridges to paths while maintaining all conflicts. This can be done in logspace since for each bridge one only needs to remember the witnesses of conflict of the bridge with its neighbours (in $\bar{C}$).(atmost 8 points.)

If we label the vertices of $\bar{C}$ as $v_{B_0}, v_{B_1}, \ldots, v_{B_{2k}}$ $(k > 1)$ for some positive integer k, and the points of attachment of bridge B_i as u_i, v_i then the points occur along C in the order: $u_0, v_{2k}, u_1, v_0, u_2, v_1, \ldots, u_{2k-1}, v_{2k-2}, u_{2k}, v_{2k-1}$.

Fig. 1. In the figure, we have a cycle with the 9-cycle as it's conflict graph . On treating $\{6, 8, 10\}$ as a single point and leaving out the edges $(6, 7), (7, 8), (8, 9)$ and $(9, 10)$, we obtain a subdivision of K_5

Now, consider the cycle $u_{2k}, u_0, v_{2k}, v_0, U'$ (Where $U' = \{\{v_1, u_3\} \cup B_3 \{v_3\} \cup \ldots \cup \{u_{2k-1}\}\}$) along with the bridges B_{2k}, B_0 the paths connecting u_0 and U', v_{2k} and U' and the path $\{v_0, u_2\} \cup B_2 \ldots \{v_{2k-2}, u_{2k}\}$ Clearly, this is a K_5 minor.(see Fig.1 for an example.)

Finally we are left with the following case:

Lemma 8. *Given a cycle in G' with three mutually conflicting bridges, it is possible to extract a Kuratowski minor of G' in logspace.*

Proof. Clearly, it is sufficient to consider 4 points of attachment for each bridge since 4 points of attachment suffice to witness conflict with two other bridges. We can contract the bridges (excluding the points of attachment) to single points and reduce the problem of finding a Kuratowski subgraph in the above graph to that of finding a Kuratowski minor in a non-planar graph with atmost 15 vertices and 24 edges. Since this is a constant sized graph we can find a Kuratowski minor by brute-force.

Acknowledgements

References

AB09. Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, New York, NY, USA, 1st edition, 2009.

AM00. Eric Allender and Meena Mahajan. The complexity of planarity testing. In *Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 87–98, 2000.

AM04. Eric Allender and Meena Mahajan. The complexity of planarity testing. *Inf. Comput.*, 189(1):117–134, 2004.

BTV09. Chris Bourke, Raghunath Tewari, and N. V. Vinodchandran. Directed planar reachability is in unambiguous log-space. *TOCT*, 1(1), 2009.

Die05. Reinhard Diestel. *Graph Theory (Graduate Texts in Mathematics).* Springer, 2005.

DKR10. Samir Datta, Raghav Kulkarni, and Sambuddha Roy. Deterministically isolating a perfect matching in bipartite planar graphs. *Theory Comput. Syst.*, 47(3):737–757, 2010.

DKT10. Samir Datta, Raghav Kulkarni, and Raghunath Tewari. Perfect matching in bipartite planar graphs is in ul. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:201, 2010.

DLN08. Samir Datta, Nutan Limaye, and Prajakta Nimbhorkar. 3-connected planar graph isomorphism is in log-space. In *Proceedings of the 28th annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 153–162, 2008.

DLN$^+$09. Samir Datta, Nutan Limaye, Prajakta Nimbhorkar, Thomas Thierauf, and Fabian Wagner. Planar graph isomorphism is in log-space. In *IEEE Conference on Computational Complexity*, pages 203–214, 2009.

DNTW09. Samir Datta, Prajakta Nimbhorkar, Thomas Thierauf, and Fabian Wagner. Graph isomorphism for $K_{3,3}$-free and K_5-free graphs is in log-space. In *FSTTCS*, pages 145–156, 2009.

HT74. John E. Hopcroft and Robert E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974.

Imm88. Neil Immerman. Nondeterministic space is closed under complementation. *SIAM J. Comput.*, 17(5):935–938, 1988.

JS82a. Joseph JáJá and Janos Simon. Parallel algorithms in graph theory: Planarity testing. *SIAM J. Comput.*, 11(2):314–328, 1982.

JS82b.   Joseph JáJá and Janos Simon. Space efficient algorithms for some graph theoretical problems. *Acta Inf.*, 17:411–423, 1982.

NTS95.   Noam Nisan and Amnon Ta-Shma. Symmetric logspace is closed under complement. *Chicago J. Theor. Comput. Sci.*, 1995, 1995.

Rei84.   John H. Reif. Symmetric complementation. *J. ACM*, 31(2):401–421, 1984.

Rei08.   Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4), 2008.

RR94.   Vijaya Ramachandran and John H. Reif. Planarity testing in parallel. *Journal of Computer and System Sciences*, 49:517–561, 1994.

Sze88.   Róbert Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Inf.*, 26(3):279–284, 1988.

Tut58.   W. T. Tutte. A homotopy theorem for matroids , I, II. *Trans. AMS*, 98:144–174, 1958.

Wes00.   Douglas B. West. *Introduction to Graph Theory (2nd Edition)*. Prentice Hall, August 2000.