Planar Graph Isomorphism is Complete for Log-space

Prajakta Nimbhorkar
(Joint work with Samir Datta, Nutan Limaye, Thomas Thierauf, Fabian Wagner)

The Institute of Mathematical Sciences, Chennai, India.

September 27, 2008

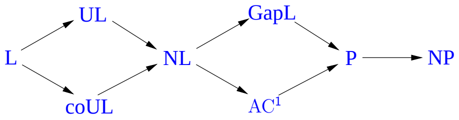
The Planar Graph Isomorphism Problem

Graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* $(G_1 \cong G_2)$ if there is a bijection $\phi : V_1 \to V_2$ such that

$$(u, v) \in E_1 \Leftrightarrow (\phi(u), \phi(v)) \in E_2$$

- $G = (V, E)$ is *planar* if it can be drawn on a plane without any crossing of edges.
- Articulation point: $v \in V$ s.t. $G(V \setminus v, E')$ is not connected.
- Biconnected graph: Graph with no articulation points.
- Separating pair: Pair $\{u, v\}$ of vertices s.t. $G(V \setminus \{u, v\}, E')$ is not connected.
- 3-connected graph: does not have a separating pair.

Overview of complexity classes and previous results for GI



$$\begin{tikzcd}
 & UL \arrow[dr] & & GapL \arrow[dr] & & \\
L \arrow[ur] \arrow[dr] & & NL \arrow[ur] \arrow[dr] & & P \arrow[r] & NP \\
 & coUL \arrow[ur] & & AC^1 \arrow[ur] & &
\end{tikzcd}$$

Figure: Containment of complexity classes

Graph Class	Lower Bound	Upper Bound
General graphs	GapL	NP
Trees	L	L
Planar graphs	L	AC^1
3-connected planar graphs	L	UL $\cap$ coUL

Table: Previously known bounds for GI

Our results

Theorem

3-connected planar GI $\in$ L. (*To appear in FSTTCS 2008. On arxiv: Jun'08*)

Theorem

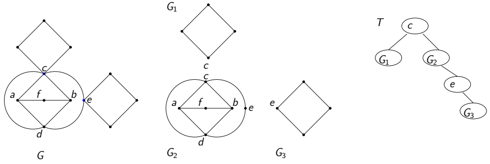
Planar GI $\in$ L. (*New, on arxiv: Sept'08*)

Outline of algorithm

- Step 1: Decompose G and H into biconnected components and construct the *biconnected component trees*.
- Step 2: Decompose each biconnected component into triconnected components and construct the *triconnected component trees*.
- Step 3: Check isomorphism for triconnected components.
- Step 4: Deduce isomorphism for biconnected components.
- Step 5: Deduce isomorphism for G and H.

Step 1: Decomposition into biconnected components

1. Identify and remove articulation points.
2. Identify the components.
3. Put copies of articulation points into respective components.
4. Construct the biconnected component tree.



Claim

This decomposition is unique and is log-space computable.

Step 2: Decomposition into triconnected components

1. Identify and remove all the separating pairs.
2. Identify the triconnected components.
3. Put a copy of each separating pair into respective components.
4. Between each copy, add a *virtual* edge.
5. Construct the triconnected component tree.

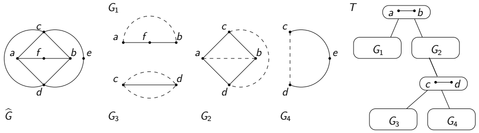


Figure: Decomposition into triconnected components

Claim

The triconnected component tree is unique and is log-space computable.

Step 3: Check isomorphism of 3-connected planar graphs

Planar combinatorial embedding ρ: Cyclic ordering of edges around each vertex in a planar drawing of G.

Crucial Ideas:

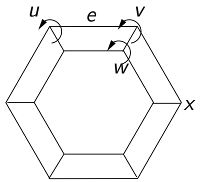
- A 3-connected planar graph has exactly two combinatorial embeddings, one being the inverse of the other [Whi33].
- Planar combinatorial embedding of G can be computed in log-space [AM00, Rei05].

Steps in the algorithm:

1. Fix a planar combinatorial embedding ρ of G.
2. Choose an edge $e = \{u, v\}$ and one of its end-points, say u.
3. Perform a "guided walk" on G in log-space, starting from $\{u, v\}$ and output the list of edges traversed.
4. Relabel the vertices according to their first occurence in this list.
5. $canon(G, \rho, \{u, v\}, u)$ is this list of edges with new vertex-labels.

"Guided walk" on G in log-space

- The "guided walk" is done using a universal exploration sequence (UXS).
- UXS: Sequence of offsets $\langle \tau_1, \ldots, \tau_{poly(n)} \rangle$ s.t. a walk according to a UXS visits all vertices of G.
- An (n, d)-UXS is log-space constructible[Rei05]. (For d regular graph on n vertices)
- If edge $\{x, y\}$ is traversed at step $i - 1$ of the walk, then in the i^{th} step, τ_i^{th} neighbour of y counted from x in counter-clockwise direction is chosen.



UXS=$\{1, 2, 0, 1, 1, \ldots\}$

List of edges=$\{(u, v), (v, w), (w, v), (v, x), \ldots\}$

New labels of vertices=$\{u = 1, v = 2, w = 3, x = 4, \ldots\}$

Figure: Example of a traversal using a UXS

Step 4: From triconnected to biconnected planar GI: Lindell's tree-isomorphism algorithm

Given two trees S and T, with roots s and t, $S < T$ according to Lindell's algorithm if

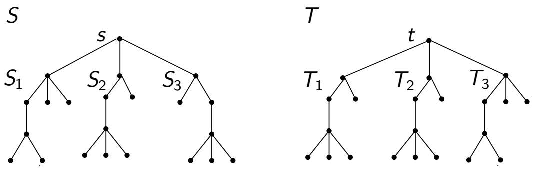
1. $|S| < |T|$, or
2. $|S| = |T|$ but s has fewer children than t, or
3. Equality is found in steps 1 and 2 but recursively $(S_1 \leq \ldots \leq S_k) < (T_1 \leq \ldots \leq T_k)$ lexicographically.

Note: Step 3 partitions the subtrees into *isomorphism classes* i.e. classes of isomorphic subtrees.

Steps 1,2 can be done in log-space.

Step 3 can also be done in log-space with a careful implementation.

Details of Lindell's algorithm



To do Step 3 in log-space:

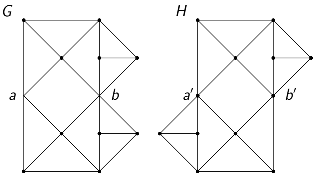
- Partition the children of S and T into *size-blocks* according to the sizes of their subtrees.
- Compare S_i and T_j only if they are in the same size-block.
- Store comparison counts for only one subtree at a time.
- Find c_1, c_2 : number of smallest children of S and T in block B (i.e. the smallest isomorphism class of B).
  If $c_1 > c_2$ then $S < T$.
  If $c_1 = c_2$ then check the number of next smallest children.
- After one block is processed, go to next block.

Analysis of Lindell's algorithm

- For a block B, comparison counts $\leq p$, $p = |B|$.
- Each subtree in B has size $\frac{n}{p}$.
- To make recursive call for subtrees S_i and T_j, only comparison counts need to be pushed on stack i.e. $O(\log p)$ bits.
- Thus the space used is $S(n) = S\left(\frac{n}{p}\right) + O(\log p) = O(\log n) \text{ for } p \geq 2$
- If $p = 1$, nothing needs to be stored on stack!

Isomorphism of triconnected component trees

Basic Idea: Order triconnected component trees similar to Lindell's algorithm.
Note: Lindell's algorithm + Isomorphism algorithm for triconnected planar graphs is not enough.
Reason: Different *orientations* of triconnected components.



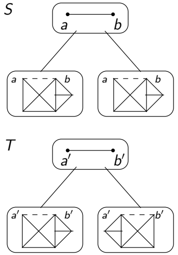
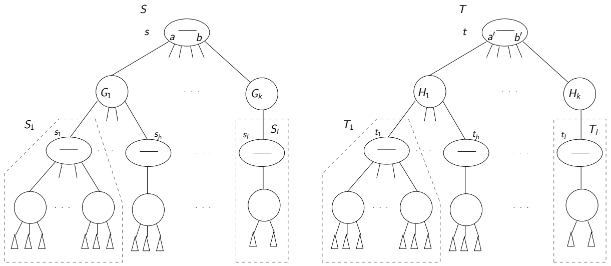


Figure: An example of non-isomorphic graphs with the same triconnected component tree

Isomorphism of triconnected component trees (continued)



Define $S < T$ if

1. $|S| < |T|$, or
2. $|S| = |T|$ but s has fewer children than t, or
3. Equality is found in steps 1 and 2 but recursively $(S_1 \leq \ldots \leq S_k) < (T_1 \leq \ldots \leq T_k)$ lexicographically.
4. Equality is found in all above steps but *orientations* of subtrees do not match.

Isomorphism of triconnected component trees (continued)

- Isomorphism of two subtrees rooted at triconnected components G, H:
  1. Recall: For 3-connected graphs, use $canon(G, \rho, \{u, v\}, u)$ Thus $canon(G, \rho, \{a, b\}, *)$ has four possibilities: two choices of ρ and two choices for $u - a$ or b.
  2. Construct and compare all four canons of G and H. Eliminate the larger ones.
  3. Make recursive calls for children when corresponding virtual edge is traversed.
  4. If $canon(G, *, \{a, b\}, *) < canon(H, *, \{a', b'\}, *)$ then return $G < H$.
  5. If min. canon of G is $canon(G, \rho, \{a, b\}, a)$ then G is said to give *orientation* $a \rightarrow b$ to parent.
- Storage on stack: $O(1)$ bits for keeping track of yet un-eliminated canons.

Isomorphism of triconnected component trees (continued)

- Isomorphism of two subtrees rooted at separating pairs $\{a, b\}, \{a', b'\}$.
  1. Partition subtrees into blocks according to their sizes.
  2. Process blocks one by one: Compare the number of smallest children of S and T in a block, then next smallest and so on... . . . Further partitions a block into isomorphism classes.
  3. *Reference orientation* of $\{a, b\}$=Orientation given by majority of the children in the smallest isomorphism class.
  4. In each isomorphism class, count and compare the number of children that give reference orientation or the other orientation to parent.
- Storage on stack: Almost similar to Lindell's algorithm.

Theorem

Biconnected planar graph isomorphism is in Log-space.

□

An example

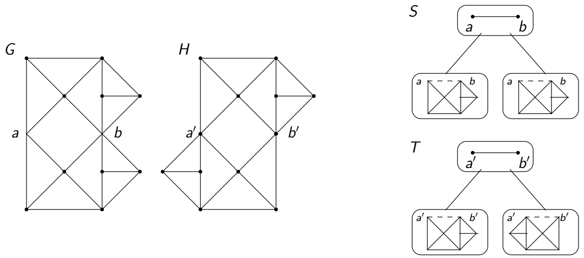


Figure: Example for comparison of orientations

Orientation count for S: $(2,0)$, for T: $(1,1)$. Thus $G \not\cong H$.

Step 5: Isomorphism of biconnected component trees

Always choose an articulation point as root of a biconnected component tree.
Similar approach as that of triconnected component trees.

- Simpler because there is no notion of orientation.
  Thus subtrees of S and T are pairwise equal $\Rightarrow S = T$.
- Difficulty:
  3-connected graph: $O(1)$ canons
  Biconnected graph: $O(n)$ canons
  ... as the canon depends on the choice of root.
- Recall: To get a log-space algorithm, we need
  1. Comparison counts be stored in log-space.
  2. Recurrence for stack space:
     $S(n) = S(\frac{n}{p}) + O(\log p) = O(\log n) \text{ for } p \geq 2$

  1 is satisfied as in Lindell's algorithm. How to get 2?

Isomorphism of biconnected component trees (continued)

S, T: Biconnected component trees.
$B \in S$: A biconnected component.
a: Parent of B in S (... articulation point)
S_B: Triconnected component tree of B.
Observations:

- S_B has a unique center R.
- Let b be a child of B. S_B contains many copies of a and b.
  If $b \in \text{a separating pair}$ (x, y) then
  $b \in \text{parent and all children of}(x, y)$
- The subgraph of S_B, $\{C \in S_B | b \in C\}$ is a subtree of S_B.
- Hence $\exists$ unique triconnected component C in S_B s.t. $b \in C$ and C is the nearest such component from R.
  Such a C is denoted by $C(b)$.

Isomorphism of biconnected component trees (continued)

Claim

Partition children of B into blocks $B_1, \ldots, B_r$ according to the sizes of their subtrees. It is possible to get $\#$ of choices of root $= O(min.\ block\text{-}size)$.

Proof idea:

- Separating pair $s(b)$ nearest to R on $C(b) \rightsquigarrow R$ path is a candidate choice of root.
- B_i be the smallest block, $|B_i| = k$. If $\exists b \in B_i$ s.t. $C(b) \neq R$ then at most $|B_i|$ choices for root i.e. $\{s(b) | b \in B_i\}$.
- If $\forall b \in B_i, C(b) = R$ then R has $O(|B_i|)$ automorphisms. Candidate roots: $O(|B_i|)$ neighbours of R.

Isomorphism of biconnected component trees (continued)

Thus, to compare subtrees rooted at $B \in S$ and $B' \in T$,

- Let $parent(B) = a, parent(B') = a'$. Colour copies of a, a' distinctly in B, B'.
- Pairwise construct and compare k possible canons of B and B' each.
- Recurse if children b, b' of B, B' in S and T are encountered simultaneously.
- Local storage: $O(\log |B|) = O(\log |B'|)$.
- Stack storage: $O(\log k)$ bits to remember id of the current canons.

$$\begin{aligned} \text{Stack space } S(n) &= S\left(\frac{n}{k}\right) + O(\log k) \\ &= O(\log n), \text{ for } k \geq 2 \end{aligned}$$

Open questions

- Improved algorithm for general graph isomorphism.
- Extending our algorithm to larger/different graph classes.

References

Eric Allender and Meena Mahajan.
The complexity of planarity testing.
In *STACS '00: Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science*, pages 87–98, 2000.

John E. Hopcroft and Robert E. Tarjan.
Dividing a graph into triconnected components.
SIAM Journal on Computing, 2(3):135–158, 1973.

Steven Lindell.
A logspace algorithm for tree canonization (extended abstract).
In *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 400–404, 1992.

Gary L. Miller and John H. Reif.
Parallel tree contraction part 2: further applications.
SIAM J. Comput., 20(6):1128–1147, 1991.

Omer Reingold.
Undirected st-connectivity in log-space.
In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 376–385, 2005.

Thomas Thierauf and Fabian Wagner.
The isomorphism problem for planar 3-connected graphs is in unambiguous logspace.
In *STACS*, pages 633–644, 2008.

H. Whitney.
A set of topological invariants for graphs.
American Journal of Mathematics, 55:235–321, 1933.

Thank You