Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Discrete Mathematics 31 (2008) 143–149
Electronic Notes in DISCRETE MATHEMATICS
www.elsevier.com/locate/endm

Planarity Algorithms via PQ-Trees
(Extended Abstract)

Bernhard Haeupler [1]

Department of Computer Science , Princeton University, Princeton NJ

Robert E. Tarjan [2]

Department of Computer Science , Princeton University, Princeton NJ
HP Laboratories, Palo Alto CA

In Memory of Shimon Even

Abstract

We give an abstract vertex-addition method for planarity testing that encompasses the algorithms of Lempel, Even, and Cederbaum, Shih and Hsu, and Boyer and Myrvold. The main difference between the former and the latter two is the order of vertex addition; the latter two differ only in implementation details. For the general method we give a direct proof of correctness that avoids the use of Kuratowski's theorem. We give a linear-time implementation that simplifies and unifies the Shih-Hsu and Boyer-Myrvold methods. Our algorithm extends to generate embeddings uniformly at random, to count embeddings, to represent all embeddings, and to produce a Kuratowski subgraph of a non-planar graph. Our algorithm keeps track of all possible embeddings by reinterpreting Booth and Lueker's PQ-tree data structure to represent circular instead of linear orders. This interpretation of PQ-trees gives the PC-trees of Shih and Hsu and leads to a simpler, more-symmetric form of PQ-tree reduction.

1571-0653/$ – see front matter
doi:10.1016/j.endm.2008.06.029

1 Introduction

The problem of testing a graph for planarity has a long and rich history; Hopcroft and Tarjan [7] gave the first linear-time algorithm, an implementation of the path-addition approach of Auslander and Parter. Earlier, Lempel, Even, and Cederbaum [9] gave a different planarity test that builds an embedding by adding vertices in a precomputed order, called an st-order. Booth and Lueker [1], acting on a suggestion of Tarjan, obtained a linear-time implementation of the Lempel-Even-Cederbaum algorithm by combining a linear-time st-ordering method [6] with their efficient PQ-tree reduction algorithm. They had independently developed the PQ-tree as a data structure to solve several ordering problems on matrices and graphs.

More recently, Shih and Hsu [10] and later but apparently independently Boyer and Myrvold [3,4] developed closely-related planarity tests and embedding algorithms that add vertices in postorder with respect to a depth-first spanning tree. Shih and Hsu's first paper [10] did not address important implementation issues, and subsequent papers [11,8] gave incorrect algorithms; Boyer et al. [2] finally gave a correct version of the Shih-Hsu algorithm.

2 Planarity Testing via Vertex Addition

Consider testing the planarity of a connected graph by building a planar embedding, adding a vertex at a time. At any step, each edge will be of one of three types: *embedded*, meaning both ends are embedded, *half-embedded*, meaning exactly one end is embedded, or *unembedded*, meaning neither end is embedded. Addition of a vertex converts some edges from half-embedded to embedded, and some other edges from unembedded to half-embedded. The half-embedded edges form the boundary between the embedded vertices and the unembedded vertices.

We would like to avoid backtracking. This requires some way of maintaining all possible embeddings of the half-embedded edges, as constrained by the embedded ones. In general the half-embedded edges can lie in many different faces of the partial embedding; keeping track of the possibilities seems difficult. As a first simplification, we require that the unembedded vertices induce a connected subgraph; that is, the vertex order is leaf-to-root on some spanning tree. Then, if the partial embedding can be extended to a complete embedding, the half-embedded edges lie in a common face, which we can take

[1] corresponding author: Bernhard Haeupler, haeupler@cs.princeton.edu

[2] The second author thanks Janet S. Yoon for contributions to an early phase of this work.

to be the outside face. The partial embedding in general consists of one or more connected components. These components partition the half-embedded edges. The half-embedded edges incident to each component are circularly ordered around the outside of the component. The set of all possible partial embeddings corresponds to a set of circular orders of half-embedded edges for each component.

Now consider the effect of adding a vertex. The half-embedded edges that become embedded as a result of this vertex addition are partitioned among the connected components existing before the vertex addition. These components are combined into a single component by the vertex addition. The effect of the vertex addition on the possible circular orders of the half-embedded edges is two-fold:

1. Reduce: For each connected component existing before the vertex addition, retain only those circular orders in which all the half-embedded edges that become embedded occur consecutively; if for some component there are no such orders, the graph is non-planar.
2. Combine: Form the set of circular orders of the new half-embedded edges as follows. For each connected component before the vertex addition, choose one of the circular orders remaining after reduction and delete from it all half-embedded edges that become embedded, to give a linear order. Catenate these linear orders, one from each connected component, with the new half-embedded edges incident to the newly added vertex (each of which forms a singleton linear order) to form a circular order. Doing this in all possible ways gives the set of circular orders for the component formed by the vertex addition.

We can prove the correctness of this abstract method by using an appropriate combinatorial corollary of the Jordan curve theorem. This is simpler than the alternative (used by Shih and Hsu and Boyer and Myrvold) of using Kuratowski's theorem (if the algorithm halts and declares the graph non-planar, then it contains a Kuratowski subgraph), which requires a tedious case analysis.

The PQ-tree data structure described in the next section allows efficient representation, reduction, and combination of sets of circular orders. This leads directly to linear-time implementation of the abstract method. But using PQ-trees further restricts the possible vertex orders of the planarity test, because PQ-trees can be combined efficiently only by attaching the root of one to an arbitrary node of the other. Restricting the method in this way leads to two natural vertex orders: st-order, the Lempel et al. strategy, or leaf-to-root order of a depth-first spanning tree, the Shih and Hsu and Boyer

and Myrvold strategy.

3 Circular Orders via PQ-trees

The *PQ-trees* of Booth and Lueker represent sets of permutations of a set, as follows. An *ordered tree* is a tree with a root, such that the children of every node are totally ordered. The *leaf order* of an ordered tree is the order in which the leaves are visited by a preorder traversal of the tree that visits the children of each node in the given total order. A PQ-tree is an ordered tree each of whose internal nodes is either a P-node or a Q-node. Two PQ-trees are *equivalent* if they are isomorphic up to arbitrary reordering of the children of P-nodes and reversal of the order of the children of Q-nodes. A PQ-tree represents the set of permutations of its leaves that are the leaf order of some equivalent tree. Given a PQ-tree T and a subset S of its leaves, a *reduction* of T on S modifies T to form a tree T' that represents the subset of the permutations represented by T such that the elements of S occur consecutively (but in arbitrary order). T is the null tree if there are no such permutations. Booth and Lueker gave an on-line algorithm for reduction that takes $O(n+m)$ time for a sequence of reductions of an n-node tree on sets of total size m.

We give a simple way to reinterpret PQ-trees to represent sets of circular orders instead of linear orders: Given a PQ-tree, we add a new root, whose only child is the original root. We call the new root the *special leaf*. (It is a leaf if we unroot the tree.) The *leaf order* of the augmented tree is the circular order of its leaves including the special leaf that begins with the special leaf, lists the other leaves in leaf order, and returns to the special leaf. The augmented tree represents the set of circular leaf orders of its equivalent trees, with equivalence defined as above. There is a one-one correspondence between leaf orders of trees equivalent to the original tree and leaf orders of trees equivalent to the augmented tree; all the properties of PQ-trees apply to the new interpretation. We also get a more symmetric view of reduction: given a subset S of the set of leaves, a reduction of the augmented tree modifies it to produce a new tree that represents the subset of circular orders in which all the elements of S occur consecutively. If S does not contain the special leaf, this corresponds to reduction of the original tree on S. If S does contain the special leaf, this corresponds to reduction of the original tree on the set of leaves that are not in S, which we call *complement reduction*. The Booth-Lueker reduction algorithm extends to the new interpretation of PQ-trees, and in particular gives an efficient implementation of complement reduction, for which some slight simplification is possible. A reduction can be done purely bottom-up.

A global view of reduction given in the Master's thesis of Young [12] avoids the template-based description of Booth and Lueker, which has many cases.

Augmented PQ-trees representing sets of circular orders correspond to the unrooted PC-trees of Shih and Hsu [10,11], if we ignore the root. A root seems to be required for efficient implementation of reduction, however. We view the implementation of PC-trees by Hsu and McConnell [8] as just a reinvention of PQ-trees and PQ-tree reduction, with the more symmetric view of reduction given by considering circular orders instead of linear orders and the global view of reduction given previously by Young.

The Shih-Hsu and Boyer-Myrvold planarity-testing methods in effect use complement reduction. Boyer and Myrvold describe their reduction process as operating on a representation of the partially embedded graph rather than on a separate data structure, but the computation is still an implementation of complement reduction: their pointer structure can be interpreted as representing a PQ-tree. Their reduction method walks down the tree and reduces it node-by-node, which seems to limit the method to handling complement reduction, and requires extra pointers, to support both bottom-up and top-down walks on the tree.

4 A Simple Linear-Time Implementation

Our linear-time implementation of the abstract planarity testing method does a depth-first search on every connected component of the input graph, testing planarity on the fly using PQ-trees. For each ancestor v in the depth-first spanning tree of the current vertex of the search it maintains a PQ-tree and a set of PQ-tree leaves $S(v)$:

• When advancing along a tree arc (v, w), construct a PQ-tree for w consisting of a root that is the special leaf and represents (v, w), with a single child that is a P-node and represents w. Initialize $S(v)$ to contain just the special leaf.

• When advancing along a back arc (v, w), add a leaf child representing (v, w) to the P-node for v, and add this leaf to $S(w)$.

• When retreating along a tree arc (v, w), reduce the PQ-tree for w to represent the circular orders such that the leaves in $S(v)$ are consecutive. If this produces the null tree, stop and declare the graph non-planar. Otherwise, delete from the reduced tree all the leaves in $S(v)$, simplify the tree by deleting P-nodes and Q-nodes with no children and repeating until all P-nodes and Q-nodes have children, and do the appropriate part of the combining step by making the root of the simplified tree a child of the P-node for v, unless v is the start vertex of the depth-first search.

This simple algorithm works for arbitrary graphs possibly having multiple edges, loops and more than one (bi-)connected component. Its proof of correctness follows easily from the correctness of the abstract planarity method. We can avoid storing the leaves and roots of the PQ-trees explicitly, at the cost of making the implementation of PQ-tree reduction a little more complicated.

5 Remarks

Our algorithm extends in various ways. If instead of deleting nodes during reduction and simplification, we remove and save the subtrees containing these nodes, we get a linear-time algorithm that produces a representation of all possible embeddings. We can also generate a single embedding, either arbitrary or uniformly randomly chosen, or count embeddings. An alternative way to view the vertex-embedding process is as the time reversal of a process of contracting a graph to a single vertex by contracting edges, which might offer some conceptual advantages.

Our interpretation of PQ-trees as representing sets of either linear or circular orders allows us to efficiently solve ordering problems with mixed circular and linear constraints.

We see at least two possible directions for further research, one concerning planarity-testing, the other concerning PQ-trees. Is there a way to unify and simplify path-addition planarity tests such as those of Hopcroft and Tarjan and Fraysseix, Mendez and Rosenstiehl [5]? Is there a way to obtain the path-addition and vertex-addition methods as different versions of a more-general approach? Is there an efficient way to re-root a PQ-tree and so allow more general ways to combine such trees efficiently? This would allow a more-general strategy for planarity-testing to run in linear time.

References

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

[2] Boyer, J., C. Fernandas, A. Noma and J. de Pina, *Lempel, Even, and Cederbaum Planarity Method*, Lecture Notes in Computer Science **3059** (2004), pp. 129–144.

[3] Boyer, J. and W. Myrvold, *Stop minding your P's and Q's: a simplified O (n) planar embedding algorithm*, Proceedings of the 10th annual ACM-SIAM

symposium on Discrete algorithms (1999), pp. 140–146.

[4] Boyer, J. and W. Myrvold, *On the Cutting Edge: Simplified O (n) Planarity by Edge Addition*, Journal of Graph Algorithms and Applications **8** (2004), pp. 241–273.

[5] de Fraysseix, H., P. de Mendez and P. Rosenstiehl, *Trémaux trees and planarity*, International Journal of Foundations of Computer Science **17** (2006), p. 1017.

[6] Even, S. and R. Tarjan, *Computing an st-numbering*, Theoretical Computer Science **2** (1976), pp. 339–344.

[7] Hopcroft, J. and R. Tarjan, *Efficient planarity testing*, Journal of the ACM **21** (1974), pp. 549–568.

[8] Hsu, W. and R. McConnell, *PC trees and circular-ones arrangements*, Theoretical Computer Science **296** (2003), pp. 99–116.

[9] Lempel, A., S. Even and I. Cederbaum, *An algorithm for planarity testing of graphs*, in: Rosentiehl, P., editor, *Theory of Graphs: International Symposium* (1967), pp. 215–232.

[10] Shih, W. and W. Hsu, *A simple test for planar graphs*, Proceedings of the International Workshop on Discrete Mathematics and Algorithms (1993), pp. 110–122.

[11] Shih, W. and W. Hsu, *A new planarity test*, Theoretical Computer Science **223** (1999), pp. 179–191.

[12] Young, S., "Implementation of PQ-Tree Algorithms," Master's thesis, University of Washington (1977).