Additional PC-Tree Planarity Conditions

Conference Paper *in* Lecture Notes in Computer Science · September 2004

DOI: 10.1007/978-3-540-31843-9_10

CITATIONS
4

READS
51

1 author:

John Michael Boyer
Independent

78 PUBLICATIONS **759** CITATIONS

SEE PROFILE

Additional PC-Tree Planarity Conditions

John M. Boyer

PureEdge Solutions Inc. Victoria, BC Canada
jboyer@acm.org, jboyer@PureEdge.com

Abstract. Recent research efforts have produced new algorithms for solving planarity-related problems. One such method performs vertex addition using the PC-tree data structure, which is similar to but simpler than the well-known PQ-tree. For each vertex, the PC-tree is first checked to see if the new vertex can be added without violating certain planarity conditions; if the conditions hold, the PC-tree is adjusted to add the new vertex and processing continues. The full set of planarity conditions are required for a PC-tree planarity tester to report only planar graphs as planar. This paper provides further analyses and new planarity conditions needed to produce a correct planarity algorithm with a PC-tree.

1 Introduction

The first linear-time planarity tests [1, 2] represent significant achievements but are also quite complex. Recent research has produced simpler linear-time planarity algorithms [3–5]. This paper discusses the planarity method of Shih and Hsu [5], which is based on a data structure called a PC-tree. The PC-tree method is a *vertex addition* method that adds each vertex to a *partial planar embedding* once it determines that planarity can be preserved while adding the vertex and all edges that connect it to other vertices in the partial embedding.

The PC-tree method processes the vertices in a post-order traversal of the depth first search (DFS) tree of the graph. Thus, there is a path of unprocessed vertices from every vertex to the root of the DFS-tree. If the graph is planar, then it must be possible to embed of the first k vertices so that all vertices with direct back edge connections to their unprocessed DFS ancestors are on the external face of the partial embedding. For each vertex i, the algorithm first checks the PC-tree for a number of defined planarity conditions. If all conditions are met, then a planarity reduction is applied to the PC-tree for vertex i.

If one or more planarity conditions were missing, then a planarity reduction would be applied when it should not be, ultimately causing a planar result to be reported on some non-planar graphs. The literature on PC-trees have not presented additional planarity conditions, instead focusing on the consecutive ones problem [6, 7] or on equating PQ-tree and PC-tree reductions [8]. A submitted book chapter [9] presents an alternate graph-theoretic view that shows the correctness of the general approach, but it uses constructs that are difficult to apply directly to a PC-tree. This paper presents the additional required

J. Pach (Ed.): GD 2004, LNCS 3383, pp. 82–88, 2004.

planarity conditions that arise directly on the PC-tree, thus allowing more reasonable comparisons of complexity and empirical performance to be made with other planarity methods. In particular, although it is reasonable to assume that the 'batch' operations of vertex addition methods are more cumbersome to implement and less efficient than a finer grain edge addition method [4], proper comparisons cannot be done with only the planarity conditions in [5].

Section 2 provides some definitions and preliminary remarks. Sections 3, 4 and 5 present additional planarity conditions and further analyses for the PC-tree. Finally, Section 6 presents some concluding remarks.

2 Preliminaries

A PC-tree represents a partial planar embedding of a graph, with C-nodes representing all biconnected components and P-nodes representing cut vertices in the partial embedding and vertices with direct back edge connections that have not been embedded yet. Every P-node is associated with a vertex of the input graph. The neighbors of a C-node are P-nodes, which form the *representative bounding cycle* (RBC) of the C-node. The RBC corresponds to the external face cycle of the biconnected component represented by the C-node (for efficiency, nodes are removed from the RBC if they represent neither cut vertices in the partial embedding nor the endpoints of unembedded back edges). The P-nodes of the RBC are connected into a cycle. Traversal through a C-node occurs on one of the two paths along the RBC cycle between two neighbors of the C-node.

The PC-tree is denoted T, and T_r denotes a subtree of T rooted by node r. The current vertex being processed is denoted i. An *i-subtree* T_w is a PC-subtree of T_i that is rooted by the P-node for w with lowpoint(w) equal to i (i.e. the unembedded back edges from w and its descendants connect to i). An *i^*-subtree* T_x is a PC-subtree of T_i that is rooted by the P-node for x with lowpoint(x) $< i$ and that contains no vertex adjacent to i in the input graph (so, every unembedded back edge connects to an ancestor of i). To simplify discussion, the direct back edges to i and its ancestors are considered to be degenerate i-subtrees and i^*-subtrees. If the root node of an i-subtree or i^*-subtree is the child of a given PC-tree node, then we say that the i-subtree or i^*-subtree is a child of that node. A *terminal node* is a P-node or C-node of the PC-tree that has one or more i-subtree children, one or more i^*-subtree children, and no descendants in the PC-tree with both i-subtree and i^*-subtree children.

For each vertex i (in post-order of the DFS tree), the PC-tree is tested for planarity conditions before adding i to the partial embedding. Because three or more terminal nodes implies non-planarity, much of the discourse in [5] focuses on the one or two terminal node cases. Shih and Hsu present four necessary conditions for maintaining planarity in the one and two terminal node cases: "In Lemma 2.5, Corollary 2.6, [and] Lemmas 3.1 and 3.2 we made the assumption that graph G is planar in deriving at those conclusions. We shall show that if these conclusions hold at each iteration, then G must be planar by showing that these conditions imply a feasible internal embedding for each 2-connected

component." [5, p. 188]. Then, the proof presented only describes how to perform the one and two terminal node planarity reductions, which does not prove that those reductions can always be performed if only the given planarity conditions are met. The remaining sections describe additional required planarity conditions and indicate how their violation implies non-planarity.

3 The *i*-*i** Subtree Patterns Around a Terminal *C*-Node

Lemma 3.2 of [5] seeks to characterize the allowable pattern of child i-subtrees and i^*-subtrees around a terminal C-node. It states that for the root j of any child i-subtree of a terminal C-node, one of the two RBC paths from j to the parent of the C-node must contain only i-subtrees. This condition is *necessary* but only *sufficient* in the one terminal node case when the terminal node has no proper ancestor with a child i^*-subtree. In the two terminal node case and the one terminal node case where the terminal node has a proper ancestor with a child i^*-subtree, it is possible to be compliant with the statement of the lemma yet still have a non-planarity condition. Theorem 1 states the additional restriction required on terminal C-nodes, and Figure 1 shows PC-trees that violate the restriction, along with the resulting $K_{3,3}$ minor.

Theorem 1. *If a terminal C-node c has a proper ancestor r with a child v such that T_v excludes c and is or contains an i^*-subtree, then c must have a child w for which an RBC path from w to the parent of c contains all of the i-subtree children of c.*

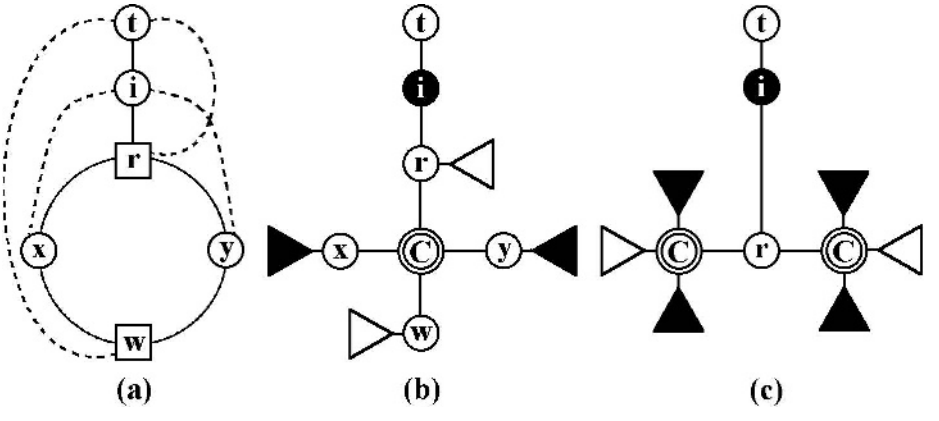

Fig. 1. (a) A $K_{3,3}$ non-planarity minor from [3]. (b) A corresponding PC-tree with one terminal C-node having the forbidden pattern of i-subtrees (dark triangles) and i^*-subtrees (light triangles). (c) An example with two terminal C-nodes, only one of which need be in the depicted state. Note the use of graph minors for simplification; in these examples, r could be any node on the path between i and the terminal C-node.

4 The i-i^* Subtree Patterns for an Intermediate C-Node

Given an intermediate C-node c along the path P between two terminal nodes, we consider the two RBC paths strictly between neighbors v and v' of c in P. The proof of Lemma 3.1 of [5] attempts to prove the following: neither RBC path of c can contain both an i-subtree and an i^*-subtree. It does not show the necessity of the broader planarity condition stated the lemma: of the two RBC paths strictly between v and v', one must contain only child i-subtrees and the other must contain only child i^*-subtrees. There are four issues. First, the proof of the simpler condition fails when the terminal node and the intermediate C-node are neighbors; the author has found other $K_{3,3}$ patterns (not depicted) for this case. Second, the proof is by contradiction but does not fully negate the lemma statement: the simpler condition (described above) can hold while still violating the lemma statement's planarity condition if both RBC paths contain only i^*-subtrees (see Figure 2(a)) or i-subtrees (reduces to Figure 1(c)). Third, stricter conditions are required if the intermediate C-node is m, the closest common ancestor of the terminal nodes, because it cannot be flipped. The graph is non-planar if an i-subtree appears below P on the RBC of m (see Figure 2(b)) or if an i^*-subtree appears on the RBC of m above P (resulting in a $K_{3,3}$ that edge contracts to the K_5 minor of Figure 3(a)). Theorem 2 states the required planarity conditions. A fourth problem is that analogous planarity conditions are required for one terminal node, producing the same non-planarity minors except the last case does not edge contract to a K_5 minor but still produces a $K_{3,3}$ (not depicted). Theorem 3 states the additional planarity conditions.

Theorem 2. *Let P denote the path between two terminal nodes u and u' with closest common ancestor m. Let c denote a C-node in $P - \{u, u'\}$ with neighbors v and v' in P. Of the two RBC paths strictly between v and v', one must contain no child i-subtrees and the other no child i^*-subtrees. Further, if $c = m$, then the RBC path containing the child i-subtrees must also contain the parent of c.*

Theorem 3. *Given one terminal node u, let P denote the path from u to the farthest ancestor u' with a child i^*-subtree. Let c be an intermediate C-node in path $P - \{u\}$. For $c \neq u'$, let v and v' denote the neighbors of c in P. For $c = u'$, let v denote the neighbor of c in P and let v' denote the closest child i^*-subtree along either RBC path from the parent p of c. The following conditions must hold: 1) The children of c in one RBC path strictly between v and v' must contain only child i-subtrees; 2) The opposing RBC path strictly between v and v' must contain only child i^*-subtrees; 3) If $c = u'$, then the RBC path containing the child i-subtrees must also contain p.*

5 Finding Non-planarity of $K_{3,3}$-Less Graphs

Consider extending Lemma 2.5 in [5] to a PC-tree that contains C-nodes. Specifically, suppose the closest common ancestor m of the two terminal nodes is a C-node whose parent has the only child i^*-subtree along the path P'. Figure 3

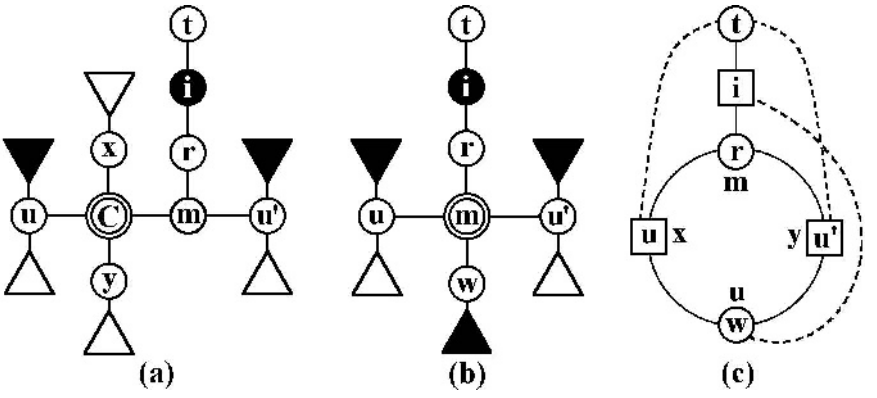

Fig. 2. (a) The intermediate C-node has child i^*-subtrees along both RBC paths between its parent (labelled m here) and the next node in path P (labelled u here). (b) The C-node labelled m has a child i-subtree below the path P between the terminal nodes. (c) The $K_{3,3}$ for these planarity condition violations; the labels m, x, y and u are the mapping for part (a), and the labels r, u, u' and w are for part (b).

depicts an example PC-tree and the corresponding K_5 minor pattern from [3]. In this case, the $K_{3,3}$ shown in the proof of Lemma 2.5 in [5] cannot be found, illustrating that the proof does not "go through for the case of general trees without any changes provided that the paths through a C-node are interpreted correctly" [5, p. 185]. Theorem 4 states the relevant planarity condition from Lemma 2.5 of [5], relying for its proof of necessity on both [5] and Figure 3.

Theorem 4. *Suppose there are two terminal nodes u and u' in T_r, and let m be their closest common ancestor. Let P' be the unique path from m to r. Every proper ancestor of m in T_r must have no child i^*-subtrees.*

This case is also important because it shows the method by which K_5 subdivisions and other $K_{3,3}$-less graphs are found by the PC-tree algorithm. In [5], the case of three terminal nodes is shown to produce a either a (subgraph homeomorphic to) $K_{3,3}$, or "we could have three terminal nodes being neighbors of a C-node, in which case we would get a subgraph homeomorphic to K_5 ..." Technically, the result is a K_5 minor, which could produce a subgraph homeomorphic to $K_{3,3}$ or K_5. Of greater importance, though, is the fact that a $K_{3,3}$ can also *always* be found in this case (though not the same one indicated for the three terminal node case in [5]). However, this case of three terminal node neighbors of a C-node is the only case mentioned in [5] for finding a K_5, yet there are many non-planar graphs that do not contain a $K_{3,3}$. Therefore, there must be *some other* condition that detects non-planarity for graphs that contain a K_5 but not a $K_{3,3}$ (e.g. all K_5 subdivisions). The K_5 in Figure 6 of [5] is equivalent to Figure 3(b). It does not result in three terminal nodes as stated in [5], but is instead discovered by violation of the planarity condition in Theorem 4.

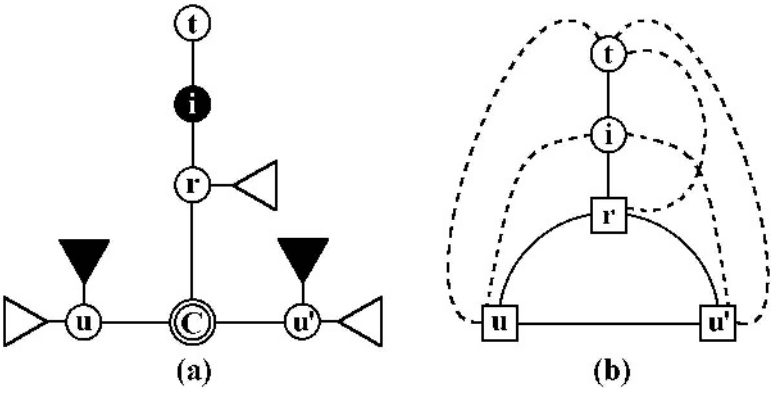

Fig. 3. (a) A PC-tree in which the closest common ancestor of terminal nodes u and u' is a C-node with a proper ancestor that has a child i^*-subtree. (b) The corresponding K_5 minor from [3]. Note: Due to the difference in definitions between graph minors and subgraph homeomorphism, this case implies a subgraph homeomorphic to $K_{3,3}$ or K_5.

6 Conclusion and Future Work

This paper presented the additional planarity conditions required to create a correct planarity algorithm using a PC-tree, allowing fair comparison with other recent approaches to planarity. While the August 2003 version of the implementation in [10] could not be empirically compared due to frequent incorrect results, Hsu also requested that a subsequent version with fixes not be empirically compared as he felt the implementation was only a proof of concept. However, there is strong evidence from [11] that a simplified vertex addition method can achieve far better performance than most prior methods, although those results also suggest that the edge addition methods in [4, 12] are faster. Future work must use the results of this paper to create correct, efficient PC-tree implementations for empirical comparisons, especially with [4, 12]. The results of this paper are also important for creating a Kuratowski subgraph isolator based on the PC-tree, the full exposition of which should translate from the graph minors used to express planarity condition violations to homeomorphic subgraphs.

References

1. Hopcroft, J., Tarjan, R.: Efficient planarity testing. Journal of the Association for Computing Machinery **21** (1974) 549–568
2. Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ–tree algorithms. Journal of Computer and Systems Sciences **13** (1976) 335–379
3. Boyer, J., Myrvold, W.: Stop minding your P's and Q's: A simplified $O(n)$ planar embedding algorithm. Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (1999) 140–146

4. Boyer, J., Myrvold, W.: On the cutting edge: Simplified $O(n)$ planarity by edge addition. Accepted to Journal of Graph Algorithms and Applications, August 2004 (Preprint at http://www.pacificcoast.net/~lightning/planarity.ps) 1–29
5. Shih, W.K., Hsu, W.L.: A new planarity test. Theoretical Computer Science **223** (1999) 179–191
6. Hsu, W.L.: A simple test for the consecutive ones property. Journal of Algorithms **42** (2002) 1–16
7. Hsu, W.L., McConnell, R.: PC-trees and circular-ones arrangements. Theoretical Computer Science **296** (2003) 59–74
8. Hsu, W.L.: PC-trees vs. PQ-trees. Lecture Notes in Computer Science **2108** (2001) 207–217
9. Hsu, W.L., McConnell, R.: PC-trees. submitted to *Handbook of Data Structures and Applications*, Dinesh P Mehta and Sartaj Sahni ed. (2004)
10. Hsu, W.L.: An efficient implementation of the PC-trees algorithm of shih and hsu's planarity test. Technical Report TR-IIS-03-015, Institute of Information Science, Academia Sinica (2003)
11. Boyer, J.M., Cortese, P.F., Patrignani, M., Di Battista, G.: Stop minding your P's and Q's: Implementing a fast and simple DFS-based planarity testing and embedding algorithm. In Liotta, G., ed.: Proceedings of the 11th International Symposium on Graph Drawing 2003. Volume 2912 of Lecture Notes in Computer Science., Springer-Verlag (2004) 25–36
12. de Fraysseix, H., Rosenstiehl, P.: A characterization of planar graphs by trémaux orders. Combinatorica **5** (1985) 127–135