A Planarity Test via Construction Sequences

Jens M. Schmidt

Max Planck Institute for Informatics, Germany

Abstract. Linear-time algorithms for testing the planarity of a graph are well known for over 35 years. However, these algorithms are quite involved and recent publications still try to give simpler linear-time tests. We give a conceptually simple reduction from planarity testing to the problem of computing a certain construction of a 3-connected graph. This implies a linear-time planarity test. Our approach is radically different from all previous linear-time planarity tests; as key concept, we maintain a planar embedding that is 3-connected at each point in time. The algorithm computes a planar embedding if the input graph is planar and a Kuratowski-subdivision otherwise.

1 Introduction

Testing the planarity of a graph is a fundamental algorithmic problem that has initiated significant contributions to data structures and the design of algorithms in the past. Although optimal linear-time algorithms for this problem are known for over 35 years [13,3], they are involved and recent publications still try to give simpler linear-time algorithms [4,6,9,11,25].

We give a linear-time planarity test that is based on a conceptually very simple reduction to the problem of computing a certain construction C of a 3-connected graph G (we will give a precise definition of C in Section 3). The existence of a similar construction has also been used by Kelmans [14] and Thomassen [27] to give a short proof of Kuratowski's Theorem. Although their proof itself is constructive (in the sense that it gives a polynomial-time planarity test) and received much attention in graph theory due to its simplicity, it has not been utilized algorithmically. We give the first linear-time planarity test that captures this proof scheme. Our hope is that this new approach will lead to simple planarity tests, just as the same concept led to simple proofs of Kuratowski's Theorem.

Currently, the fastest algorithm known for computing C achieves a linear running time [24], but is quite involved. For that reason, our reduction does not qualify to be regarded as a simple planarity test yet. However, every simplification made for computing C will immediately result in a simpler linear-time planarity test. In fact, much less is needed, as our reduction relies only on the part of the construction C until a first non-planar graph occurs; thus, one may assume planarity for computing the necessary part of C. In the case that a quadratic running time is allowed, a very simple algorithm that computes C (and, thus, planarity) is known [23].

Recent planarity tests like [4,6,9,11,25] maintain a planar embedding at each step, where all steps either add paths/edges (*path addition method*) or vertices (*vertex addition method*) to the embedding; for a thorough survey on planarity tests, we refer to Patrignani [22]. In our algorithm, each step will essentially add an edge, possibly after subdividing one or two edges in advance. Unlike all previous linear-time planarity tests, we maintain a planar embedding that is always 3-connected. This is a key concept for the following reason. The 3-connectivity constraint fixes the planar embedding (up to flipping), which will allow to test efficiently whether the addition of a next edge e preserves planarity.

A well-known connection between 3-connectivity and planarity is that both can be characterized by conditions on *segments* (or *components*) of cycles [30]. In fact, the first linear-time tests on planarity and 3-connectivity [12,13] due to Hopcroft and Tarjan use such conditions. A detailed exposition of the connection was given later by Vo and Williamson [29,30,33], respectively, with an emphasis on explaining the algorithms in [12,13]. Nevertheless, the precise interplay between 3-connectivity and planarity and its algorithmic consequences are still far from understood; e. g., the known connection suggests to ask whether there is a general approach that combines linear-time 3-connectivity and planarity tests. Our reduction makes a first step towards such a general approach. The combining element is C; on the one hand, C proves a graph to be 3-connected, on the other hand, C provides a unique embedding as long as the constructed graphs are planar, which allows to check planarity efficiently.

A planarity test is *certifying* in the sense of [16] if its *yes*/*no*-output is augmented with a planar embedding if the input graph is planar and a Kuratowski-subdivision otherwise. The first two linear-time planarity tests of Hopcroft and Tarjan [13] and Booth and Lueker [3] did not give a planar embedding for planar input graphs. Mehlhorn and Mutzel [18] and Chiba, Nishizeki, Abe and Ozawa [7] extended these tests to compute a planar embedding in the same asymptotic running time. The algorithm presented here is certifying.

2 Preliminaries

We use standard graph-theoretic terminology from [2]. Let $G = (V, E)$ be a simple finite graph with $n := |V|$ and $m := |E|$. Multiedges do not matter for planarity and can be removed in advance by performing two bucket sorts on the endpoints of edges in E.

A vertex whose deletion increases the number of connected components is called a *cut vertex*. A graph G is *biconnected* if it is connected and contains no cut vertex. A *biconnected component* of a graph G is a maximal biconnected subgraph of G. A pair of vertices whose deletion disconnects a biconnected graph is called a *separation pair*. A biconnected graph is *triconnected* if it contains no separation pair.

A *(straight-line) planar embedding* of a graph $G = (V, E)$ is an injective function $\pi\colon V \to \mathbb{R}^2$ such that for any two distinct edges ab and cd the straight line segments $\overline{\pi(a)\pi(b)}$ and $\overline{\pi(c)\pi(d)}$ are internally disjoint (i. e., they may only intersect at their endpoints). Two embeddings Emb_1 and Emb_2 of the same planar graph are *(combinatorially) different* if there is a vertex v such that the cyclic order of edges around v in Emb_1 and Emb_2 is different.

A *subdivision* of a graph G (a *G-subdivision*) is a graph obtained by replacing the edges of G with internally disjoint paths of length at least one. Triconnected graphs and their subdivisions have the following property, which we will use throughout this paper.

Lemma 1 (Whitney [32], Thm. 1.1 in [20]). *Every subdivision of a triconnected graph has a unique planar embedding (up to flipping).*

The *triconnected components* of a graph G are obtained by the following recursive process on every biconnected component H of G: As long as there is a separation pair $\{x, y\}$ in H, we split H into two subgraphs H_1 and H_2 that partition $E(H)$ and have only x and y in common, followed by adding the edge $e = xy$ to both H_1 and H_2. We refer to [10] for a precise definition of this process. The edge e that was added to H_1 (respectively, H_2) is called the *virtual edge* of H_1 (H_2) and can be seen as a replacement of the graph H_2 (H_1) in this decomposition.

The graphs resulting from this process are either sets of three parallel edges (*triple-bonds*), triangles or simple triconnected graphs. To obtain the triconnected components of G, triple-bonds containing a common virtual edge are successively merged to maximal sets of parallel edges (*bonds*); similarly, triangles containing a common virtual edge are successively merged to maximal cycles (*polygons*). Thus, a triconnected component of G is either a bond, a polygon, or a simple triconnected graph. The triconnected components form a tree, which is called *SPQR-tree* of G [10].

It is well-known that a graph G is planar if and only if all its biconnected components are planar [13]. A similar result holds for the triconnected components of G: If G is planar, all triconnected components of G are planar, as every triconnected component is a minor of G. Conversely, if all triconnected components of a graph G are planar, we can successively merge the planar embeddings of two triconnected components containing the same virtual edge to a bigger planar embedding [31, Lemma 6.2.6], and obtain a planar embedding for G in linear time. This gives the following result.

Lemma 2 ([17]). *A graph is planar if and only if all its triconnected components are planar.*

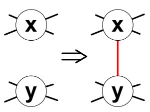
As bonds and cycles are planar, planarity has only to be checked for simple triconnected graphs. The triconnected components can be computed in linear time [10,12]. Although this is not a trivial task, reliable implementations are publicly available [21] and future steps will benefit greatly from this.

3 Constructions of Triconnected Graphs

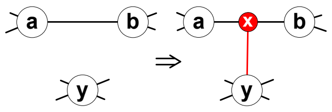
With the above arguments we can assume that the input graph G is simple and triconnected. We will make use of a special construction of triconnected graphs due to Barnette and Grünbaum [1].

Definition 1. *Let G be a simple triconnected graph with $n \geq 4$. We define the following operations on G (all vertices and edges are assumed to be distinct; see Figure 1).*

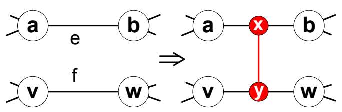
(a) Add an edge xy between two non-adjacent vertices x and y.
(b) Subdivide an edge ab by a vertex x and add the edge xy for a vertex $y \notin \{a, b\}$.
(c) Subdivide two non-parallel edges e and f by vertices x and y, respectively, and add the edge xy (note that e and f may intersect in one vertex).
(d) Add a new vertex x and join it to exactly three old vertices a, b and c.



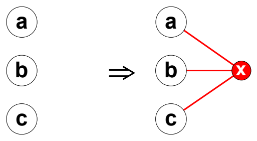
(a) x and y non-adjacent



(b) $y \notin \{a, b\}$



(c) e and f are not parallel edges



(d) adding a *claw* with center x

Fig. 1. Operations on triconnected graphs

Operations 1a-c correspond to *adding edges* (the *added* edge is xy) while Operation 1d corresponds to *adding a claw* (i. e., $K_{1,3}$) with a designated *center vertex* x. The *attachments* of an operation O on G are the vertices and edges in G involved in the operation, i. e., the attachments of Operations 1a–d are $\{x, y\}$, $\{ab, y\}$, $\{ab, vw\}$ and $\{a, b, c\}$, respectively. Let *suppressing* a vertex x with exactly two non-adjacent neighbors y and z be the operation of deleting x and adding the edge yz.

Applying any of the Operations 1a–d to G generates a graph that is simple and triconnected again. A classical result of Barnette and Grünbaum [1] and Tutte [28] characterizes the triconnected graphs in terms of the first three operations.

Theorem 1 (**[1,28]**)**.** *A simple graph G with $n \geq 4$ is triconnected if and only if G can be constructed from K_4 using only operations of Types 1a–c.*

For testing planarity, we will use the following slightly modified construction. It restricts operations of Type 1a to be at the end of the construction; in order to achieve this, we allow to use additional operations of Type 1d. Having all Type 1a operations at the end of the construction will allow for an easier efficient data structure in the planarity test.

Theorem 2 ([23]). *A simple graph G with $n \geq 4$ is triconnected if and only if G can be constructed from K_4 using only operations of Types 1a–d such that all operations of Type 1a are applied last.*

A *construction sequence* C of G is a sequence of operations that constructs G from K_4 precisely as stated in Theorem 2. Note that any edge that was added by an Operation 1a in C will not be subdivided by later operations.

A construction sequence has a space complexity that is linear in the size of G by using a labeling scheme on vertices and edges that essentially assigns a new label to one half of an edge e after e was subdivided by an operation [23]. The labeling scheme allows additionally for a constant-time access to the edges and vertices that are involved in an operation O, i. e., to the edge e that is added by O and to the vertices and edges on which the endpoints of e lie.

Recently, it was shown that a construction C' of G as stated in Theorem 1 can be computed in linear time [24]. The algorithm is certifying and hence a reliable implementation has already been made publicly available [19]. A construction sequence C can be obtained from C' by a simple linear-time transformation as pointed out in [23].

4 The Planarity Test

We can assume that the input graph G is simple and triconnected. Observe that if $n \leq 3$, G is planar. Assume $n \geq 4$. Let C be a construction sequence of G.

The planarity test starts with the (unique) planar embedding of K_4 and computes iteratively a planar embedding for the graph that is obtained from the next operation O in C if possible. The following lemma characterizes under which conditions an operation O in C preserves planarity.

Lemma 3. *Let H be a planar embedding of a simple triconnected graph on at least 4 vertices and let H' be the graph that is obtained from H by applying an operation O of Type 1a–d. Then H' is planar if and only if the attachments of O are part of one face f of H.*

Proof. $\Leftarrow$: Clearly, subdividing edges in the facial cycle of f preserves planarity and so does the addition of an edge or a claw inside f.

$\Rightarrow$: Assume to the contrary that not all attachments of O are contained in one face of H. Note that when O is of Type 1d it is possible that every two of the three attachments are contained in a face of H, respectively. We will show a contradiction to the unique embedding of H. Let *Emb* be the planar embedding that is obtained from the planar embedding of H' by reversing Operation O,

i. e., by deleting the added edge in H' and suppressing all vertices of degree two if O is of Type 1a–c, and by deleting the center vertex of the added claw in H' if O is of Type 1d. Then *Emb* and H embed the same simple triconnected graph, but are combinatorially different, as *Emb* has a face containing all attachments of O, while H has no such face by assumption. This contradicts Lemma 1. □

If all attachments of O are in one face f, applying O gives a planar embedding. If all operations in C satisfy this condition, we obtain a planar embedding of G. Otherwise, let H be the graph obtained from the first operation in C that does not satisfy the condition of Lemma 3. Then H is non-planar and G must be non-planar, as G contains a subdivision of H as a subgraph. We will show how to extract this subdivision in linear time in the next section. Lemma 3 suggests the following Algorithm 1.

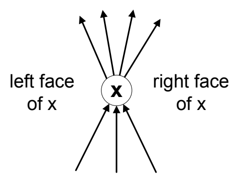
Algorithm 1 PlanarityTest(G) ▷ G simple and triconnected with $n \geq 4$

```
1: compute a construction sequence $C = O_1, \ldots, O_k$ of $G$
2: initialize the (unique) planar embedding $H$ of $K_4$
3: for $i = 1$ to $k$ do
4:     if all attachments of $O_i$ are in one face $f$ of $H$ then    ▷ planar
5:         apply $O_i$ to $H$ by adding the edge or claw inside $f$
6:     else                                                          ▷ non-planar
7:         compute a Kuratowski-subdivision
```

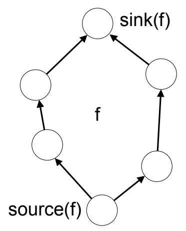
It remains to discuss how the condition in Lemma 3 can be checked efficiently for every operation in C.

A *plane st-graph* is an embedding of a planar directed acyclic graph with exactly one source s and exactly one sink t such that s and t are contained in the external face of the embedding. It is well-known that every biconnected planar graph can be oriented and drawn as plane *st*-graph (see, e. g., [5, Lemmas 1+2]). In every step of Algorithm 1, the planar embedding H is triconnected and thus biconnected. To check the condition in Lemma 3 efficiently, we will maintain H as plane *st*-graph and use a data structure that is able to answer queries whether edges and vertices are contained in the same face of H in amortized constant time.

We modify a data structure due to Djidjev [8, Lemma 3.1], which runs on a standard word-RAM. The original data structure maintains a plane *st*-graph H in which the incoming and the outgoing edges for any vertex x appear consecutively around x; hence, the boundary of each face f in H consists of two oriented paths from a common start vertex (the *source* of f) to a common end vertex (the *sink* of f); see [26]. Note that every vertex is source or sink of at least one face, as H has minimum degree 3. Additionally, every vertex $x \notin \{s, t\}$ is contained in exactly two faces for which x is neither source nor sink; we call these faces the *left* and the *right* face of x, respectively (see Figure 2). The data structure maintains pointers to the source and sink for each face in H and a pointer from



(a) *left* and *right* face of a vertex



(b) *source* and *sink* vertex of a face

Fig. 2. Plane *st*-graphs

each vertex $x \notin \{s,t\}$ to its left and right face. The following two queries for triconnected graphs H are supported by performing simple tests along the above pointer structure.

(1) Given a vertex a and an edge b of H, output a face of H that contains a and b or report that there is none.
(2) Given two vertices a and b of H such that a is a source or sink of at most 11 faces, output a face of H that contains a and b or report that there is none.

Each of these queries takes worst-case time $O(1)$. Note that the only query for which we cannot expect a constant running time using the above pointer structure would be a query where a and b are source or sink vertices of an unbounded number of faces. That is why query (2) assumes only a constant number of such faces. We augment the data structure by the following query type and show that each such query can be computed in worst-case time $O(1)$.

(3) Given three vertices a, b and c of H, output a face of H that contains a, b and c or report that there is none.

We can compute the set F of all left and right faces of the vertices a, b and c in constant time; note that F contains at most 6 faces. If there is a face f in H containing a, b and c, at least one vertex in $\{a,b,c\}$ is neither source nor sink of f, which implies that f must be in F. For a query (3), it therefore suffices to test each face $f \in F$ for containing a, b and c, respectively. A vertex v is contained in f if and only if v is either source or sink of f, which can be checked in time $O(1)$, or one of the remaining vertices in f, which can be checked in time $O(1)$ by testing whether f is the left or right face of v.

The data structure additionally supports each of the following modifications to H in amortized time $O(1)$ and maintains a plane *st*-graph after every modification.

(4) Subdivide an edge.
(5) Given two non-adjacent vertices a and b and a face f of H that contains a and b, add the edge ab inside f.

Clearly, K_4 can be embedded as a plane st-graph and we initialize H with this embedding. Every operation O of Type 1a–d can be converted into at most three of the modifications (4) and (5). E. g., we can add a claw having its attachments $\{a, b, c\}$ in a common face by consecutively inserting the edge ab, subdividing ab with a new vertex x and adding the edge xc. For operations O of Type 1b–d, the condition in Lemma 3 can be checked in constant time by one query (1) or one query (3).

It only remains to show how we can check the condition in Lemma 3 if O is of Type 1a. According to Theorem 2, all operations in C that follow O will be of Type 1a, which implies that H is a spanning subgraph of G. In other words, each of the remaining operations in C adds only an edge that will not be subdivided afterwards. Hence, the order in which these remaining edges E' are added does not matter.

We use a trick similar as in [8, Lemma 3.2] and add the remaining edges E' in an order such that each added edge has an endpoint that is the sink or source of at most 11 faces. If we know this order, we can use query (2) to ensure that every step can be computed in constant time.

In order to compute this order, we maintain an auxiliary graph H^A whose vertex set consists of all vertices in $V(H)$ that are incident to an edge in E'. There is an edge between two vertices a and b in H^A if a and b are source and sink vertices of the same face. Note that H^A may have parallel edges. We construct H^A in linear time when the first operation of Type 1a in C is encountered; after every modification (5), H^A can be updated in time $O(1)$, as each face f stores a pointer to its source and sink.

As H^A is planar and has at most two parallel edges between every two vertices (as H is simple and triconnected), it contains at most $6|V(H^A)| - 12$ edges. Hence, there is at least one vertex of degree at most 11 in H^A. We note that the degree bound 6 proposed in [8, Lemma 3.2] should also be 11, as the auxiliary graph used there is not necessarily simple.

Before the first operation of Type 1a in C is applied to H, we construct a list *Small* of all vertices in H^A having degree at most 11 in linear time; again, this list is easy to maintain under modifications (5) in time $O(1)$. Now we just choose successively a vertex $v \in \mathit{Small}$ and an edge $e = vw$ in E' and perform modification (5) with v and w if v and w have been reported to be in the same face. This allows to check the condition in Lemma 3 for each of the remaining edges in E' in constant time using query (2). We conclude the following theorem.

Theorem 3. *The planarity test Algorithm 1 can be implemented in linear time.*

5 Extensions

A *Kuratowski-subdivision* is a subdivision of either a $K_{3,3}$ or of a K_5 and proves every graph that contains it to be non-planar. We show how a Kuratowski-subdivision can be computed if an operation O is encountered that has not all attachment vertices on one face in H. The computation follows in parts the arguments given in the short proof of Kuratowski's Theorem in [27]. The fact that the Kuratowski-subdivision is computed in a triconnected component does not matter; it is straight-forward to get a corresponding Kuratowski-subdivision in the input graph by reversing the splits that were done to obtain the triconnected components.

We first recall planarity-related terminology.

Definition 2. *For a cycle C in a graph G, let a* C-component *be either an edge $e \notin C$ with both endpoints in C or a connected component of $G \setminus V(C)$ together with all edges that join the component to C and all endpoints of these edges. The* vertices of attachment *of a C-component H are the vertices in $H \cap C$.*

Two C-components H_1 and H_2 *avoid each other* if C contains two vertices u and v such that H_1 has all vertices of attachment on one path in C from u to v and H_2 has all vertices of attachment on the other path in C from u to v.

Two C-components *overlap* if they do not avoid each other. Let two C-components H_1 and H_2 be *C-equivalent* if $H_1 \cap C = H_2 \cap C$ and this set contains exactly three vertices. Let H_1 and H_2 be *skew* if C contains four distinct vertices x_1, x_2, x_3 and x_4 in cyclic order such that x_1 and x_3 are in H_1 and x_2 and x_4 are in H_2. We will need the following basic fact about C-components.

Lemma 4 ([27]). *Two C-components overlap if and only if they are either skew or C-equivalent.*

Now we are prepared to compute a Kuratowski-subdivision.

Lemma 5. *Let H be a planar embedding of a simple triconnected graph on at least 4 vertices and let O be an operation of Type 1a–d on H whose attachments are not all contained in one face of H. Then the graph H' that is obtained from H by applying O contains a subdivision of K_5 or $K_{3,3}$, and this subdivision can be computed in linear time.*

Proof. First assume that O adds a claw and every two of the three attachments $\{a, b, c\}$ of O are contained in a face of H, respectively; we call these three faces f_1, f_2 and f_3. Let J be a closed Jordan curve in $f_1 \cup f_2 \cup f_3$ that intersects H exactly at a, b and c. Since a, b and c are not all contained in a face of H, there are vertices v_{in} and v_{out} strictly inside and strictly outside J, respectively. Since H is 3-connected, we can compute from v_{in} and v_{out} three internally vertex-disjoint paths to $\{a, b, c\}$, respectively, by performing one depth first search. Adding the claw of O to these paths gives a $K_{3,3}$-subdivision in H'.

The only remaining case is that O has at least two attachments a and b that are not contained in one face of H; note that a and b may be edges. As $H \setminus a$ is 2-connected, it contains a cycle C that is the boundary of the face which contains a in its interior. By assumption, $b \notin C$. Let H_a and H_b be the C-components of H containing a and b, respectively. By definition of C, H_a is the only C-component in the interior of C.

We show that H_a and H_b overlap. Assume the contrary. Then H_b has two vertices of attachment u and v such that H_a has all vertices of attachment on one path $P_a \subset C$ from u to v and H_b has all vertices of attachment on the other path $P_b \subset C$ from u to v. If a is a vertex, a and b are in different components of $H \setminus \{u, v\}$, since H is a planar embedding. This contradicts H to be triconnected. Otherwise, a is an edge (which will be subdivided by O) and $H_a = a$. Then, as H is simple, P_a has length at least two, which implies that an inner vertex in P_a is in a different component of $H \setminus \{u, v\}$ than b. This contradicts H to be triconnected. Thus, H_a and H_b overlap.

According to Lemma 4, H_a and H_b are either skew or C-equivalent. The cycle C, H_a and H_b can be easily computed in linear time. Deciding whether H_a and H_b are skew and computing the vertices x_1, x_2, x_3 and x_4 on C, whose existence defines this property amounts to one traversal along C. If a is an edge, subdivide a and let a' be the new vertex of degree two; otherwise let $a' = a$. Define b' accordingly. Due to Menger's Theorem, there are either two or three internally disjoint paths from a' to C in H_a (and from b' to C in H_b), depending on whether H_a (H_b) is an edge. These paths can be computed by a depth-first search that starts with the desired vertex.

If H_a and H_b are skew, we compute two of these paths in H_a that end at x_1 and x_3, respectively, and two in H_b that end at x_2 and x_4, respectively. Taking the union of these four paths, C and T forms a $K_{3,3}$-subdivision, where T is either the added edge of O or the path of length two from a to b if O adds a claw. If H_a and H_b are C-equivalent, the union of the three paths in H_a and H_b, respectively, C and T gives a K_5-subdivision. □

We remark that our algorithm can be easily extended to output always a $K_{3,3}$-subdivision in linear time when the input graph G is 3-connected, non-planar and $G \neq K_5$. This is based on the following variant of Kuratowski's Theorem for triconnected graphs.

Lemma 6 ([15]). *A simple triconnected graph $G \neq K_5$ is planar if and only if G does not contain a $K_{3,3}$-subdivision.*

Note that we get a K_5-subdivision K only in the case that O adds a claw. The desired $K_{3,3}$-subdivision can then be obtained from K by rerouting one of the paths of K that ends at a to the center vertex of the claw.

Open Questions. The most immediate question is whether there is a simple linear-time algorithm that computes the construction sequence C of a triconnected graph. This would immediately imply a simple linear-time planarity test. As argued before, one may even assume planarity to find such a sequence. Further, it seems possible that such an algorithm, or the existing one in [24], can be extended to compute the triconnected components of the input graph, similarly as in the triconnectivity test of Hopcroft and Tarjan [12]. This would subsume the computation of C and the preprocessing of the graph into triconnected components. The proposed new algorithmic approach to planarity testing might also allow to recognize other subclasses of planar graphs efficiently (e.g., planar graphs that contain no subdivision of $K_5 - e$ or of W_4).

References

1. D. W. Barnette and B. Grünbaum. On Steinitz's theorem concerning convex 3-polytopes and on some properties of 3-connected graphs. In *Many Facets of Graph Theory*, pages 27–40, 1969.
2. J. A. Bondy and U. S. R. Murty. *Graph Theory.* Springer, 2008.
3. K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using *PQ*-Tree algorithms. *J. Comput. Syst. Sci.*, 13:335–379, 1976.
4. J. M. Boyer and W. J. Myrvold. On the cutting edge: Simplified $O(n)$ planarity by edge addition. *Journal of Graph Algorithms an Applications*, 8(3):241–273, 2004.
5. U. Brandes. Eager st-Ordering. In *Proceedings of the 10th European Symposium of Algorithms (ESA'02)*, pages 247–256, 2002.
6. U. Brandes. The left-right planarity test. Manuscript submitted for publication, 2009.
7. N. Chiba, T. Nishizeki, S. Abe, and T. Ozawa. A linear algorithm for embedding planar graphs using *PQ*-Trees. *J. Comput. Syst. Sci.*, 30(1):54–76, 1985.
8. H. N. Djidjev. A linear-time algorithm for finding a maximal planar subgraph. *SIAM J. Discrete Math.*, 20(2):444–462, 2006.
9. H. d. Fraysseix, P. O. de Mendez, and P. Rosenstiehl. Trémaux Trees and planarity. *Int. J. Found. Comput. Sci.*, 17(5):1017–1030, 2006.
10. C. Gutwenger and P. Mutzel. A linear time implementation of SPQR-trees. In *Proceedings of the 8th International Symposium on Graph Drawing (GD'00)*, pages 77–90, 2001.
11. B. Haeupler and R. E. Tarjan. Planarity algorithms via PQ-Trees (extended abstract). *Electronic Notes in Discrete Mathematics*, 31:143–149, 2008.
12. J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973.
13. J. E. Hopcroft and R. E. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.
14. A. K. Kelmans. A new planarity criterion for 3-connected graphs. *Journal of Graph Theory*, 5:259–267, 1981.
15. A. Liebers. Planarizing graphs — A survey and annotated bibliography. *J. Graph Algorithms Appl.*, 5(1), 2001.
16. R. M. McConnell, K. Mehlhorn, S. Näher, and P. Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, 2011.
17. S. McLane. A structural characterization of planar combinatorial graphs. *Duke Mathematical Journal*, 3:466–472, 1937.
18. K. Mehlhorn and P. Mutzel. On the embedding phase of the Hopcroft and Tarjan planarity testing algorithm. *Algorithmica*, 16(2):233–242, 1996.
19. A. Neumann. Implementation of Schmidt's algorithm for certifying triconnectivity testing. Master's thesis, Universität des Saarlandes and Graduate School of CS, Germany, 2011.
20. T. Nishizeki and N. Chiba. *Planar graphs: Theory and algorithms.* North-Holland, 1988.
21. OGDF - The Open Graph Drawing Framework, 06/2013. http://www.ogdf.net.
22. M. Patrignani. Planarity testing and embedding. In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization.* CRC Press, to appear.
23. J. M. Schmidt. Construction sequences and certifying 3-connectedness. In *Proceedings of the 27th Symposium on Theoretical Aspects of Computer Science (STACS'10)*, pages 633–644, 2010.

24. J. M. Schmidt. Certifying 3-connectivity in linear time. In *Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP'12)*, pages 786–797, 2012.
25. W. K. Shih and W. L. Hsu. A new planarity test. *Theor. Comput. Sci.*, 223:179–191, 1999.
26. R. Tamassia and I. G. Tollis. A unified approach to visibility representation of planar graphs. *Discrete & Computational Geometry*, 1:321–341, 1986.
27. C. Thomassen. Kuratowski's theorem. *Journal of Graph Theory*, 5(3):225–241, 1981.
28. W. T. Tutte. Connectivity in graphs. In *Mathematical Expositions*, volume 15. University of Toronto Press, 1966.
29. K.-P. Vo. Finding triconnected components of graphs. *Linear and Multilinear Algebra*, 13:143–165, 1983.
30. K.-P. Vo. Segment graphs, depth-first cycle bases, 3-connectivity, and planarity of graphs. *Linear and Multilinear Algebra*, 13:119–141, 1983.
31. D. B. West. *Introduction to Graph Theory.* Prentice Hall, 2001.
32. H. Whitney. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54(1):150–168, 1932.
33. S. G. Williamson. Embedding graphs in the plane — algorithmic aspects. In *Combinatorial Mathematics, Optimal Designs and Their Applications*, volume 6 of *Annals of Discrete Mathematics*, pages 349–384. Elsevier, 1980.