Chapter 5
Algorithms for 1-Planar Graphs

Seok-Hee Hong

Abstract A *1-planar* graph is a graph that can be embedded in the plane with at most one crossing per edge. It is known that testing 1-planarity of a graph is NP-complete. This chapter reviews the algorithmic results on 1-planar graphs. We first review a linear time algorithm for testing maximal 1-planarity of a graph if a *rotation system* (i.e., the circular ordering of edges for each vertex) is given. A graph is *maximal 1-planar* if the addition of an edge destroys 1-planarity. Next, we sketch a linear time algorithm for testing outer-1-planarity. A graph is *outer-1-planar* if it has an embedding in which every vertex is on the outer face and each edge has at most one crossing. The 1-plane graphs have two forbidden subgraphs to admit a straight-line drawing. We review a linear time algorithm for constructing a straight-line drawing of 1-plane graphs. Finally, we conclude with reviews on recent related results.

5.1 Introduction

Recent research topics in topological graph theory and graph drawing generalize the notion of planarity to sparse non-planar graphs, called *beyond planar graphs*, either with forbidden edge crossing patterns or with specific types of edge crossings.

This chapter reviews algorithmic results on 1-planar graphs, i.e., graphs that can be embedded with at most one crossing per edge [25]. The 1-planar graphs are introduced by Ringel [26] in the context of simultaneously coloring vertices and faces of planar graphs. Subsequently, the combinatorial aspects of 1-planar graphs have been investigated.

For example, Borodin [6] investigated coloring for 1-planar graphs, and Borodin et al. [7] studied the acyclic colorability of 1-planar graphs; Zhang and Wu [30] studied the edge colorability of 1-planar graphs. In particular, Pach and Toth [25] proved that a 1-planar graph with n vertices has at most $4n - 8$ edges, which is a tight upper bound.

S.-H. Hong (✉)
University of Sydney, Sydney, Australia
e-mail: seokhee.hong@sydney.edu.au

S.-H. Hong and T. Tokuyama (eds.), *Beyond Planar Graphs*,
https://doi.org/10.1007/978-981-15-6533-5_5

There are a number of structural results on 1-planar graphs by Fabrici and Madaras [12], and *maximal 1-planar graphs* by Hudak et al. [21] and Suzuki [28]. Suzuki [27] also investigated structural properties of *optimal 1-planar graphs* (i.e., 1-planar graphs with the maximum number of $4n - 8$ edges).

The class of 1-planar graphs is not closed under edge contraction; accordingly, computational problems seem difficult. Grigoriev and Bodlaender [15], and Korzhik and Mohar [22] independently proved that testing 1-planarity of a graph is NP-complete. It remains NP-hard, even if the rotation system is given as part of the input, shown by Auer et al. [2]. Furthermore, Cabello and Mohar [8] showed that NP-hardness holds even if the input graph is an *almost planar graph* (i.e., deletion of an edge makes the resulting graph planar). More recently, Bannister et al. [3] studied the fixed parameter complexity of 1-planarity.

On the positive side, efficient polynomial time algorithms are known for special subclasses of 1-planar graphs. For example, a linear time algorithm is available for testing maximal 1-planarity, if the rotation system is given, by Eades et al. [10]. Hong et al. [16] and Auer et al. [1] independently presented linear time algorithms for testing outer-1-planarity.

The classical *Fáry's Theorem* [13] showed that every *plane graph* (i.e., a planar graph with a given planar embedding) admits a planar straight-line drawing. However, 1-plane graphs (i.e., 1-planar graphs with a given 1-planar embedding) have two forbidden subgraphs to admit a straight-line drawing, shown by Thomassen [29]. Hong et al. [20] presented linear time testing and drawing algorithms to construct such a straight-line drawing of 1-plane graphs if it exists.

This chapter reviews algorithmic results on 1-planar graphs. More specifically, we describe three linear time algorithms in the following sections:

1. Section 5.2: linear time algorithm by Eades et al. [10] for testing maximal 1-planarity of a graph G with a given rotation system.
2. Section 5.3: linear time algorithm by Hong et al. [16] for testing outer-1-planarity of a graph.
3. Section 5.4: linear time algorithm by Hong et al. [20] for constructing a straight-line drawing of a 1-plane graph.

Section 5.5 concludes with reviews on recent progress.

5.2 Testing Maximal 1-Planarity

Eades et al. [10] proved the following main theorem.

Theorem 5.1 *There exists a linear time algorithm that tests whether graph G with a given rotation system Φ has a maximal* 1*-planar embedding consistent with Φ. If such an embedding exists, it is unique and the algorithm computes the embedding.*

First, it was shown that in any maximal 1-planar embedding, the subgraph induced by planar edges (called the *red graph*) is spanning and biconnected, and that if

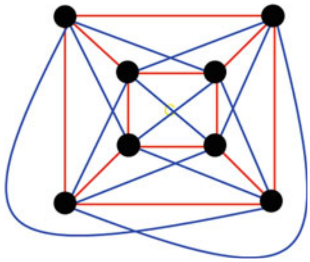

Fig. 5.1 Maximal 1-planar graph

the rotation system does admit a maximal 1-planar embedding, then it is unique. Figure 5.1 shows an example of a maximal 1-planar graph.

Note that a rotation system Φ does not define crossings between edges. However, for a planar graph, a rotation system uniquely determines a planar embedding. Therefore, to determine a 1-planar embedding $\xi(G_\Phi)$, it is sufficient to determine a rotation system of the planar embedding $\xi(G_P)$ of *planarization* G_P of the 1-planar embedding.

An embedding $\xi(G)$ of a graph G defines the crossing-free (called *red*) edges as well as the crossing (called *blue*) edges. Denote the subgraph of a graph G induced by the red edges as *red graph* G_R. Now, we sketch a linear time algorithm to test maximal 1-planarity of a graph with a given rotation system, consisting of the following five steps:

Algorithm: Testing Maximal 1-Planarity
Input: G_Φ, a graph G with a rotation system Φ.
Output: 1-planar embedding $\xi(G_\Phi)$ or "no".

1. If $|E(G)| > 4n - 8$ or G is not biconnected, then return("no").
2. Compute the red planar subgraph G_R of G_Φ.
3. If G_R is not planar or not biconnected, then return("no").
4. Test 1-planarity of G_Φ, and compute $\xi(G_\Phi)$ and $\xi(G_P)$.
5. Test maximality of $\xi(G_\Phi)$.

Steps 1 and 3 use the Pach–Toth bound [25], a standard biconnectivity algorithm, and a planarity testing algorithm. In the following, we sketch Steps 2, 4, and 5.

Step 2: Computing the Red Subgraph G_R
Consider graph G as a directed graph, with two directed edges (u, v) and (v, u) for each pair u, v of adjacent vertices. We say that a directed edge (v_2, v_3) is the *rightmost continuation* of a directed edge (v_1, v_2), if vertex v_3 is the vertex that precedes v_1 in the circular ordering of v_2. We say that a walk $v_1, \ldots, v_t$ is a *completed rightmost walk*, if: (i) for every $i \in \{1, \ldots, t\}$, the directed edge (v_i, v_{i+1}) is the rightmost continuation of the directed edge (v_{i-1}, v_i), and (ii) for all $i, j \in \{1, \ldots, t\}$, if $(v_i, v_{i+1}) = (v_j, v_{j+1})$, then $i = j$. See Fig. 5.2, for examples.

Completed rightmost walks characterize the colors of the edges of G_Φ, as in the following lemma.

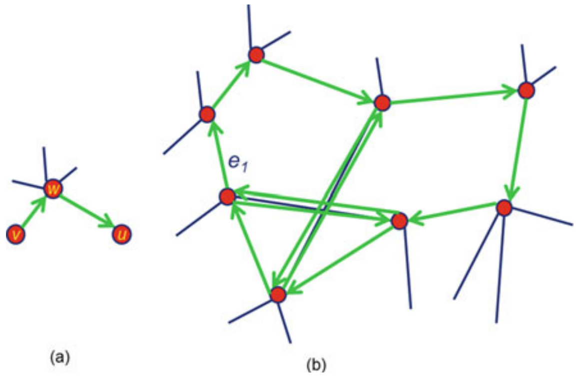

Fig. 5.2 Examples of **a** rightmost continuation; **b** completed rightmost walk

Lemma 5.1 *Let G be a 1-plane graph with a given rotation system Φ, whose red graph G_R is spanning and biconnected. An edge e of G is red if and only if there is a completed rightmost walk on G_Φ that traverses e only in one direction.*

We can design an algorithm that takes G_Φ as input, and computes the color of the edges as follows:

(i) simply traverse the graph with rightmost walks;
(ii) by marking edges after the traversal, we can color the edges red or blue.

Step 4: Computing a 1-Planar Embedding of G_Φ
We now test whether there exists a 1-planar embedding of G_Φ consistent with the colors. If such an embedding exists, we compute a planar embedding of the planarization G_P of G_Φ.

After testing biconnectivity and planarity of G_R in Step 3, we have a planar embedding $\xi(G_R)$ of G_R which preserves the given rotation system Φ of G.

Since the 1-planar embedding of G_Φ is unique, if it exists, this implies that we can use the rotation system Φ to identify the red facial cycles and the blue edges inside each red face. Then crossings can be detected by traversing each red face; the traversal detects any edge with more than one crossing.

Lemma 5.2 *There exists a linear time algorithm that tests whether there is a 1-planar embedding of G_Φ that is consistent with Φ such that G_R is the red subgraph. If such an embedding $\xi(G_\Phi)$ exists, it is unique and the algorithm computes the planar embedding $\xi(G_P)$ of the planarization of $\xi(G_\Phi)$.*

Step 5: Testing Maximality of 1-Planar Embedding
Now, we show that maximality of a 1-planar embedding of G with a given rotation system Φ can be tested in linear time. Let $\xi(G)$ be a maximal 1-planar embedding of a graph G, G_P be the planarization of G, and $\xi(G_P)$ be the planar embedding of G_P induced by $\xi(G)$. Let f be a facial cycle in $\xi(G_P)$.

Note that maximal 1-planar graphs have the following properties:

- Each crossing in $\xi(G)$ induces a 4-clique.

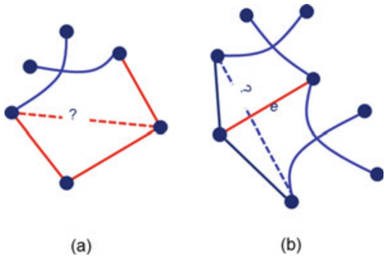

Fig. 5.3 Testing maximality of an 1-planar embedding: **a** testing possible addition of a red edge; **b** testing possible addition of a blue edge

- The face f has at most four real vertices, and at most eight vertices (real plus virtual).

We can simply test whether there are two nonadjacent real vertices v_1 and v_2 such that adding the edge (v_1, v_2) does not destroy 1-planarity as follows:

(i) First test each face whether it contains two such v_1 and v_2 (see Fig. 5.3a);
(ii) Then test red edge e whether we can add (v_1, v_2) by crossing e, where v_1 and v_2 are on the faces separated by e (see Fig. 5.3b).

Using the properties of maximal 1-planar graphs above, we can perform these testings in linear time.

5.3 Testing Outer-1-Planarity

We now review a linear time algorithm by Hong et al. [16] to test the outer-1-planarity of a graph G. The following theorem summarizes the main results.

Theorem 5.2 *There is a linear time algorithm to test whether a given graph is outer-1-planar. The algorithm computes an outer-1-planar embedding if it exists.*

To prove Theorem 5.2, a subclass of outer-1-planar graphs, called *one-sided-outer-1-planar (OSO1P)* graph, was introduced as follows. Let G be a graph with vertices s and t, and $G_{+(s,t)}$ be the graph obtained by adding the edge (s, t), if it is not already in G. If $G_{+(s,t)}$ has an outer-1-planar embedding with the edge (s, t) on the outer face, then G is called *one-sided-outer-1-planar (OSO1P)* with respect to (s, t).

A graph is outer-1-planar if and only if its biconnected components are outer-1-planar. Therefore, the algorithm focuses on the biconnected case, using the *SPQR tree* [5] to represent the decomposition of a biconnected graph into triconnected components. We now review the basic terminology of the SPQR tree.

Each node ν in the SPQR tree is associated with a graph called the *skeleton* of ν, denoted by $\sigma(\nu)$. There are four types of nodes ν:

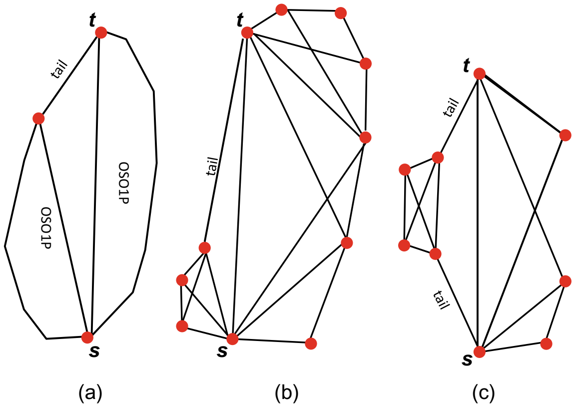

Fig. 5.4 AOSO1P graph consists of a parallel composition of an OSO1P graph and an OSO1P S-node with a tail: **a** general shape of an AOSO1P graph with respect to (s, t); **b** AOSO1P graph with respect to (s, t); **c** AOSO1P graph with respect to both (s, t) and (t, s)

- S-node: $\sigma(\nu)$ is a simple cycle with at least three vertices;
- P-node: $\sigma(\nu)$ consists of two vertices connected by at least three edges;
- Q-node: $\sigma(\nu)$ consists of two vertices connected by a *real* edge and a *virtual* edge; and
- R-node: $\sigma(\nu)$ is a simple triconnected graph.

A rooted SPQR tree can be obtained by choosing an arbitrary node as its root. Let ρ be the parent node of an internal node ν. The graph $\sigma(\rho)$ has exactly one *virtual edge e* in common with $\sigma(\nu)$, called the *parent virtual edge* of $\sigma(\nu)$, and a *child virtual edge* in $\sigma(\rho)$. Denote the graph formed by the union of $\sigma(\nu)$ over all descendants ν of ρ by G_ρ.

Let μ be an S-node with parent separation pair (u, v). A *tail at u* for μ is a Q-node child (that is, a real edge) with parent virtual edge (u, x) for some vertex x. A P-node ν is called *almost one-sided outer-1-planar (AOSO1P) with respect to the directed edge* (s, t), if G_ν consists of a parallel composition of an OSO1P graph with respect to (s, t) and an S-node μ such that μ has a tail at t and μ is OSO1P with respect to (s, t). See Fig. 5.4 for examples.

If G is an outer-1-planar graph, then $\sigma(\nu)$ and G_ν are outer-1-planar graphs. If G_ν is a one-sided outer-1-planar (OSO1P) graph with respect to the parent virtual edge (s, t) of ν, then denote ν as a one-sided outer-1-planar (OSO1P) node with respect to (s, t).

Step 1: Testing OSO1P and AOSO1P

The algorithm traverses the SPQR tree of G from the leaves toward the root, computing two boolean labels $OSO1P(\nu, s, t)$ and $AOSO1P(\nu, s, t)$ which indicate whether ν is OSO1P or AOSO1P with respect to (s, t). The label $AOSO1P(\nu, s, t)$ is computed for each P-node ν only.

Note that the only triconnected outer-1-planar graph is K_4 with unique outer-1-planar embedding. Therefore, the R-node case is easy; however, P-node and S-node cases are more involved.

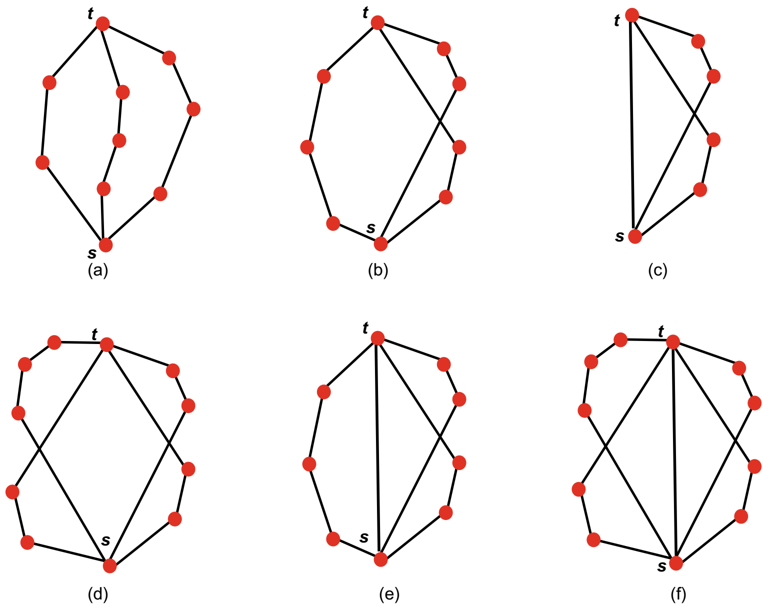

Fig. 5.5 Embeddings of a set of paths sharing endpoints: **a** planar embedding; **b** outer-1-planar embedding of three non-trivial paths; **c** outer-1-planar embedding of three paths, where one path is trivial; **d** outer-1-planar embedding of four non-trivial paths; **e** outer-1-planar embedding of four paths, where one path is trivial; **f** outer-1-planar embedding of five paths, where one path is trivial

Figure 5.5 illustrates structural properties on the possible outer-1-planar embeddings of a set of paths that share endpoints. Let P be a set of paths between two vertices s and t. A path from s to t is called *non-trivial*, if it contains more than two vertices.

We now describe each case (i.e., R-node, P-node, S-node) in detail.

(i) R-node: Let ν be an R-node with parent virtual edge (u, v). Then G_ν is OSO1P with respect to (u, v) if and only if:

1. $\sigma(\nu)$ is isomorphic to K_4; and
2. an edge (u, a) of $\sigma(\nu)$ with $a \neq v$ incident with u represents a child Q-node of ν; an edge (v, b) of $\sigma(\nu)$ with $b \neq u$ represents a child Q-node of ν; and (u, a) crosses (v, b); and
3. for each child ν' of ν, ν' is OSO1P with respect to (c, d), where (c, d) is the parent virtual edge of ν'.

Figure 5.6a shows an example of an OSO1P R-node: $\sigma(\nu)$ is K_4, where the inner crossing edges are real edges (i.e., Q-node children), and outer edges are OSO1P child nodes. Figure 5.6b shows a non-OSO1P R-node, where crossing edges are not real edges.

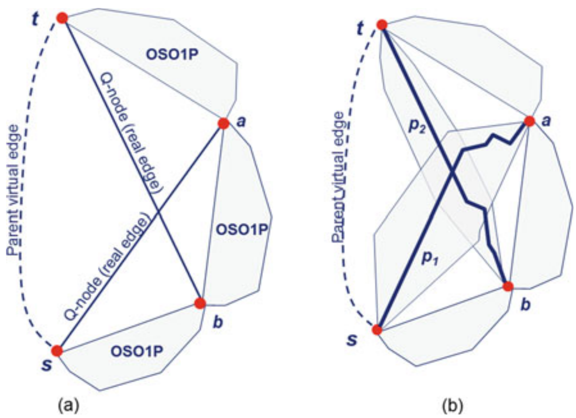

Fig. 5.6 Examples of **a** OSO1P R-node; **b** non-OSO1P R-node

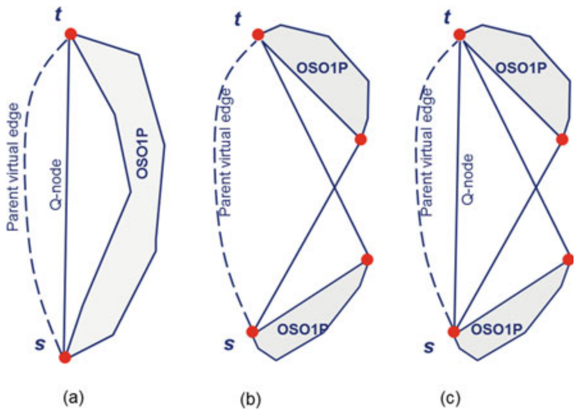

Fig. 5.7 Illustration for an OSO1P P-node

(ii) P-node: Based on the structural properties shown in Fig. 5.5, an OSO1P P-node can have at most three children. Let ν be a P-node with parent virtual edge (s, t). Then G_ν is OSO1P with respect to (s, t) if and only if

- ν has two children, where one is a Q-node (s, t), and the other is OSO1P with respect to (s, t) (see Fig. 5.7a); or
- ν has two children, where one is an S-node with tail at s which is OSO1P with respect to (s, t), and the other is an S-node with tail at t which is OSO1P with respect to (s, t) (see Fig. 5.7b); or
- ν has three children, where one is a Q-node (s, t), another is an S-node with tail at s which is OSO1P with respect to (s, t), and the other is an OSO1P S-node with tail at t which is OSO1P with respect to (s, t) (see Fig. 5.7c).

It is straightforward to extend the above conditions to test whether a P-node ν is AOSO1P.

(iii) S-node: Let ν be an S-node with children $\nu_1, \nu_2, \ldots, \nu_k$, where the parent virtual edge of ν_i is (s_{i-1}, s_i); see Fig. 5.8a. If each child ν_i is OSO1P with respect to

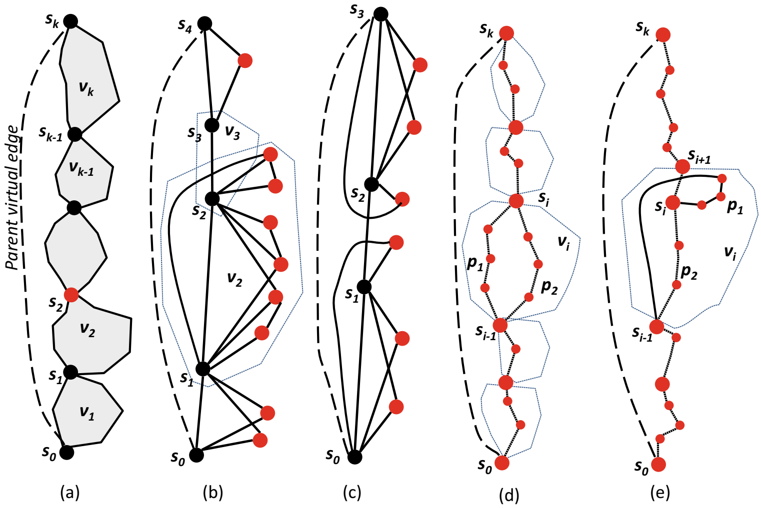

Fig. 5.8 Examples of **a** S-node; **b** OSO1P S-node with a child ν_2 that is not OSO1P; **c** S-node that satisfies the necessary conditions, but is not OSO1P; **d** Two paths p_1 and p_2 in G_{ν_i}; **e** The path p_1 crosses the edge (s_i, s_{i+1})

(s_{i-1}, s_i), then clearly ν is OSO1P with respect to (s_0, s_k); however, the converse is false. Figure 5.8b shows the necessary conditions, where ν is OSO1P with respect to (s_0, s_k), however, the child ν_2 is not OSO1P with respect to (s_1, s_2). Note that ν_3 is a Q-node, and an edge from the skeleton of ν_2 crosses this edge.

Let ν be an S-node with children $\nu_1, \nu_2, \ldots, \nu_k$, where the parent virtual edge of ν_i is (s_{i-1}, s_i), and G_ν is OSO1P with respect to (s_0, s_k). Then for $1 \le i \le k$:

- ν_i is OSO1P with respect to (s_{i-1}, s_i); or
- $i < k$, ν_i is AOSO1P with respect to (s_i, s_{i-1}), and ν_{i+1} is a Q-node; or
- $i > 1$, ν_i is AOSO1P with respect to (s_{i-1}, s_i), and ν_{i-1} is a Q-node.

The above conditions are necessary for an S-node to be OSO1P, however not sufficient; e.g., see Fig. 5.8c. The problem is that the Q-node represented by the edge (s_1, s_2) has two crossings. We can express sufficient conditions for an OSO1P S-node recursively, as follows.

Let ν be an S-node with children $\nu_1, \nu_2, \ldots, \nu_k$, where the parent virtual edge of ν_i is (s_{i-1}, s_i), and let $G(\nu_1, \nu_2, \ldots, \nu_k)$ denote the series composition of graphs $G_{\nu_1}, G_{\nu_2}, \ldots, G_{\nu_k}$. Then G_ν is OSO1P with respect to (s_0, s_k) if and only if:

- G_{ν_1} is OSO1P with respect to (s_0, s_1) and $G(\nu_2, \nu_3, \ldots, \nu_k)$ is OSO1P with respect to (s_1, s_k); or
- ν_1 is a Q-node, G_{ν_2} is AOSO1P with respect to (s_1, s_2), and $G(\nu_3, \nu_4, \ldots, \nu_k)$ is OSO1P with respect to (s_2, s_k); or
- G_{ν_1} is AOSO1P with respect to (s_1, s_0), ν_2 is a Q-node, and $G(\nu_3, \nu_4, \ldots, \nu_k)$ is OSO1P with respect to (s_2, s_k).

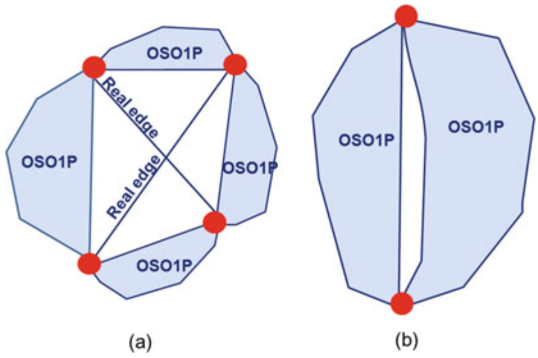

Fig. 5.9 Testing O1P at the root node: **a** Root R-node; **b** Root P-node

Step 2: Testing Outer-1-Planarity
After computing labels $OSO1P(\nu, s, t)$ and $AOSO1P(\nu, s, t)$ for all internal nodes ν of the SPQR tree, we can test whether the whole graph G (i.e., the root ρ) is outer-1-planar. We can require the root node ρ to be an R-node or a P-node.

For the root R-node, the algorithm tests the following conditions (see Fig. 5.9a): G is outer-1-planar (O1P) if and only if

1. $\sigma(\rho)$ is isomorphic to K_4, and
2. at least two children of ρ are Q-nodes, and
3. for each child node ν of $\sigma(\rho)$ with parent virtual edge (a, b), G_ν is OSO1P with respect to (a, b).

For the root P-node, testing O1P is simpler: G is outer-1-planar if and only if $\sigma(\rho)$ is a parallel composition of two OSO1P graphs (see Fig. 5.9b).

5.4 Straight-Line Drawing Algorithm for 1-Planar Graphs

The classical *Fáry's Theorem* [13] proved that every *plane graph* (i.e., planar topological embedding of a planar graph) has a planar straight-line drawing. Indeed, planar straight-line drawing is one of the most popular drawing conventions in Graph Drawing [4, 24]. For example, de Fraysseix et al. [14] showed that planar straight-line grid drawing can be efficiently constructed in a *quadratic area*.

On the other hand, Thomassen [29] showed that there are two 1-plane graphs that cannot be drawn with straight-line edges. More specifically, he proved that a 1-plane graph G admits a straight-line 1-planar drawing if and only if G contains neither the B graph (see Fig. 5.10a) nor the W graph (see Fig. 5.10b).

Based on Thomassen's characterization, Hong et al. [20] presented linear time testing and drawing algorithms, proving the following main theorem.

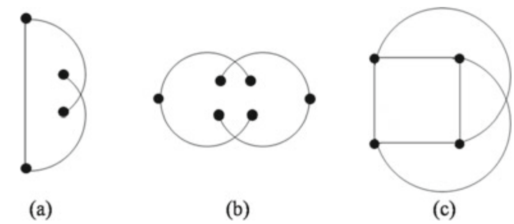

Fig. 5.10 **a** The B graph; **b** the W graph; **c** 1-plane embedding of K_4 containing B subgraph

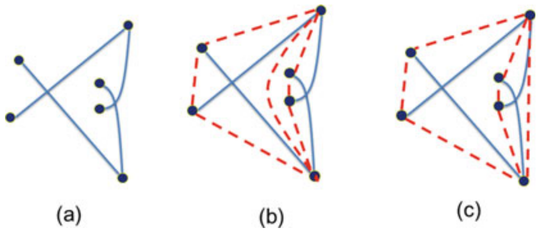

Fig. 5.11 Example of an augmentation: **a** 1-plane graph without the B graph or the W graph; **b** bad augmentation introducing B graph; **c** good augmentation not introducing B graph

Theorem 5.3 *There is a linear time algorithm to test whether a 1-plane embedding contains the B graph or the W graph, and a linear time drawing algorithm to construct a straight-line 1-planar drawing if it exists.*

Here, we mainly explain the drawing algorithm consisting of two steps: an augmentation step and a drawing step.

Step 1: Red-Maximal Augmentation

The first step, called *red-maximal augmentation*, is to augment a 1-plane graph G by adding edges without introducing new crossings while preserving the straight-line drawability of G. Denote the crossing-free edges of a 1-plane graph G as *red* edges.

A *red augmentation* $G' = (V, E')$ of $G = (V, E)$ is a 1-plane graph with $E \subseteq E'$ such that no edge in $E' - E$ has a crossing. A 1-plane graph is *red-maximal* if the addition of any edge makes a crossing. The red-maximal 1-plane graphs have nice properties, which are helpful for the drawing algorithm.

Computing a red-maximal augmentation G^+ of a 1-plane graph G, preserving the absence of B and W subgraphs, consists of two steps: (i) the first step adds edges for each crossing γ with a 4-cycle; (ii) the second step triangulates any remaining faces.

The first step adds edges to a 1-plane graph G without the B subgraph or the W subgraph until each crossing γ is surrounded by a 4-cycle. Note that there are different ways to add the edge (a, b), as shown in Fig. 5.11: Fig. 5.11b introduces the B subgraph, while Fig. 5.11c does not.

Furthermore, there may be many crossing vertices γ that share the same neighbors, (a, b), as shown in Fig. 5.12c. Nevertheless, it is always possible to route the edge (a, b) without introducing the B subgraph, using the orientation of crossings with respect to (a, b): *clockwise* (see Fig. 5.12a) or *anticlockwise* (see Fig. 5.12b). For example, in Fig. 5.12c, the edge (a, b) can be added between γ_j and γ_{j+1} without introducing the B graph.

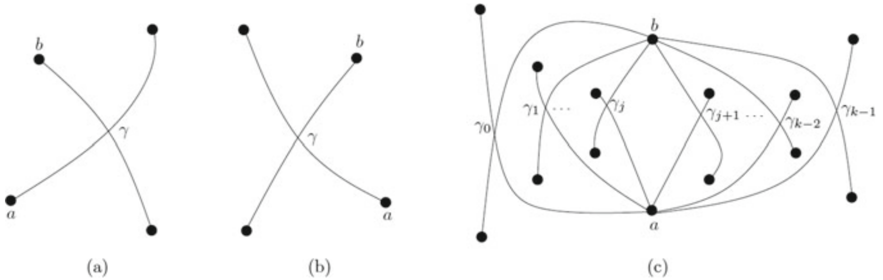

Fig. 5.12 **a** The crossing γ is *clockwise* with respect to (a, b); **b** γ is *anticlockwise* with respect to (a, b); **c** (a, b) is a separation pair with many crossings: the edge (a, b) can be added between γ_j and γ_{j+1} without introducing the B graph

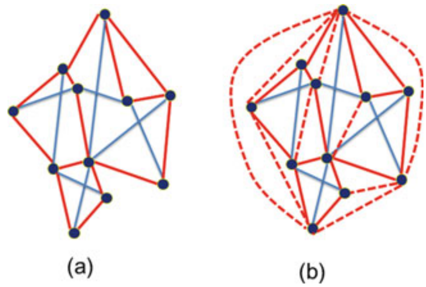

Fig. 5.13 Example of a red-maximal augmentation: **a** 1-plane embedding; **b** red-maximal augmentation of **a**

Based on the orientation of crossings, we can add edges to obtain an augmentation such that each crossing is surrounded by a 4-cycle, without introducing the W subgraph and the B subgraph.

The second step is triangulating the remaining faces. Let a and b be two nonadjacent vertices in the 1-plane graph G' after the first step, sharing a face f. We can add the edge (a, b) inside f, without crossing any edge, and without introducing the W subgraph and the B subgraph. Figure 5.13 shows an example of a red-maximal augmentation.

The following lemma summarizes the results of the augmentation step.

Lemma 5.3 *Let G be a 1-plane graph without the B subgraph or the W subgraph. Then there is a red-maximal augmentation G^+ of G without the B subgraph or the W subgraph, which can be computed in linear time.*

Properties of Red-Maximal 1-Plane Graphs

The structure of a red-maximal 1-plane graph is relatively simple; this simplifies the drawing algorithm. Let G^+ be a red-maximal 1-plane graph that does not contain the B subgraph or the W subgraph, and G^* be the planarization of G^+. Then G^+ and G^* have the following properties:

- If f is an internal face of G^* with no crossing vertex, then f is a 3-cycle.

- If f is an internal face of G^* with a crossing vertex, then f is either a 3-cycle, 4-cycle, or 5-cycle.
- If f is the outer face of G^*, then f has no crossing vertices.
- If f is the outer face of G^*, then f is either a 3-cycle or 4-cycle. If f is a 4-cycle, then it induces a 4-clique with a crossing.
- If γ is a crossing between edges (a, c) and (b, d), then there is a path P of red edges from a to b such that the cycle C in G^* formed by the edges (a, γ) and (γ, b), and P contains no vertices strictly inside C.

Step 2: Drawing Algorithm

The input of the drawing algorithm is a red-maximal augmentation G^+ without the B subgraph or the W subgraph. Let G_r be the subgraph of red edges of G^+. Based on the structural properties of G^+, both G^+ and G_r are biconnected. Therefore, the drawing algorithm uses the SPR tree, a simplified version of the SPQR tree [5] without Q-nodes.

Let $\sigma(\nu)$ denote the *skeleton* of node ν in the SPR tree, which has one of the three types: (i) S-node: $\sigma(\nu)$ is a simple cycle with at least three vertices; (ii) P-node: $\sigma(\nu)$ consists of two vertices connected by at least three edges; (iii) R-node: $\sigma(\nu)$ is a simple triconnected graph.

The algorithm uses the SPR tree of the red subgraph G_r of G^+, rooted at a node whose skeleton contains the vertices on the outer face. Let $\sigma(\nu)^-$ denote a graph after deleting the parent virtual edge from $\sigma(\nu)$. The algorithm uses a similar approach for star-shaped drawings of planar graphs [17], however, in a simplified way due to the nice properties of the red-maximal augmentation.

More specifically, the algorithm recursively processes each node ν in the SPR tree in a top-down manner, from the root node to the leaf nodes, as follows:

1. Construct a *convex drawing* D_ν of $\sigma(\nu)$ in a given convex polygon P_ν.
2. Re-insert crossing edges in the corresponding face of D_ν with straight-line edges.
3. For each child μ of ν, define a convex polygon P_μ and replace the corresponding virtual edge in D_ν with a drawing of $\sigma(\mu)$.

The algorithm uses a convex drawing algorithm of Chiba et al. [9] as a subroutine for drawing R-nodes, as follows: It takes a convex polygon P_ν and the plane graph $\sigma(\nu)$ as input, and computes a *convex* drawing D_ν of $\sigma(\nu)$. Since each face of D_ν is a convex polygon, we can re-insert the crossing edges using straight lines, without introducing any new crossings.

In fact, the algorithm processes each node ν differently, based on its type (i.e., R-node, S-node, and P-node). Here, we give a brief sketch for each case.

(i) R-node: First, construct a convex drawing D_ν of $\sigma(\nu)$ for the root R-node (respectively, $\sigma(\nu)^-$ for non-root R-node) inside a given convex polygon P_ν. Next, re-insert the crossing edges in the corresponding face in D_ν as straight-line segments.

After inserting crossing edges, define a drawing region and a convex polygon P_μ for drawing $\sigma(\mu)$ of each child node μ recursively. Figure 5.14 shows an example of a root R-node.

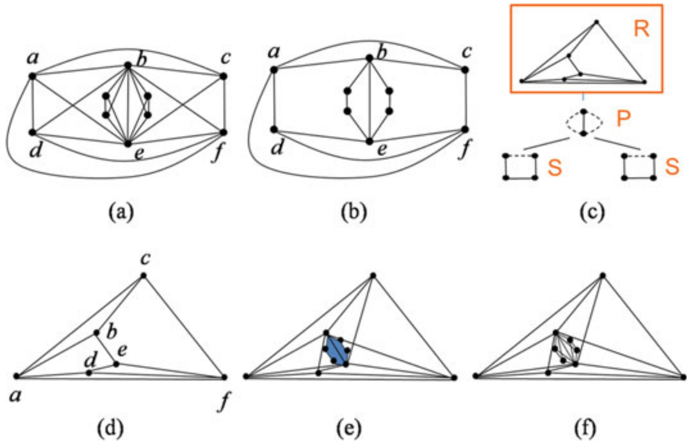

Fig. 5.14 Example of a root R-node ν: **a** G^+; **b** G_r; **c** SPR tree of G_r; **d** convex drawing of $\sigma(\nu)$; **e** re-insert crossing edges in a convex face and define a drawing area and convex polygon P_μ for drawing $\sigma(\mu)$ of a child node μ; **f** re-insert crossing edges inside P_μ

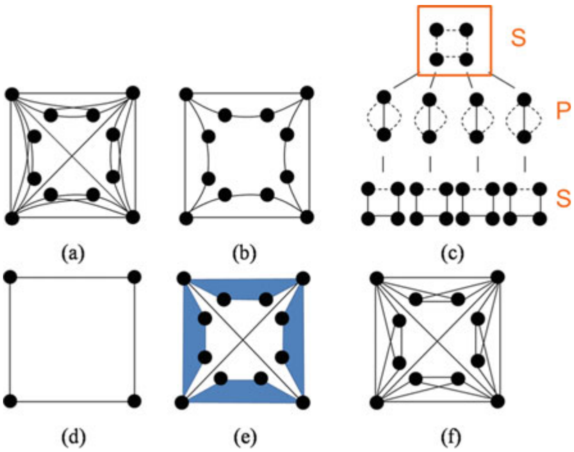

Fig. 5.15 Example of a root S-node ν: **a** G^+; **b** G_r; **c** SPR tree of G_r; **d** convex drawing of $\sigma(\nu)$; **e** re-insert crossing edges in a convex face and define a drawing area and convex polygon P_μ for drawing $\sigma(\mu)$ of child node μ; **f** re-insert crossing edges inside P_μ

(ii) S-node: If ν is a root S-node, draw $\sigma(\nu)$ as a triangle or a rectangle; re-insert the crossing edges, if $\sigma(\nu)$ is a 4-cycle. Then define a drawing region and a convex polygon P_μ for drawing $\sigma(\mu)$ of each child node μ recursively.

If ν is a non-root S-node, then we draw $\sigma(\nu)^-$ as a path. Then, the main task is to define a drawing area and a convex polygon P_μ for drawing $\sigma(\mu)$ of each child node μ recursively. Figure 5.15 shows an example of the root S-node.

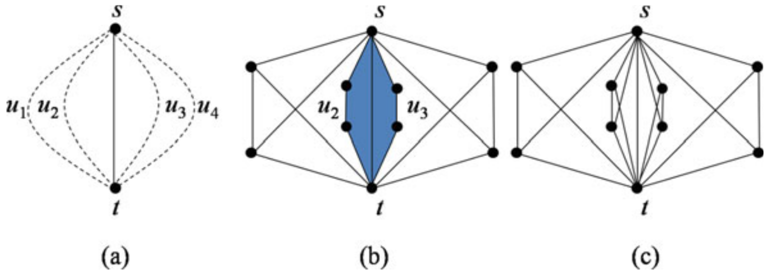

Fig. 5.16 Defining drawing area for the children of a P-node ν: **a** $\sigma(\nu)$; **b** define left trapezoids for μ_1 and μ_2, and right trapezoids for μ_3 and μ_4; re-insert crossing edges in a convex drawing of $\sigma(\mu_1)$ (respectively, $\sigma(\mu_4)$) and define a drawing area and convex polygon P_{μ_2} (respectively, P_{μ_3}) for drawing $\sigma(\mu_2)$ (respectively, $\sigma(\mu_3)$); **c** re-insert crossing edges in the drawing D_{μ_2} (respectively, D_{μ_3})

(iii) P-node: The main task for P-node is to define a drawing area and a convex polygon P_μ for drawing $\sigma(\mu)$ of each child node μ recursively. For R-node child μ, define P_μ as either a triangle or a rhombus; for S-node child μ, define P_μ as either a triangle or a trapezoid, based on the properties of the red-maximal augmentation.

Let vertices s and t be the separation pair of ν, and denote the virtual edges between s and t as $u_1, u_2, \ldots, u_m$, in left-to-right order, as in Fig. 5.16a. Denote the corresponding children of ν as $\mu_1, \mu_2, \ldots, \mu_m$. Suppose that the edge $e = (s, t)$ occurs between u_k and u_{k+1}. The polygons P_{μ_i} must be drawn based on the ordering: define a *left* triangle (or trapezoid) for μ_1, μ_2, ..., μ_k, and a *right* triangle (or trapezoid) for μ_{k+1}, μ_{k+2}, ..., μ_m to avoid edge crossings. See Fig. 5.16b.

First, draw $\sigma(\mu_1)$ inside the polygon P_{μ_1}, and re-insert crossing edges in the drawing D_{μ_1}. Then, define a drawing area for $\sigma(\mu_2)$ with a convex polygon P_{μ_2}, such that it does not cross any edges already drawn in D_{μ_1}. For an example, see Fig. 5.16b.

Next, draw $\sigma(\mu_2)$ inside the polygon P_{μ_2}, and re-insert crossing edges in the drawing D_{μ_2}. Repeat this process until we process $\sigma(\mu_k)$. Similarly, process μ_{k+1}, μ_{k+2}, ..., μ_m symmetrically, starting from μ_m and working toward μ_{k+1}. See Fig. 5.16c for an example.

When replacing each virtual edge in the convex drawing D_ν of $\sigma(\nu)$ with a drawing of $\sigma(\mu)$, where μ is a child node of ν, we can define a convex polygon P_μ for the boundary of $\sigma(\mu)$ thin enough not to create any new crossings.

The following lemma summarizes the results of the drawing step.

Lemma 5.4 *Let G^+ be a red-maximal 1-plane graph without the B subgraph or the W subgraph. Then there is a linear time algorithm to construct a straight-line 1-planar drawing of G^+.*

Exponential Area

It was also shown that some 1-plane graphs require exponential area for any straight-line grid 1-planar drawing. More specifically, for all $k > 1$, there is a 1-plane graph

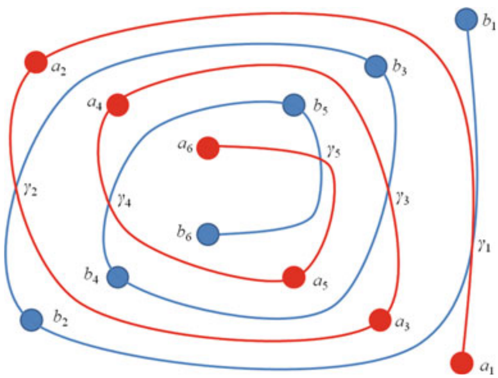

Fig. 5.17 A 1-plane graph G_k for which every straight-line grid 1-planar drawing has exponential area. Here, the case $k = 6$ is shown

G_k with $2k$ vertices and $2k - 2$ edges such that any straight-line grid (i.e., each vertex has integer coordinates) 1-planar drawing of G_k requires at least 2^{k-1} area. See Fig. 5.17 for an example, where $k = 6$.

5.5 Recent Progress

This chapter reviews the algorithmic results on 1-planar graphs. More specifically, we review three linear time algorithms for testing *maximal* 1-planar graphs with a given rotation system, testing *outer* 1-planar graphs, and drawing 1-plane graphs with straight-line edges.

We now briefly review recent results on related topics, mainly focusing on the algorithmic aspects.

- *Testing full-outer-2-planarity:* A graph is *fully-outer-2-planar* if it admits an outer-2-planar embedding (i.e., each vertex is placed on the outer boundary and no edge has more than two crossings) such that no crossing appears along the outer boundary.
  Hong and Nagamochi [19] showed that triconnected full-outer-2-planar graphs have a constant number of full-outer-2-planar embeddings. Based on these properties, linear time algorithms for testing full-outer-2-planarity of a connected, biconnected, and triconnected graph were presented. The algorithms also produce a full-outer-2-planar embedding of a graph, if it exists.
- *Re-embedding 1-plane graph:* Re-embedding of a 1-plane graph is to change a given 1-planar embedding with B or W subgraph to a new 1-planar embedding without the B subgraph or the W subgraph, by changing the rotation system or the outer face of the given 1-planar embedding, while preserving the same set of pairs of crossing edges.
  Hong and Nagamochi [18] presented a characterization of forbidden configuration (i.e., a given 1-plane graph can be re-embedded into a straight-line drawable 1-

planar embedding if and only if it does not contain the configuration). Based on the characterization, a linear time algorithm for finding a straight-line drawable 1-planar embedding or the forbidden configuration was presented.

- *Straight-line drawings of almost planar graphs:* The almost planar graph consists of a planar graph plus one edge, also called graphs with *1-skewness* (i.e., removal of an edge makes the graph planar).
  Eades et al. [11] presented a characterization of almost planar topological graphs that admit a straight-line drawing. Based on the characterization, linear time algorithms for testing whether an almost planar graph admits a straight-line drawing, and for constructing such a drawing if it exists, were presented. It was also shown that some almost planar graphs require an exponential area for any straight-line grid drawing.
- *Straight-line drawings of general embedded non-planar graphs:* Nagamochi [23] investigated the stretchability problem of a general embedded topological graph. He showed that there is a 3-planar embedding and quasi-planar embedding that admits no straight-line drawing, which cannot be characterized by forbidden configuration.
  He also considered a problem of whether a given embedded graph G admits a straight-line drawing under the same *frame*, which is defined by a fixed biconnected planar spanning subgraph of G. He presented forbidden configurations (i.e., a given embedding admits a straight-line drawing under the same frame if and only if it contains no forbidden configuration) for the problem.
  It was shown that if a given embedding is *quasi-planar* (i.e., no pairwise crossing edges) and its crossing-free edges induce a biconnected spanning subgraph, then the stretchability can be tested using forbidden configurations in polynomial time.

For the last decades, 1-planar graphs have been extensively studied and consequently many combinatorial and algorithmic questions are already solved. Many combinatorial results are also available for k-planar graphs, including structural properties, geometric representations, as well as the relationships between various beyond planar graphs. For details, we refer to corresponding chapters in this book.

However, many fundamental algorithmic questions on the other classes of beyond planar graphs are remained to be solved and deserve further investigation. For details, we refer to open problems listed in corresponding chapters in this book.

Acknowledgements This work is supported by ARC (Australian Research Council) Discovery Project grant.

References

1. Auer, C., Bachmaier, C., Brandenburg, F.J., Gleißner, A., Hanauer, K., Neuwirth, D., Reislhuber, J.: Outer 1-planar graphs. Algorithmica **74**(4), 1293–1320 (2016). https://doi.org/10.1007/s00453-015-0002-1

2. Auer, C., Brandenburg, F.J., Gleißner, A., Reislhuber, J.: 1-planarity of graphs with a rotation system. J. Graph Algorithms Appl. **19**(1), 67–86 (2015). https://doi.org/10.7155/jgaa.00347
3. Bannister, M.J., Cabello, S., Eppstein, D.: Parameterized complexity of 1-planarity. J. Graph Algorithms Appl. **22**(1), 23–49 (2018). https://doi.org/10.7155/jgaa.00457
4. Battista, G.D., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall (1999)
5. Battista, G.D., Tamassia, R.: On-line maintenance of triconnected components with spqr-trees. Algorithmica **15**(4), 302–318 (1996)
6. Borodin, O.V.: Solution of the Ringel problem on vertex-face coloring of planar graphs and coloring of 1-planar graphs. Metody Diskret. Analiz. **41**, 12–26, 108 (1984)
7. Borodin, O.V., Kostochka, A.V., Raspaud, A., Sopena, E.: Acyclic colouring of 1-planar graphs. Discret. Appl. Math. **114**(1–3), 29–41 (2001)
8. Cabello, S., Mohar, B.: Adding one edge to planar graphs makes crossing number and 1-planarity hard. SIAM J. Comput. **42**(5), 1803–1829 (2013). https://doi.org/10.1137/120872310
9. Chiba, N., Yamanouchi, T., Nishizeki, T.: Linear Time Algorithms for Convex Drawings of Planar Graphs. Progress in Graph Theory, pp. 153–173 (1984)
10. Eades, P., Hong, S., Katoh, N., Liotta, G., Schweitzer, P., Suzuki, Y.: A linear time algorithm for testing maximal 1-planarity of graphs with a rotation system. Theor. Comput. Sci. **513**, 65–76 (2013). https://doi.org/10.1016/j.tcs.2013.09.029
11. Eades, P., Hong, S., Liotta, G., Katoh, N., Poon, S.: Straight-line drawability of a planar graph plus an edge. In: Dehne, F., Sack, J., Stege, U. (eds.) Algorithms and Data Structures - 14th International Symposium, WADS 2015, Victoria, BC, Canada, August 5–7, 2015. Proceedings, Lecture Notes in Computer Science, vol. 9214, pp. 301–313. Springer (2015). https://doi.org/10.1007/978-3-319-21840-3_25
12. Fabrici, I., Madaras, T.: The structure of 1-planar graphs. Discret. Math. **307**(7–8), 854–865 (2007)
13. Fáry, I.: On straight line representations of planar graphs. Acta Sci. Math. Szeged **11**, 229–233 (1948)
14. de Fraysseix, H., Pach, J., Pollack, R.: How to draw a planar graph on a grid. Combinatorica **10**(1), 41–51 (1990). https://doi.org/10.1007/BF02122694
15. Grigoriev, A., Bodlaender, H.L.: Algorithms for graphs embeddable with few crossings per edge. Algorithmica **49**(1), 1–11 (2007). https://doi.org/10.1007/s00453-007-0010-x
16. Hong, S., Eades, P., Katoh, N., Liotta, G., Schweitzer, P., Suzuki, Y.: A linear-time algorithm for testing outer-1-planarity. Algorithmica **72**(4), 1033–1054 (2015). https://doi.org/10.1007/s00453-014-9890-8
17. Hong, S., Nagamochi, H.: An algorithm for constructing star-shaped drawings of plane graphs. Comput. Geom. **43**(2), 191–206 (2010). https://doi.org/10.1016/j.comgeo.2009.06.008
18. Hong, S., Nagamochi, H.: Re-embedding a 1-plane graph into a straight-line drawing in linear time. In: Hu, Y., Nöllenburg, M., (eds.) Graph Drawing and Network Visualization - 24th International Symposium, GD 2016, Athens, Greece, September 19–21, 2016, Revised Selected Papers, Lecture Notes in Computer Science, vol. 9801, pp. 321–334. Springer (2016). https://doi.org/10.1007/978-3-319-50106-2_25
19. Hong, S., Nagamochi, H.: A linear-time algorithm for testing full outer-2-planarity. Discret. Appl. Math. **255**, 234–257 (2019). https://doi.org/10.1016/j.dam.2018.08.018
20. Hong, S.H., Eades, P., Liotta, G., Poon, S.H.: Fáry's theorem for 1-planar graphs. In: Gudmundsson, J., Mestre, J., Viglas, T., (eds.) Proceedings of COCOON 2012, Lecture Notes in Computer Science, vol. 7434, pp. 335–346. Springer (2012)
21. Hudák, D., Madaras, T., Suzuki, Y.: On properties of maximal 1-planar graphs. Discuss. Math. Graph Theory **32**(4), 737–747 (2012). https://doi.org/10.7151/dmgt.1639
22. Korzhik, V.P., Mohar, B.: Minimal obstructions for 1-immersions and hardness of 1-planarity testing. J. Graph Theory **72**(1), 30–71 (2013)
23. Nagamochi, H.: Straight-line drawability of embedded graphs, Technical Report 2013–005. Kyoto University, Japan, Department of Applied Mathematics and Physics (2013)

24. Nishizeki, T., Rahman, M.S.: Planar Graph Drawing. Lecture Notes Series on Computing, vol. 12. World Scientific (2004). https://doi.org/10.1142/5648
25. Pach, J., Tóth, G.: Graphs drawn with few crossings per edge. Combinatorica **17**(3), 427–439 (1997)
26. Ringel, G.: Ein Sechsfarbenproblem auf der Kugel. Abh. Math. Sem. Univ. Hamburg **29**, 107–117 (1965)
27. Suzuki, Y.: Optimal 1-planar graphs which triangulate other surfaces. Discret. Math. **310**(1), 6–11 (2010). https://doi.org/10.1016/j.disc.2009.07.016
28. Suzuki, Y.: Re-embeddings of maximum 1-planar graphs. SIAM J. Discret. Math. **24**(4), 1527–1540 (2010). https://doi.org/10.1137/090746835
29. Thomassen, C.: Rectilinear drawings of graphs. J. Graph Theory **12**(3), 335–341 (1988)
30. Zhang, X., Wu, J.L.: On edge colorings of 1-planar graphs. Inform. Process. Lett. **111**(3), 124–128 (2011). http://dx.doi.org/10.1016/j.ipl.2010.11.001