Compact Drawings of 1-Planar Graphs with Right-Angle Crossings and Few Bends

Steven Chaplick, Fabian Lipp, Alexander Wolff, and Johannes Zink(✉)

Lehrstuhl für Informatik I, Universität Würzburg, Würzburg, Germany
{steven.chaplick,fabian.lipp,alexander.wolff,
johannes.zink}@uni-wuerzburg.de
http://www1.informatik.uni-wuerzburg.de/en/staff

Abstract. We study the following classes of beyond-planar graphs: 1-planar, IC-planar, and NIC-planar graphs. These are the graphs that admit a 1-planar, IC-planar, and NIC-planar drawing, respectively. A drawing of a graph is *1-planar* if every edge is crossed at most once. A 1-planar drawing is *IC-planar* if no two pairs of crossing edges share a vertex. A 1-planar drawing is *NIC-planar* if no two pairs of crossing edges share two vertices.

We study the relations of these beyond-planar graph classes to *right-angle crossing* (*RAC*) graphs that admit compact drawings on the grid with few bends. We present four drawing algorithms that preserve the given embeddings. First, we show that every n-vertex NIC-planar graph admits a NIC-planar RAC drawing with at most one bend per edge on a grid of size $\mathcal{O}(n) \times \mathcal{O}(n)$. Then, we show that every n-vertex 1-planar graph admits a 1-planar RAC drawing with at most two bends per edge on a grid of size $\mathcal{O}(n^3) \times \mathcal{O}(n^3)$. Finally, we make two known algorithms embedding-preserving; for drawing 1-planar RAC graphs with at most one bend per edge and for drawing IC-planar RAC graphs straight-line.

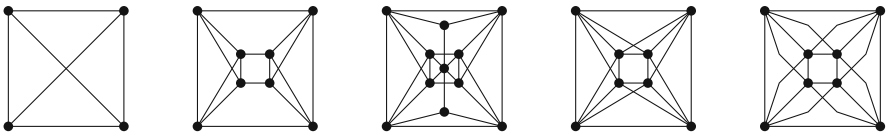
1 Introduction

In graph theory and graph drawing, *beyond-planar* graph classes have experienced increasing interest in recent years. A prominent example is the class of *1-planar graphs*, that is, graphs that admit a drawing where each edge is crossed at most once. The 1-planar graphs were introduced by Ringel [18] in 1965; Kobourov et al. [15] surveyed them recently. Another example that has received considerable attention are RAC_k *graphs*, that is, graphs that admit a poly-line drawing where all crossings are at right angles and each edge has at most k bends. The RAC_k graphs were introduced by Didimo et al. [7]. Using right-angle crossings and few bends is motivated by several cognitive studies suggesting a positive correlation between large crossing angles or small curve complexity and the readability of a graph drawing [13,14,17].

The full version of this paper is available on arXiv [4] and the appendices are given therein.

T. Biedl and A. Kerren (Eds.): GD 2018, LNCS 11282, pp. 137–151, 2018.
https://doi.org/10.1007/978-3-030-04414-5_10

We investigate the relationships between (certain subclasses of) 1-planar graphs and RAC_k graphs that admit drawings on a polynomial-size grid. The prior work and our contributions are summarized in Fig. 2. A broader overview of beyond-planar graph classes is given in a recent survey by Didimo et al. [8].



(a) RAC_0 drawing. (b) IC-planar drawing. (c) NIC-planar drawing. (d) 1-planar drawing. (e) 1-planar RAC_1 drawing.

Fig. 1. Examples of different types of drawings. Figure 1d and e show drawings of the same graph. Figure 1e is taken from the Annotated Bibliography on 1-Planarity [15].

Basic Terminology. A mapping Γ is called a *drawing* of the graph $G = (V, E)$ if each vertex $v \in V$ is mapped to a point in $\mathbb{R}^2$ and each edge uv is mapped to a simple open Jordan curve in $\mathbb{R}^2$ such that the endpoints of this curve are $\Gamma(u)$ and $\Gamma(v)$. For convenience, we will refer to the points and simple open Jordan curves of a drawing as vertices and edges. The topologically connected regions of $\mathbb{R}^2 \setminus \Gamma$ are the *faces* of Γ. The unbounded face of Γ is its *outer* face; the other faces are *inner* faces. Each face defines a circular list of bounding edges (resp. edge sides), which we call its *boundary list*. Two drawings of a graph G are *equivalent* when they have the same set of boundary lists for their inner faces and outer faces. Each equivalence class of drawings of G is an *embedding*. A *k-bend (poly-line) drawing* is a drawing in which every edge is drawn as a connected sequence of at most $k+1$ line segments. The (up to) k inner vertices of an edge connecting these line segments are called *bend points* or *bends*. A 0-bend drawing is more commonly referred to as a *straight-line* drawing. A *drawing on the grid of size $w \times h$* is a drawing where every vertex, bend point, and crossing point has integer coordinates in the range $[0, w] \times [0, h]$. In any drawing we require that vertices, bends, and crossings are pairwise distinct points. A drawing is *1-planar* if every edge is crossed at most once. A 1-planar drawing is *independent-crossing planar* (*IC-planar*) if no two pairs of crossing edges share a vertex. A 1-planar drawing is *near-independent-crossing planar* (*NIC-planar*) if any two pairs of crossing edges share at most one vertex. A drawing is *right-angle-crossing* (*RAC*) if (*i*) it is a poly-line drawing, (*ii*) no more than two edges cross in the same point, and (*iii*) in every crossing point the edges intersect at right angles. We further specialize the notion of RAC drawings. A drawing is RAC_k if it is RAC and k-bend; it is $\mathrm{RAC}^{\mathrm{poly}}$ if it is RAC and on a grid whose size is polynomial in its number of vertices. Examples for IC-planar, NIC-planar, 1-planar, and RAC drawings are given in Fig. 1. The *planar*, *1-planar*, *NIC-planar*, *IC-planar*, and *RAC_k* graphs are the graphs that admit a crossing-free, 1-planar, NIC-planar,

IC-planar, and RAC_k drawing, respectively. More specifically, $\mathrm{RAC}_k^{\mathrm{poly}}$ is the set of graphs that admit a $\mathrm{RAC}_k^{\mathrm{poly}}$ drawing. A *plane*, *1-plane*, *NIC-plane*, and *IC-plane* graph is a graph given with a specific planar, 1-planar, NIC-planar, and IC-planar embedding, respectively. In a 1-planar embedding the edge crossings are known and they are stored as if they were vertices. We will denote an embedded graph by $(G, \mathcal{E})$ where G is the graph and $\mathcal{E}$ is the embedding of this graph. For a point p in the plane, let $x(p)$ and $y(p)$ denote its x- and y-coordinate, respectively. Given two points p and q, we denote the straight-line segment connecting them by $\overline{pq}$ and its length, the Euclidean distance of p and q, by $\|\overline{pq}\|$.

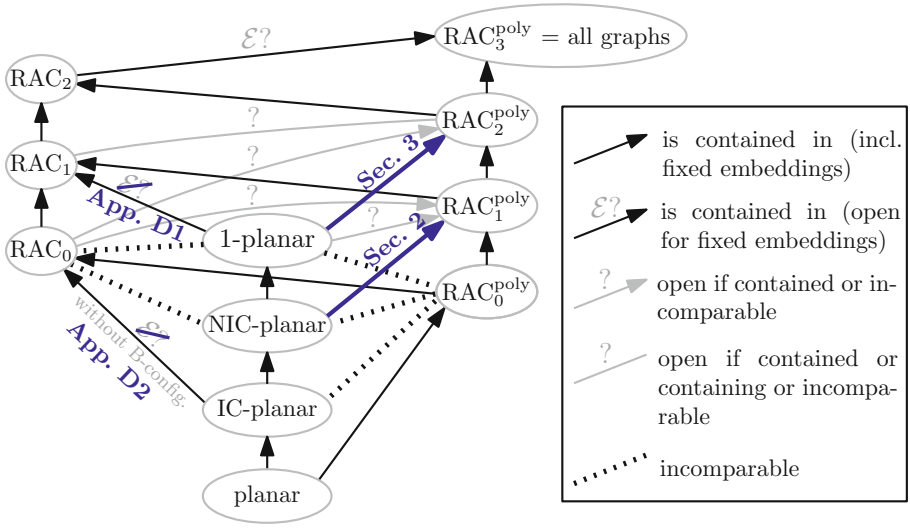


Fig. 2. Relating some classes of (beyond-)planar graphs and RAC graphs. Our main results are the containment relationships indicated by the thick blue arrows (Color figure online).

Previous Work. In the diagram in Fig. 2, we give an overview of the relationships between classes of 1-planar graphs and RAC_k graphs. Clearly, the planar graphs are a subset of the IC-planar graphs, which are a subset of the NIC-planar graphs, which are a subset of the 1-planar graphs. It is well known that every plane graph can be drawn with straight-line edges on a grid of quadratic size [10,19]. Every IC-planar graph admits an IC-planar RAC_0 drawing but not necessarily in polynomial area [3]. Moreover, there are graphs in $\mathrm{RAC}_0^{\mathrm{poly}}$ that are not 1-planar [9] and, therefore, also not IC-planar. The class of RAC_0 graphs is incomparable with the classes of NIC-planar graphs [1] and 1-planar graphs [9]. Bekos et al. [2] showed that every 1-planar graph admits a 1-planar RAC_1 drawing, but their recursive drawings may need exponential area. Every graph admits a RAC_3 drawing in polynomial area, but this does not hold if a given embedding of the graph must be preserved [7].

Our Contributions. We contribute four new results; two main results and two adaptations of prior results. First, we constructively show that every NIC-plane graph admits a RAC_1 drawing in quadratic area; see Sect. 2. This improves upon a side result by Liotta and Montecchiani [16], who showed that every IC-plane graph admits a RAC_2 drawing on a grid of quadratic size. Second, we constructively show that every 1-plane graph admits a RAC_2 drawing in polynomial area; see Sect. 3. Beside these two main results, we show how to preserve a given embedding when computing RAC drawings. Precisely, we show Theorem 1 in Appendix D.1 by adapting an algorithm of Bekos et al. [2] and we show Theorem 2 in Appendix D.2 by adapting an algorithm of Brandenburg et al. [3].

Theorem 1. *Any n-vertex 1-plane graph admits an embedding-preserving RAC_1 drawing. It can be computed in $\mathcal{O}(n)$ time.*

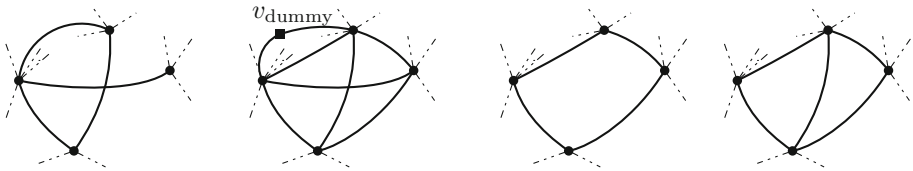
Theorem 2. *Any straight-line drawable n-vertex IC-plane graph admits an embedding-preserving RAC_0 drawing. It can be computed in $\mathcal{O}(n^3)$ time.*

2 NIC-Planar 1-Bend RAC Drawings in Quadratic Area

In this section we constructively show that quadratic area is sufficient for RAC_1 drawings of NIC-planar graphs. We prove the following.

Theorem 3. *Any n-vertex NIC-plane graph $(G, \mathcal{E})$ admits a NIC-planar RAC_1 drawing that respects $\mathcal{E}$ and lies on a grid of size $\mathcal{O}(n) \times \mathcal{O}(n)$. The drawing can be computed in $\mathcal{O}(n)$ time.*

Preprocessing. Our algorithm gets an n-vertex NIC-plane graph $(G, \mathcal{E})$ as input. We first aim to make $(G, \mathcal{E})$ biconnected and planar so that we can draw it using the algorithm by Harel and Sardas [11]. Around each crossing in $\mathcal{E}$, we insert up to four dummy edges to obtain *empty kites*. A *kite* is a K_4 that is embedded such that (i) every vertex lies on the boundary of the outer face, and (ii) there is exactly one crossing, which does not lie on the boundary of the outer face. A kite K as a subgraph of a graph H is said to be *empty* if there is no edge of $H \backslash K$ that is on an inner face of K or crosses edges of K. Inserting a dummy edge could create a pair of parallel edges. If this happens, we subdivide the original edge participating in this pair by a dummy vertex (see the transition from Fig. 3a – b). Note that we never create parallel dummy edges since G is NIC-planar. After this, we remove both crossing edges from each empty kite and obtain *empty quadrangles* (see Fig. 3c). We store each such empty quadrangle in a list Q. At the end of the preprocessing, we make the resulting plane graph biconnected via, e.g., the algorithm of Hopcroft and Tarjan [12]. Since each empty quadrangle is contained in a biconnected component, no edges are inserted into it. Let $(G', \mathcal{E}')$ be the resulting plane biconnected graph.



(a) crossing as it initially appears (b) empty kite and subdivided original edge (c) empty quadrangle (d) divided quadrangle

Fig. 3. Modifying the crossings and computing the BCO.

Drawing Step. Now, we draw a graph that we obtain from $(G', \mathcal{E}')$ by first producing a *biconnected canonical ordering (BCO)*[1]. We use the algorithm by Harel and Sardas [11], which is a generalization of the algorithm of Chrobak and Payne [5], which in turn is based on the shift algorithm of de Fraysseix et al. [10]. The algorithm of Harel and Sardas consists of two phases. Given a plane biconnected graph H, in the first phase a BCO Π of the vertices in H is computed. In the second phase, H is drawn according to Π on a grid of size $(2|V(H)|-4) \times (|V(H)|-2)$. Unlike the classical shift algorithm, the algorithm of Harel and Sardas computes the (biconnected) canonical ordering *bottom-up*, which we will exploit here. Let $\Pi_k = (v_1, \ldots, v_k)$ be a partial BCO of H after step k, and let H_k be the plane subgraph of H induced by Π_k. We say that a vertex u is *covered* by v_k if u is on the boundary of the outer face of H_{k-1}, but not on that of H_k.

We perform the following additional operations when we compute the BCO $\hat{\Pi}$. Whenever we reach an empty quadrangle $q = (a, b, c, d)$ of the list Q for the first time, i.e., when the first vertex of q—say a—is added to the BCO, we insert an edge inside q from a to the vertex opposite a in q, that is, to c. We call the resulting structure a *divided quadrangle* (see Fig. 3d). In two special cases, we perform further modifications of the graph. They will help us to guarantee a correct reinsertion of the crossing edges in the next step of the algorithm. Namely, when we encounter the last vertex $v_{\text{last}} \in \{b, c, d\}$ of q, we distinguish three cases.

Case 1: $v_{\text{last}} = c$ (see Fig. 4a). Here, no operations are performed.

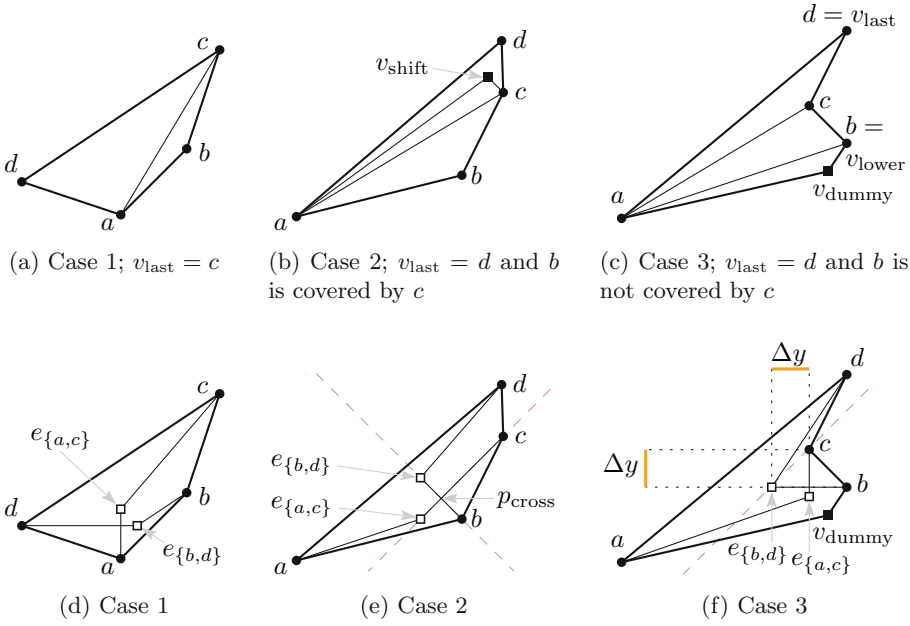
Case 2: $v_{\text{last}} \in \{b, d\}$, and the other of $\{b, d\}$ is covered by c (see Fig. 4b).
We insert a dummy vertex v_{shift}, which we call *shift vertex*, into the current BCO directly before v_{last} and make it adjacent to a and c. Observe that, if v_{shift} is the k-th vertex in $\hat{\Pi}$, this still yields a valid BCO since v_{shift} has two

[1] BCOs are a generalization of canonical orderings that assume only biconnectivity (instead of triconnectivity). In a BCO of a plane graph H, the subgraph H_k of H induced by $v_1, \ldots, v_k$ is connected, the edge v_1v_2 lies on the boundary of the outer face and all vertices in $H - H_k$ lie within the outer face of H_k. For $k > 2$, the vertex v_k has one or more neighbors in H_{k-1}. If v_k has exactly one neighbor u in H_{k-1}, then it has a legal support on the outer face of H_{k-1}, i.e., in the circular order of adjacent vertices around u, it follows or precedes a vertex in H_{k-1}.

neighbors in $\hat{\Pi}_{k-1}$ and is on the outer face of the subgraph induced by $\hat{\Pi}_{k-1}$. Later, we will remove v_{shift}, but for now it forces the algorithm of Harel and Sardas to shift a and c away from each other before v_{last} is added.

Case 3: $v_{\text{last}} \in \{b, d\}$, and neither b nor d is covered by c (see Fig. 4c).
Let $\{v_{\text{lower}}\} = \{b, d\} \setminus v_{\text{last}}$. We subdivide the edge av_{lower} via a dummy vertex v_{dummy}. If av_{lower} is an original edge of the input graph, this edge will be bent at v_{dummy} in the final drawing. We insert v_{dummy} into the current BCO directly before v_{lower}. To obtain a divided quadrangle again, we insert the dummy edge av_{lower}, which we will remove before we reinsert the crossing edges. This will give us some extra space inside the triangle $(a, v_{\text{dummy}}, v_{\text{lower}})$ for a bend point. Inserting v_{dummy} as k-th vertex into $\hat{\Pi}$ keeps $\hat{\Pi}$ valid since v_{dummy} uses the support edge incident to a that would have been covered by v_{lower} otherwise. Then, v_{lower} has at least two neighbors in $\hat{\Pi}_k$, namely a and v_{dummy}.

We draw the resulting plane biconnected $\hat{n}$-vertex graph $(\hat{G}, \hat{\mathcal{E}})$ according to its BCO $\hat{\Pi}$ via the algorithm by Harel and Sardas and obtain a crossing-free drawing $\hat{\Gamma}$. We do not modify the actual drawing phase.



(a) Case 1; $v_{\text{last}} = c$

(b) Case 2; $v_{\text{last}} = d$ and b is covered by c

(c) Case 3; $v_{\text{last}} = d$ and b is not covered by c

(d) Case 1

(e) Case 2

(f) Case 3

Fig. 4. Divided quadrangles produced in the three cases of the drawing step (a)–(c) and the crossing edges after the reinsertion step (d)–(f) in our algorithm. For orientation, lines with slope 1 or −1 are dashed violet. (Color figure online)

Postprocessing (Reinserting the Crossing Edges). We refine the underlying grid of $\hat{\Gamma}$ by a factor of 2 in both dimensions. Let $q = (a, b, c, d)$ be a quadrangle in Q,

where a is the first and v_{last} the last vertex in $\hat{\Pi}$ among the vertices in q. From q, we first remove the chord edge ac and obtain an empty quadrangle. Then, we distinguish three cases for reinserting the crossing edges that we removed in the preprocessing. These are the same cases as in the description of the modified computation of the BCO before. In this case distinction we omit some lengthy but straight-forward calculations; see Zink's master's thesis [21] for the details.

Case 1: $v_{\text{last}} = c$ (see Fig. 4a).
Since c is adjacent to a, b, and d in $\hat{G}$, it has the largest y-coordinate among the vertices in q. Assume that $y(d)$ is smaller or equal to $y(b)$ since the other case is symmetric. An example of a quadrangle in this case before and after the reinsertion of the crossing edges is given in Fig. 4a and d, respectively. We will have a crossing point at $(x(a), y(d))$. To this end, we insert the edge ac with a bend at $e_{ac} = (x(a), y(d)+1)$ and we insert the edge bd with a bend at $e_{bd} = (x(a)+1, y(d))$. Clearly the crossing is at a right angle. Observe that q is convex since c is the last drawn vertex of q and c is adjacent to b, a, and d in this circular order in the embedding and observe that both bend points lie inside q. Therefore, it follows that both crossing edges lie completely inside q.

Case 2: $v_{\text{last}} \in \{b, d\}$, and the other of $\{b, d\}$ is covered by c (see Fig. 4b).
Assume that $y(d) > y(b)$; the other case is symmetric. An example of a quadrangle in this case before and after the reinsertion of the crossing edges is given in Fig. 4b and e, respectively. We remove v_{shift} in addition to removing the edge ac. We define the crossing point $p_{\text{cross}} = (x_{\text{cross}}, y_{\text{cross}})$ as the intersection point of the lines with slope 1 and -1 through c and b, respectively. The coordinates of this crossing point are $x_{\text{cross}} = (x(c) - y(c) + x(b) + y(b))/2$ and $y_{\text{cross}} = (-x(c) + y(c) + x(b) + y(b))/2$. Since we refined the grid by a factor of 2 in each dimension, the above coordinates are both integers. We place the two bend points onto the same lines at the closest grid points that are next to p_{cross}, i.e., we draw the edge ac with a bend point at $e_{ac} = (x_{\text{cross}} - 1, y_{\text{cross}} - 1)$ and we insert the edge bd with a bend point at $e_{bd} = (x_{\text{cross}} - 1, y_{\text{cross}} + 1)$. We do not intersect or touch the edge ad because we shifted a far enough away from c by the extra shift due to v_{shift}. Moreover, the points e_{ac} and p_{cross} on the line with slope 1 through c are inside the empty quadrangle q since b is covered by c (then b is below the line with slope 1 through c) and $y(b)$ is at most equal to $y(e_{ac})$.

Case 3: $v_{\text{last}} \in \{b, d\}$, and neither b nor d is covered by c (see Fig. 4c).
Assume that $y(d) > y(b)$; again, the other case is symmetric. An example of a quadrangle in this case before and after the reinsertion of the crossing edges is given in Figs. 4c and f, respectively. Note that the edge ab is a dummy edge, which we inserted during the computation of $\hat{\Pi}$, and next to this edge, there is the path $av_{\text{dummy}}b$. This path is the former edge ab. We will reinsert the edges ac and bd such that they cross in $(x(c), y(b))$. We will bend the edge bd on the line with slope 1 through c at $y = y(b)$ because from this point we always "see" d inside q. So, we define $x_{\text{bend}} := x(c) - \Delta y$ with $\Delta y := y(c) - y(b)$. First, we remove the dummy edge ab. Second, we insert the edge ac with a bend point at $e_{ac} = (x(c), y(b) - 1)$. Third, we insert the edge bd with a bend

point at $e_{bd} = (x_{\text{bend}}, y(b))$. Note that e_{ac} might be below the straight-line segment $\overline{ab}$ since a could have been shifted far away from c. However, e_{ac} cannot be on or below the path $av_{\text{dummy}}b$ because $y(v_{\text{dummy}}) < y(e_{ac})$ and the slope of the line segment $\overline{v_{\text{dummy}}b}$ is either greater than 1 or negative. Therefore, the crossing edges ac and bd lie completely inside the pentagonal face $(a, v_{\text{dummy}}, b, c, d)$.

Result. After we have reinserted the crossing edges into each quadrangle of Q, we remove all dummy edges and transform the remaining dummy vertices to bend points. The resulting drawing Γ is a RAC_1 drawing that preserves the embedding of the NIC-plane input graph $(G, \mathcal{E})$. In Appendix A (p. 15), we bound the size of the grid that our drawings need, as follows.

Lemma 4. *Every vertex, bend point, and crossing point of the drawing returned by our algorithm lies on a grid of size at most* $(16n - 32) \times (8n - 16)$.

The shift algorithm of Harel and Sardas runs in linear time [11]. Also, our additional operations can be performed in linear time [21]. This proves Theorem 3. We give a full example of a NIC-plane RAC_1 drawing generated by a Java implementation of our algorithm in Figs. 9 and 10 in Appendix B.

3 1-Planar 2-Bend RAC Drawings in Polynomial Area

In this section we constructively prove the following.

Theorem 5. *Any n-vertex 1-plane graph $(G, \mathcal{E})$ admits a 1-planar RAC_2 drawing that respects $\mathcal{E}$ and lies on a grid of size $\mathcal{O}(n^3) \times \mathcal{O}(n^3)$. The drawing can be computed in $\mathcal{O}(n)$ time.*

The idea of our algorithm is to draw a slightly modified, planarized version of the 1-plane input graph with a variant of the shift algorithm (by Harel and Sardas [11]) and then "manually" redraw the crossing edges so that they cross at right angles and have at most two bends each. The difficulty is to find grid points for the bend points and the crossings so that the redrawn edges do not touch or cross the surrounding edges drawn by the shift algorithm. To this end, we refine our grid and place the middle part of each crossing edge onto a horizontal or vertical grid line so that the edge crossings are at right angles.

Preprocessing. Our algorithm gets an n-vertex 1-plane graph $(G, \mathcal{E})$ as input. First, we planarize G by replacing each crossing point by a vertex (see Fig. 5a). We will refer to them as *crossing vertices*. Second, we enclose each crossing vertex by a *subdivided kite*, which is an empty kite where the four boundary edges are subdivided by a vertex (see Fig. 5b). We use subdivided kites instead of empty kites to maintain the embedding and to avoid adding parallel edges. Third, we make the graph biconnected using, e.g., the algorithm of Hopcroft and Tarjan [12]. Note that we do not insert edges into inner faces of subdivided kites because all vertices and edges of a subdivided kite are in the same biconnected

(a) Planarized crossing where the crossing point became a crossing vertex c.

(b) Enclosing the crossing vertex c by a subdivided kite.

Fig. 5. A crossing point is replaced by a crossing vertex c and we insert four 2-paths of two dummy edges and a dummy vertex to induce a subdivided kite at each crossing. The vertices d_1, d_2, d_3, and d_4 are the dummy vertices of these 2-paths.

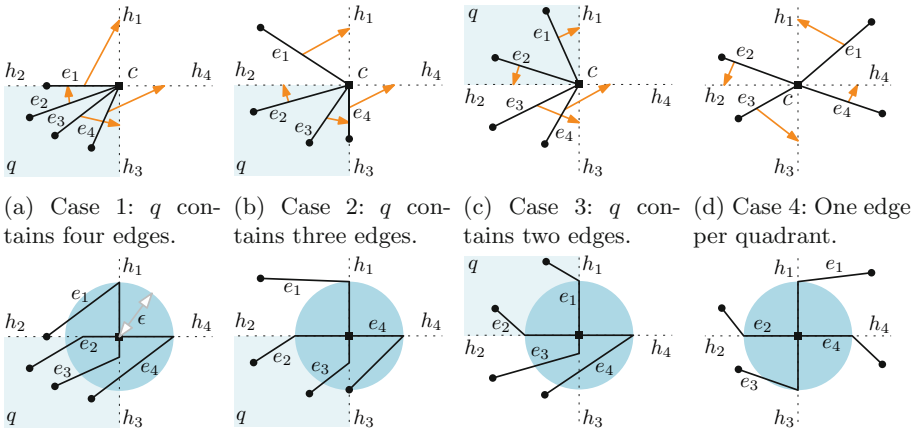
component. After these three steps, we have a biconnected plane graph $(G', \mathcal{E}')$. We draw $(G', \mathcal{E}')$ using the algorithm of Harel and Sardas [11]. This algorithm returns a crossing-free straight-line drawing Γ' of $(G', \mathcal{E}')$, whose vertices lie on a grid of size $(2n' - 4) \times (n' - 2)$, where n' is the number of vertices of G'.

Assignment of Edges to Axis-Parallel Half-Lines. For each crossing vertex c there are four incident edges in G'. They correspond to two edges of G. Consider the circular order around c in $(G', \mathcal{E}')$. The first and the third edge incident to c correspond to one edge in $(G, \mathcal{E})$; symmetrically, the second and fourth incident edge correspond to one edge. To obtain a RAC drawing from this, we redraw each of the four edges around c. Consider an edge ac from a vertex a of the subdivided kite to the crossing vertex c. This edge is then redrawn with a bend point b that lies on an axis-parallel line through c. For an example how a crossing in Γ' is replaced by a RAC crossing, see the transition from Fig. 8a to f. In order to obtain a right-angle crossing, we bijectively assign the four incident edges to the four axis-parallel half-lines originating in c. We call such a mapping an *assignment*. We do not take an arbitrary assignment, but take care to avoid extra crossings with edges that are redrawn or previously drawn. We call an assignment A *valid* if there is a way to redraw each edge e with one bend so that the bend point of e lies on the half-line $A(e)$ and the resulting drawing is plane.

To ensure that our valid assignment can be realized on a small grid, we introduce further criteria. We say that an edge e_1 *depends* on another edge e_2 with respect to an assignment A if e_2 lies in the angular sector between e_1 and the half-line $A(e_1)$. In Fig. 6a, for example, the edge e_3 depends on e_4 and e_2 depends on e_1, but e_1 and e_4 do not depend on any edge. We call edges (such as e_1 and e_4) that do not depend on other edges *independent*. We define the *dependency depth* of an assignment to be the largest integer k with $0 \leq k \leq 3$ such that there is a chain of $k+1$ edges $e_1, e_2, \ldots, e_{k+1}$ incident to c such that e_1 depends on e_2 and $\ldots$ and e_k depends on e_{k+1}, but there is no such chain of $k+2$ edges. For example, in Fig. 6a, b, and c, the assignment has a dependency depth of 1, whereas in Fig. 6d, the assignment has a dependency depth of 0. Showing that there is a valid assignment of dependency depth at most 1 will imply the existence of an appropriate set of grid points for the bend points as

formalized in Lemmas 7 and 8. In fact, as we will see in the discussion below, if we could avoid dependencies, our drawing would fit on a grid of size $\mathcal{O}(n^2) \times \mathcal{O}(n^2)$. Unfortunately, with our current approach this seems to be unavoidable.

We now construct an assignment that we will show in Lemma 6 to be valid and to have dependency depth at most 1. The four cases of our assignment are given in order of priority. Note that, in Cases 1 and 2, our assignment always contains dependencies; see Fig. 6a and b. Note further that it is enough to specify the assignment of one edge; the remaining assignment is determined since the circular orders of the edges and the assigned half-lines must be the same.



(a) Case 1: q contains four edges. (b) Case 2: q contains three edges. (c) Case 3: q contains two edges. (d) Case 4: One edge per quadrant.

(e) Case 1. (f) Case 2. (g) Case 3. (h) Case 4.

Fig. 6. The four cases of our assignment procedure: (a)–(d) indicate the assignment with orange arrows and show that the dependency depth is always at most 1, (e)–(f) show that the assignment is valid; the radius of the light blue disk is ϵ.

Case 1: There is a quadrant q that contains all four incident edges; see Fig. 6a. Take the two "inner" edges in q and assign them to the two half-lines that bound q, while keeping the circular order.

Case 2: There is a quadrant q that contains three incident edges; see Fig. 6b. Consider the edge outside q, say e_1, and assign it to the closest half-line h_i that does not bound q.

Case 3: There is a quadrant q that contains two incident edges; see Fig. 6c. Assign the incident edges in q to their closest half-lines.

Case 4: Each quadrant contains exactly one incident edge; see Fig. 6d. Assign each edge to its closest half-line in counter-clockwise direction.

See also Appendix C, where we prove the following lemma on p. 16.

Lemma 6. *Our assignment procedure returns a valid assignment with dependency depth at most 1.*

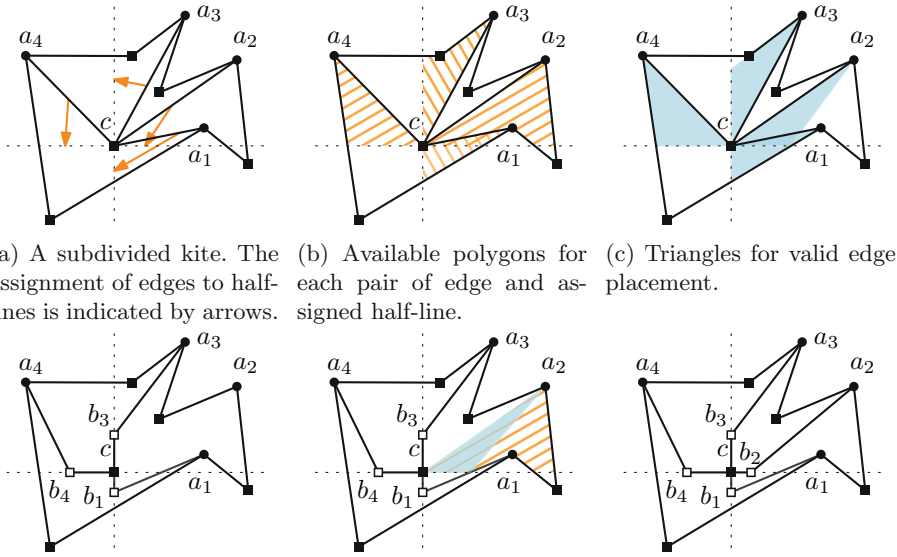
Note that Lemma 6 already gives us a RAC_2 drawing of the input graph, but in order to get a (good) bound on the grid size of the drawing, we have to place the bend points on a grid that is as coarse as possible, but still fine enough to provide us with grid points where we need them: on the half-lines emanating from the crossing vertices. This is what the remainder of this section is about.

(a) available polygon (b) triangle for valid edge placement given points p and q

Fig. 7. Example of an available polygon in which we determine the points p and q and with them the triangle for valid edge placement and the line segment $\overline{qc}$.

Placement of Bend Points on the Grid. In Γ', we have a drawing of a subdivided kite for every crossing in the 1-plane input graph. It is an octagon with a central crossing vertex c of degree four in its interior. For an example, see Fig. 8a. We will redraw the straight-line edges between c and its four adjacent vertices as 1-bend edges according to the assignment A computed in the previous step. The segment of such a 1-bend edge ac that ends at c will lie on the axis-parallel half-line $A(ac)$. If we pair and concatenate the 1-bend edges that enter c from opposite sides, we obtain two 2-bend edges and a right-angle crossing in c; see Fig. 8f. It remains to show how the bend points for the edges are placed on the grid. We proceed as follows.

First, we determine for each edge ac incident to a crossing vertex c the available region into which we can redraw ac with a bend b on $A(ac)$. The region between $\overline{ac}$ and the half-line $A(ac)$ inside the subdivided kite defines an *available polygon*. Examples of such an available polygon are given in Figs. 7a and 8b. Note that the available polygons might overlap (as they do once in Fig. 8b). Observe that there is only a triangle inside each available polygon in which the new line segment $\overline{ab}$ can be placed. Such a *triangle for valid edge placement* is determined by a, c and a corner point p of the available polygon. The point p is the corner point (excluding a and c) for which the angle between $\overline{ac}$ and $\overline{ap}$ inside the available polygon is the smallest. These triangles for valid edge placement are depicted in Figs. 7b and 8c. Again, they might overlap. Observe that in such a triangle, the angle at a cannot become arbitrarily small because every determining point lies on a grid point. Let q be the intersection point of the line through $\overline{ap}$ and the half-line $A(ac)$. One can see q as the projection of p onto $A(ac)$ seen from a. Note that we have a degenerated case if $a \in A(ac)$. Then, the available polygon has no area and equals the line segment $\overline{ac}$. In this case let $a = p = q$. Moreover, note that p can be equal to q because the intersection of



(a) A subdivided kite. The assignment of edges to half-lines is indicated by arrows.

(b) Available polygons for each pair of edge and assigned half-line.

(c) Triangles for valid edge placement.

(d) After the insertion of the bend points of the three independent edges.

(e) Available polygon and triangle for valid edge placement for the edge a_2c which depends on a_1c.

(f) Result after the insertion of the bend point b_2.

Fig. 8. Transformation from a planarized crossing to a RAC_2 crossing.

$A(ac)$ and an edge of the subdivided kite is also a corner point of the available polygon. This is the only case where p may not be a grid point.

We will place the bend point b onto the line segment $\overline{qc}$, but observe that the triangles for valid edge placement of two edges e_1 and e_2 might overlap if e_1 depends on e_2 in A. To solve this, we first draw the independent edges, then recompute the available polygons and the triangles for valid edge placement for the other edges, and finally draw those edges. Remember that our assignment procedure returns only assignments with dependency depth at most 1. Let Γ' be drawn on a grid of size $\tilde{n} \times \tilde{n}$. We refine the grid by a factor of $\tilde{n}$ in each dimension. The next step in our algorithm relies on the following lemma (which we prove in Appendix C, p. 19).

An important tool in our analysis will be the so-called *Farey sequence* [20] of order $\tilde{n} - 1$, which is the sequence of all reduced fractions from 0 to 1 with numerator and denominator being positive integers bounded by $\tilde{n} - 1$.

Lemma 7. *For any independent edge ac, the interior of the line segment $\overline{qc}$ contains at least one grid point of the refined $\tilde{n}^2 \times \tilde{n}^2$ grid.*

Using Lemma 7, we pick for each independent edge any grid point of $\overline{qc}$, place a bend point b on it, and replace the segment $\overline{ac}$ by the two segments $\overline{ab}$ and $\overline{bc}$. In Fig. 8c, the edges a_1c, a_3c, and a_4c are independent, but a_2c depends on a_1c.

We again refine the grid by a factor of $\tilde{n}$ in each dimension. The grid size is now $\tilde{n}^3 \times \tilde{n}^3$. For the remaining edges incident to a crossing vertex c, we compute new available polygons and triangles for valid edge placement since we need to take the 1-bend edges into account that were inserted in the previous step. Now the following lemma (proved in Appendix C, p. 22) yields grid points for the bend points of the remaining edges.

Lemma 8. *After having redrawn the independent edges, the interior of the line segment $\overline{qc}$ of each edge depending on an independent edge contains at least one grid point of the refined $\tilde{n}^3 \times \tilde{n}^3$ grid.*

For each remaining edge incident to a crossing vertex c we pick any grid point of its line segment $\overline{qc}$ and place a bend point b on it. Again, we replace $\overline{ac}$ by the two line segments $\overline{ab}$ and $\overline{bc}$.

Result. Finally, we remove the dummy edges and dummy vertices that bound the subdivided kites and interpret the crossing vertices as crossing points. We return the resulting RAC_2 drawing Γ. It is drawn on a grid of size $(8n'^3 - 48n'^2 + 96n' - 64) \times (4n'^3 - 24n'^2 + 48n' - 32)$, where n' is the number n of vertices of G plus 5 times the number of crossings $\mathrm{cr}(\mathcal{E})$ in $\mathcal{E}$. Note that $\mathrm{cr}(\mathcal{E}) \leq n - 2$ for 1-plane graphs [6]. If we ignore the bend points, the drawing is on a grid of size $(2n' - 4) \times (n' - 2)$, i.e., its size is quadratic. Again, the algorithm by Harel and Sardas [11] and our modification run in linear time. Therefore, we conclude the correctness of Theorem 5.

4 Conclusion and Open Questions

We have shown that any n-vertex NIC-plane graph admits a $\mathrm{RAC}_1^{\mathrm{poly}}$ drawing in $\mathcal{O}(n^2)$ area and that any n-vertex 1-plane graph admits a $\mathrm{RAC}_2^{\mathrm{poly}}$ drawing in $\mathcal{O}(n^6)$ area. We have also shown how to adjust two existing algorithms for drawing certain 1-planar graphs such that their embedding is preserved. More precisely, we have proved that any 1-plane graph admits a RAC_1 drawing. This answers an open question explicitly asked by the authors of the original algorithm [2]. We have also proved that any straight-line drawable IC-plane graph admits a RAC_0 drawing, where the original algorithm did not necessarily preserve the embedding [3]. The diagram in Fig. 2 leaves some open questions. Does any 1-planar graph admit a $\mathrm{RAC}_1^{\mathrm{poly}}$ drawing? Can we draw any graph in RAC_0 with only right-angle crossings in polynomial area when we allow one or two bends per edge? What is the relationship between RAC_1 and $\mathrm{RAC}_2^{\mathrm{poly}}$? Can we compute $\mathrm{RAC}_2^{\mathrm{poly}}$ drawings of 1-plane graphs in $o(n^6)$ area?

References

1. Bachmaier, C., Brandenburg, F.J., Hanauer, K., Neuwirth, D., Reislhuber, J.: NIC-planar graphs. Discrete Appl. Math. **232**, 23–40 (2017). https://doi.org/10.1016/j.dam.2017.08.015
2. Bekos, M.A., Didimo, W., Liotta, G., Mehrabi, S., Montecchiani, F.: On RAC drawings of 1-planar graphs. Theor. Comput. Sci. **689**, 48–57 (2017). https://doi.org/10.1016/j.tcs.2017.05.039
3. Brandenburg, F.J., Didimo, W., Evans, W.S., Kindermann, P., Liotta, G., Montecchiani, F.: Recognizing and drawing IC-planar graphs. Theor. Comput. Sci. **636**, 1–16 (2016). https://doi.org/10.1016/j.tcs.2016.04.026
4. Chaplick, S., Lipp, F., Wolff, A., Zink, J.: Compact drawings of 1-planar graphs with right-angle crossings and few bends. Arxiv report (2018). http://arxiv.org/abs/1806.10044v4
5. Chrobak, M., Payne, T.H.: A linear-time algorithm for drawing a planar graph on a grid. Inf. Process. Lett. **54**(4), 241–246 (1995). https://doi.org/10.1016/0020-0190(95)00020-D
6. Czap, J., Hudák, D.: On drawings and decompositions of 1-planar graphs. Electr. J. Comb. **20**(2), 54 (2013). http://www.combinatorics.org/ojs/index.php/eljc/article/view/v20i2p54
7. Didimo, W., Eades, P., Liotta, G.: Drawing graphs with right angle crossings. Theor. Comput. Sci. **412**(39), 5156–5166 (2011). https://doi.org/10.1016/j.tcs.2011.05.025
8. Didimo, W., Liotta, G., Montecchiani, F.: A survey on graph drawing beyond planarity. Arxiv report (2018). http://arxiv.org/abs/1804.07257
9. Eades, P., Liotta, G.: Right angle crossing graphs and 1-planarity. Discrete Appl. Math. **161**(7–8), 961–969 (2013). https://doi.org/10.1016/j.dam.2012.11.019
10. 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
11. Harel, D., Sardas, M.: An algorithm for straight-line drawing of planar graphs. Algorithmica **20**(2), 119–135 (1998). https://doi.org/10.1007/PL00009189
12. Hopcroft, J.E., Tarjan, R.E.: Algorithm 447: efficient algorithms for graph manipulation. Commun. ACM **16**(6), 372–378 (1973). https://doi.org/10.1145/362248.362272
13. Huang, W., Eades, P., Hong, S.: Larger crossing angles make graphs easier to read. J. Vis. Lang. Comput. **25**(4), 452–465 (2014). https://doi.org/10.1016/j.jvlc.2014.03.001
14. Huang, W., Hong, S., Eades, P.: Effects of crossing angles. In: Proceedings IEEE VGTC Pacific Visualization Symposium (PacificVis 2008), pp. 41–46 (2008). https://doi.org/10.1109/PACIFICVIS.2008.4475457
15. Kobourov, S.G., Liotta, G., Montecchiani, F.: An annotated bibliography on 1-planarity. Comput. Sci. Rev. **25**, 49–67 (2017). https://doi.org/10.1016/j.cosrev.2017.06.002
16. Liotta, G., Montecchiani, F.: L-visibility drawings of IC-planar graphs. Inf. Process. Lett. **116**(3), 217–222 (2016). https://doi.org/10.1016/j.ipl.2015.11.011
17. Purchase, H.: Which aesthetic has the greatest effect on human understanding? In: DiBattista, G. (ed.) GD 1997. LNCS, vol. 1353, pp. 248–261. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63938-1_67
18. Ringel, G.: Ein Sechsfarbenproblem auf der Kugel. Abh. Math. Seminar Univ. Hamburg **29**(1–2), 107–117 (1965)

19. Schnyder, W.: Embedding planar graphs on the grid. In: Johnson, D.S. (ed.) Proceedings 1st ACM-SIAM Symposium Discrete Algorithms (SODA 1990), pp. 138–148 (1990). http://dl.acm.org/citation.cfm?id=320176.320191
20. Wikipedia contributors: Farey sequence—Wikipedia, the free encyclopedia (2018). https://en.wikipedia.org/w/index.php?title=Farey_sequence&oldid=844932264. Accessed 8 June 2018
21. Zink, J.: 1-planar RAC drawings with bends. Master's thesis, Institut für Informatik, Universität Würzburg (2017). http://www1.pub.informatik.uni-wuerzburg.de/pub/theses/2017-zink-master.pdf