Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 23, no. 4, pp. 653–682 (2019)
DOI: 10.7155/jgaa.00506

How to Draw a Planarization

Thomas Bläsius[1] *Marcel Radermacher*[2] *Ignaz Rutter*[3]

[1]Research Group Algorithm Engineering, Hasso Plattner Institute, Germany
[2]Faculty of Informatics, Karlsruhe Institute of Technology (KIT), Germany
[3]Department of Computer Science and Mathematics, University of Passau, Germany

Abstract

We study the problem of computing straight-line drawings of non-planar graphs with few crossings. We assume that a crossing-minimization algorithm is applied first, yielding a *planarization*, i.e., a planar graph with a dummy vertex for each crossing, that fixes the topology of the resulting drawing. We present and evaluate two different approaches for drawing a planarization in such a way that the edges of the input graph are as straight as possible. The first approach is based on the planarity-preserving force-directed algorithm ImPrEd [28], the second approach, which we call *Geometric Planarization Drawing*, iteratively moves vertices to their locally optimal positions in the given initial drawing.

Our evaluation shows that both approaches significantly improve the initial drawing and that our geometric approach outperforms the force-directed approach. To the best of our knowledge, this is the first paper concerned with the generation of a straight-line drawing that respects an arbitrary planarization.

Submitted:	Reviewed:	Revised:	Accepted:
January 2019	March 2019	May 2019	July 2019
	Final:	Published:	
	July 2019	September 2019	

Article type:	Communicated by:
Regular paper	G. Liotta

This work was initiated within the FYS *Heuristische Verfahren zur Visualisierung von dynamischen Netzwerken*, financially supported by the "Concept for the Future" of KIT within the framework of the German Excellence Initiative. The work was partially supported by grant RU 1903/3-1 and WA 654/21-1 of the German Research Foundation (DFG). A preliminary version of the paper was published at SOFSEM'17 [2]

E-mail addresses: thomas.blaesius@hpi.de (Thomas Bläsius) radermacher@kit.edu (Marcel Radermacher) rutter@fim.uni-passau.de (Ignaz Rutter)

1 Introduction

In his seminal paper "How to Draw a Graph" [31], Tutte showed that every planar graph admits a planar straight-line drawing. His result has been strengthened in various ways, e.g., by improving the running time, the required area [5] or to restrict the position of some vertices to points on a line [8, 21]. In practice, however, many graphs are non-planar and we are interested in finding straight-line drawings with few crossings. Unfortunately, crossing minimization for straight-line drawings is $\exists\mathbb{R}$-complete, i.e., as hard as the existential theory of the reals [26]. We thus need to relax either the condition of minimizing the number of crossings or the requirement of straight edges. Approximating the rectilinear crossing number seems difficult, and for complete graphs K_n, it is only known for $n \leq 27$ [3]. Radermacher et al. [24] require straight-line edges and heuristically minimize the number of crossings. In this paper, we follow the second approach, i.e., we insist on a small (though not necessarily minimum) number of crossings and optimize the straightness of the edges in the drawing.

In contrast to the geometric setting, the crossing number for topological drawings has received considerable attention and there is a plethora of results on crossing minimization; see [4] for a survey. The output of these algorithms typically is a planarization G_p of the input graph G together with a planar embedding. To profit from the results in this area, we focus on the problem of drawing G_p such that for each edge of G the corresponding *planarization path* in the drawing of G_p is as straight as possible.

This type of problem is prototypical for several fundamental problems in graph drawing that ask for a geometric realization of a given combinatorial description of a drawing. The most prominent examples are the topology-shape-metrics framework for orthogonal graph drawing [29] and the fundamental ($\exists\mathbb{R}$-complete) problem Stretchability, which asks whether a given arrangement of pseudolines can be realized by geometric lines [22]. There have been several other works that consider the problem of realizing a given combinatorial description of a drawing geometrically.

Thomassen [30] gives a characterization for 1-planar graphs that admit a straight-line drawing. Moreover, he shows that there is no finite number of forbidden configurations that characterize the straight-line drawable 2-planar graphs. Di Giacomo et al. show that if the set of edges without crossings of a non-planar graph form a connected subgraph then there is a drawing of the same graph with at most three bends per edge that respects prescribed topological constraints [13]. Otherwise, the number of bends is in $\Omega(\sqrt{n})$), where n is the number of vertices of G. Eades et al. study when a (maximal) planar graph with an additional edge has straight-line drawing [12]. Radermacher and Rutter consider the problem of computing such a realizable embedding of a planar graph with an additional edge with a minimal number of crossings for restricted planar graph classes [25].

Chan et al. [6] prove that a linear number of bends per edges is sufficient to extend a given straight-line drawing of a planar graph. Given a fixed convex drawing of a face f of a planar graph, Mchedlidze et al. [20] introduce a linear-time algorithm to test whether there is a straight-line drawing of a planar graph that extends the drawing of f. Grilli et al. [14] study the problem of realizing a given simultaneous planar embedding of two (or more) graphs with few bends per edge. For a survey on graph drawing beyond planarity see [9].

The algorithm of Dwyer et al. [11] minimizes the stress of a layout while preserving the topology of the drawing. Didimo et al. [10] present an algorithm that is able to preserve the topology unless changing the topology improves the number of crossings. Bertault [1] presents PrEd, a force-directed layout algorithm for planar graphs that preserves the combinatorial embedding of the input drawing; the approach was later improved by Simonetto et al. [28]. To the best of our knowledge the problem of producing a drawing of an arbitrary planarization such that the planarization paths are drawn as straight as possible has not been investigated prior to this work.

Contribution and Outline. We study the problem of finding a drawing of a given planarization G_p of a graph G such that the planarization paths corresponding to the edges of G are drawn as straight as possible. We present two approaches, one is based on an adaption of ImPrEd that includes additional forces to facilitate straightening the planarization paths (Sec. 3). The second is a geometric framework that iteratively moves the vertices of a given drawing one by one to locally optimal positions such that (i) the planarization and its planar embedding are preserved and (ii) the angles on planarization paths influenced by that vertex are optimized (Sec. 4). This framework has several degrees of freedom, such as the vertex processing order and the exact placement strategy for vertices. We experimentally evaluate the modified ImPrEd algorithm (ImPrEd++) and several configurations of the Geometric Planarization Drawing approach in a quantitative study (Sec. 5). We show that all our methods significantly increase the straightness compared to the initial drawing and that the geometric algorithms typically outperform ImPrEd++ in terms of quality. Statistical tests are used to show that these results are significant with 95% confidence.

2 Preliminaries

Intuitively, a planarization of a graph G is the graph resulting from placing dummy vertices at the intersections of edges in a drawing of G. More formally, let $G = (V, E)$ be a graph and let $G_p = (V \dot{\cup} V_p, E' \dot{\cup} E_p)$ be a planar graph such that every edge in E_p is incident to at least one vertex in V_p. The vertices in V_p are called *dummy vertices*. Then G_p is a *planarization* of G if the following conditions hold. (i) Dummy vertices have degree 4, (ii) $E' \subseteq E$, (iii) for every edge $e = uw \in E \setminus E'$, G_p contains a *planarization path* from u to w whose edges are in E_p and whose internal vertices are in V_p, (iv) for any two distinct edges $e, e' \in E \setminus E'$ the paths p_e and $p_{e'}$ are edge-disjoint, and (v) the paths p_e, $e \in E \setminus E'$ cover all edges in E_p. We call the planarization G_p *k-planar* if

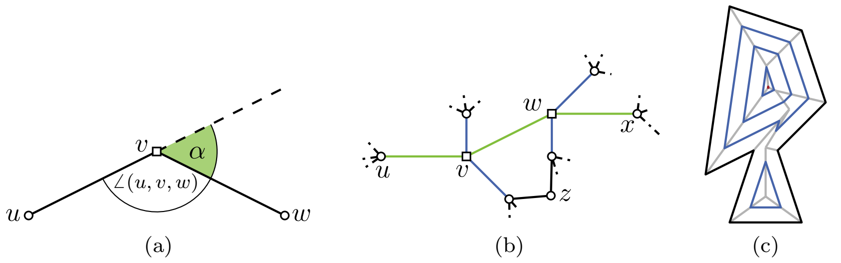


Figure 1: (a) The deviation angle sd-$\alpha(u, v, w) = \alpha$ of the dissected pair (u, v, w). (b) Vertex u and w are tail vertices of the dissected pair (u, v, w). Since w is a dummy vertex of the dissected pair (v, w, x), w is a hybrid vertex. z is an independent vertex. (c) A (grey) straight skeleton of a (black) polygon and a set of (blue) shrinked polygons. The geometric center is depicted in red.

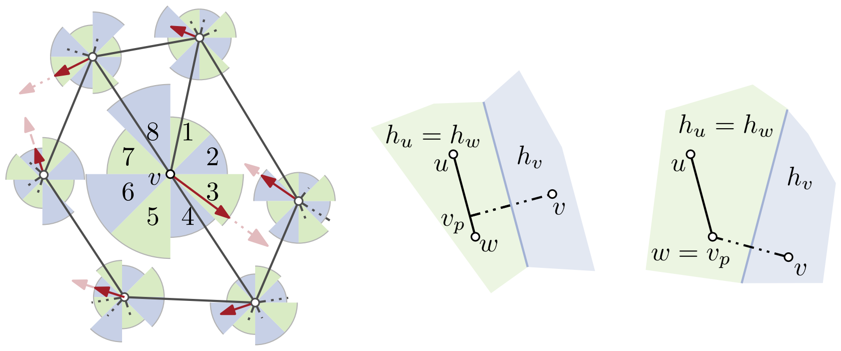
the longest planarization path has k dummy vertices, i.e., there are at most k crossings per edge.

A *dissected pair* (u, v, w) is a pair $uv, vw \in E_p$ of edges that belong to the same planarization path; see Fig. 1a. Note that formally (u, v, w) and (w, v, u) do not coincide but we for the purpose of this paper we consider the two dissected pairs to be the same. The *straight-line-deviation angle* sd-$\alpha(u, v, w)$ of (u, v, w) is the angle sd-$\alpha(u, v, w) = \pi - \angle(u, v, w)$. We simply refer to a straight-line-deviation angle as *deviation angle.* A deviation angle is *active* with respect to v (also called *v-active*) if moving v can alter that angle. This notation allows us to formalize our problem of drawing the planarization paths of G_p as straight as possible as follows: Given an embedded planarization G_p of G and an angle α, is there a planar straight-line drawing of G_p with the given embedding such that all deviations angles are smaller than α, i.e., sd-$\alpha(u, v, w) \leq \alpha$ for every dissected pair (u, v, w) of G_p? The respective optimization problem asks for the minimum angle α.

For a dissected pair (u, v, w), v is a dummy vertex and u and w are *tail* vertices; see Fig. 1b. A dummy that is not a tail is called *pure dummy* and a tail that is not a dummy is called *pure tail.* Vertices that are both, tail and dummy, are called *hybrid.* A vertex that is neither a dummy nor a tail vertex is called *independent.*

Let P be a polygon and let v be a vertex of P. A point p in the interior of P is *visible* from v if the straight line connecting p with v does not intersect an edge of P. The *visibility region of v in P* is the set of all points in P that are visible from v. The *size* of a polygon P is the number of its vertices.

A *shrinked* polygon P' of a polygon P is the result of moving the vertices towards the interior of a polygon P with constant speed along the *straight skeleton* of P [16]; see Fig. 1c. A *geometric center* of a polygon P is obtained by shrinking P to a single point. In case that the shrinking process yields disconnected polygons, we consider the center of the polygon with the largest



(a) (b) (c)

Figure 2: (a) Radial zones (blue and green) used by ImPrEd. Forces (light red) are cropped at the boundary of the zones (dark red) (b,c) Construction of the half-planes L_x in case (b) that the projection of v lies on uv or (c) the projection does not lie on uv.

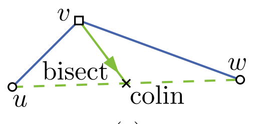
area as the center of P.

3 Force-Directed Planarization Drawing

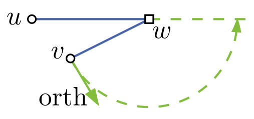
We present a force-directed approach ImPrEd++ for straightening the planarization paths in a given drawing based on ImPrEd [28], a spring embedder that is able to preserve the planar embedding of a given drawing. ImPrEd preserves the combinatorial embedding of a planar straight-line drawing as follows. Let $Z_1, \dots, Z_8$ be a partition of the unit disk around a vertex v into eight octants, refer to Fig. 2a. The radius of each octant Z_i is scaled by a value R_i such that any movement of v by a direction lying inside Z_i preserves the combinatorial embedding. In order to allow a more flexible movement of each vertex, we substitute the radial zones with a convex polygon P_v. The polygon corresponds to the construction given in the correctness proof of ImPrEd [28].

For each vertex v, let L_v be a set of half-planes constructed as follows; see Fig. 2b and Fig. 2c. For each edge uw of G and let v_p^{uw} be the projection onto the line through uw. If the projection v_p^{uw} does not lie on the segment uw, set v_p^{uw} to the closest point on uw. Let l_v be the line perpendicular to the segment vv_p^{uw} through the middle point of the segment vv_p^{uw}. For each vertex $x \in \{u, v, w\}$ we add the half-plane h_x of l_v that contains x to the set L_x. Finally, the polygon P_v is the intersection of all half-planes in the set L_v.

To reduce the deviation angles, we introduce the new forces F^d for dummy vertices and F^t for tail vertices. Hybrid vertices are affected by both forces. For independent vertices, we apply the same forces as ImPrEd.



(a)



(b)

Figure 3: Our new forces. If v is a dummy vertex (a), move it along the bisector of the adjacent segments. If v is a tail vertex (b), move it gradually along an arc.

Let v be a dummy vertex and let (u, v, w) be a dissected pair containing v. To encourage placing v collinearly between u and w, we apply a force in the direction of the unit length bisector $\mathrm{bisect}(u, v, w)$ of the vectors $u-v$ and $w-v$; see Fig. 3a. Let $\mathrm{colin}(u, v, w)$ denote the point on the bisector that is collinear with u and w. We use the dummy force $F^d(v, (u, w)) = \lambda(\mathrm{colin}(u, v, w) - v)$, where $0 < \lambda < 1$ is a damping factor. To form the dummy force $F^d(v)$ for v, we sum over the two dissected pairs where v is the dummy vertex.

For a tail vertex v and a dissected pair (u, w, v), we want to place v on the extension of the segment uw; see Fig. 3b. To accomplish this, we try to perform a radial movement of v around w over several iterations of the spring embedder. Hence, we introduce a force in the normalized direction $\mathrm{orth}(u, w, v)$ of the tangent at v with the circle centered at w and passing through v. The direction of $\mathrm{orth}(u, w, v)$ is chosen such that it points away from the segment uw. The strength of the force is proportional to $\mathrm{dist}(v, w)$ with a damping factor of $0 < \kappa < 1$, i.e., $F^t(v, (u, w)) = \kappa\, \mathrm{dist}(w, v)\, \mathrm{orth}(u, v, w)$. To obtain the resulting force for a tail vertex v, we sum over all dissected pairs where v is a tail vertex.

4 Geometric Planarization Drawing

The spring embedder described in Sec. 3 restricts the movement of each vertex in a very conservative manner, i.e., the restrictions ensure a preservation of the given planar embedding. This may waste a lot of potential; see Fig. 4a and Fig. 4b. The approach presented in this section aims to tap the full potential by making each movement locally optimal. As the simultaneous movement of multiple vertices leads to non-trivial and non-local dependencies, we move only a single vertex in each step.

To make this precise, we need to answer two questions. First, to which points can a vertex v be moved such that the planar embedding is preserved? Second, which of these points is the best position for v? Concerning the first question, we call the set of points satisfying this property the *planarity region* of v and denote it by $\mathcal{PR}(v)$. We show in Sec. 4.1 how to compute $\mathcal{PR}(v)$ efficiently. Concerning the second question, we define the *cost* of a point $p \in \mathcal{PR}(v)$ to be the maximum of all v-active deviation angles when placing v at p. A point in $\mathcal{PR}(v)$ is a *locally optimal* position for v if $\mathcal{PR}(v)$ contains no other point with

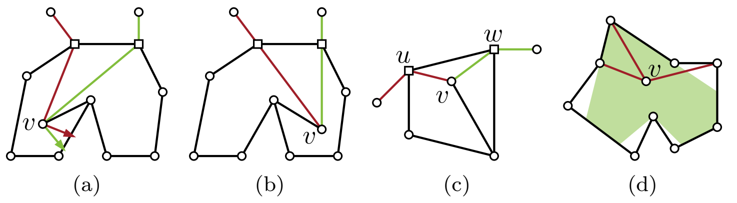


Figure 4: An initial drawing (a) that is difficult to repair using the force-directed algorithm although v could be moved to an optimal position without violating planarity (b). (c) The closer v lies to the edge uw, the better are the v-active angles. (d) The (green) planarity region of v.

strictly smaller cost. In Sec. 4.2, we show how to compute an arbitrarily exact approximation of the locally optimal position.

The overall algorithm can be described as follows. We iterate over all vertices of the graph. In each step, the current vertex is moved to its locally optimal position. We repeat until we reach a drawing that is stable or a given number of iterations is exceeded.

One important degree of freedom in this algorithm is the order in which we iterate over the vertices. Another choice we have not fixed so far is the placement of independent vertices. As an independent vertex has no active angle, each point in its planarity region is equally good. We propose and evaluate different ways of filling these degrees of freedom in Sec. 5.

For a tail or dummy vertex v, it can happen that there exists no locally optimal position due to the fact that $\mathcal{PR}(v)$ is an open set. The cost may for example go down, the closer we place v to an edge connecting two other vertices; see Fig. 4c. We therefore shrink $\mathcal{PR}(v)$ slightly and consider it to be a closed set. On one hand, this ensures that a locally optimal position always exists. On the other hand, it (partially) prevents that vertices are placed too close to edges, which is usually not desirable in a drawing. The offset by which we shrink $\mathcal{PR}(v)$ is discussed in Sec. 5, where we describe our exact evaluation setup.

4.1 Planarity Region

In case that a dummy vertex v is a cut-vertex of G_p, we can reduce the number of crossings in G; see Fig. 9.

Let G_p be a planarization with a given drawing and let v be a vertex of G_p. Let f_v be the face of $G_p - v$ that contains the current position of v. Assume for now that f_v is bounded by a polygon surr(v), which we call the *surrounding* of v. Consider a point p in the interior of f_v and assume that we use p as the new position for v. Clearly, the resulting drawing is planar if and only if p is visible from each of v's neighbors; see Fig. 4d.

Thus, the planarity region $\mathcal{PR}(v)$ is the intersection of all visibility regions in surr(v) with respect to the neighbors of v. It follows that the planarity region can be obtained by first computing the visibility polygons of v's neighbors in $\mathrm{surr}(v)$, and then intersecting these visibility polygons. Let n_v be the number of vertices of the surrounding polygon $\mathrm{surr}(v)$ and let d_v be the degree of v. Observe that if v is not a cut-vertex then $\mathrm{surr}(v)$ does not have holes and computing the d_v visibility polygons takes $O(d_v n_v)$ time [18]. To intersect these d_v visibility polygons (each having size $O(n_v)$), one can use a sweep-line algorithm [23] consuming $O((k + d_v n_v) \log n_v)$ time, where k is the number of intersections between segments of the visibility polygons. As there are at most $d_v n_v$ segments, $k \in O(d_v^2 n_v^2)$ holds, yielding the running time $O(d_v^2 n_v^2 \log n_v)$ for computing the planarity region. We first show that we can improve this running time in case that v is not a cut-vertex. Subsequently, we show how to modify $\mathrm{surr}(v)$ so that we are able to apply the following lemma.

Lemma 1 *If v is not a cut-vertex, then the planarity region $\mathcal{PR}(v)$ of v has size $O(n_v)$ and can be computed in $O(d_v n_v \log n_v)$ time.*

Proof: Let P_u be the visibility polygon of u in $\mathrm{surr}(v)$. A segment w on the boundary of P_u that is not part of a segment of $\mathrm{surr}(v)$ is called *window*. We say that the window w is *generated* by u; compare Fig. 5. Instead of intersecting the visibility polygons of all neighbors, we compute the planar subdivision induced by the segments of $\mathrm{surr}(v)$ and all windows generated by neighbors of v. As there are only $O(d_v n_v)$ windows, this can be done (again using a simple sweep-line algorithm) in $O((k + d_v n_v) \log n_v)$ time, where k is the number of intersections between segments, i.e., the number of vertices of the resulting planar subdivision H. We show the following three claims.

Claim 1 *The planarity region of v is a face of the subdivision H.*

Claim 2 *Every window intersects with $O(d_v)$ segments.*

Claim 3 *It suffices to consider $O(n_v)$ windows.*

The first claim implies that we can compute the planarity region in linear time in the size of H as we only need to find the face of H containing the previous position of v (which is clearly contained in the planarity region). Each vertex of H is either a vertex of $\mathrm{surr}(v)$ or an intersection of a window with a segment (which is either also a window or a segment of $\mathrm{surr}(v)$). Thus, the second and third claim show that $k \in O(d_v n_v)$ holds. It is moreover not hard to see that no two different edges on the boundary of a face of H belong to the same segment of $\mathrm{surr}(v)$ or to the same window. Thus, each face (and in particular the planarity region) is bounded by only $O(n_v)$ edges, which concludes the proof.

To prove Claim 1, first note that $\mathrm{surr}(v)$ is the outer face of H, as every window lies completely inside $\mathrm{surr}(v)$. Let f be the face of H containing the previous position of v. We step by step remove subgraphs of H that eliminate only faces that cannot be part of the planarity region $\mathcal{PR}(v)$. In the end, only the face f remains, which shows $\mathcal{PR}(v) = f$. For this purpose consider an edge e incident to f. If e is not on the outer face of H, then e is part of a window w.

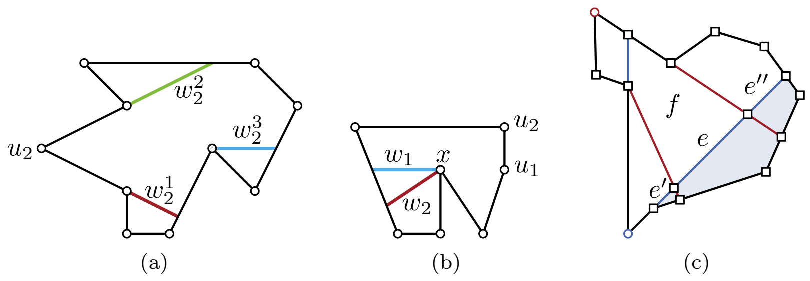


Figure 5: (a) Three windows generated by the neighbor u_2. (b) The window w_2 (generated by u_2) is dominated by w_1 (generated by u_1). (c) The edges e' and e'' extend e to a path π that correspond to a window of the neighbor u_1 (blue). Removing the blue region that does not contain f, reduces the size of H (squared vertices).

We can extend e to a path π between vertices on the outer face such that the edges on π are all part of w. Then π separates H into two parts. Faces in the part not containing f clearly cannot be part of the planarity region due to the window w. Thus, we can remove this part, which has the effect that e now lies on the outer face. Once all edges incident to f lie on the outer face, the claim follows.

For Claim 2, observe that every window has two intersections with segments of surr(v). Thus, all remaining intersections are with other windows. Let w_1 be a window generated by the neighbor u_1 of v and let u_2 be another neighbor of v. We show that w_1 intersects at most two windows generated by u_2, which directly implies the claim. To this end, consider three windows w_2^1, w_2^2, and w_2^3 generated by u_2; see Fig. 5a. Since the lines through w_i^2 intersect in u_2, the planar subdivision of surr(v) with these three windows has four inner faces; one face incident to all three windows (and to edges of surr(v)), and one face for each window w_2^i (for $i \in \{1, 2, 3\}$) that is only incident to w_2^i and edges of surr(v). A window w_1 intersecting all three windows w_2^1, w_2^2, and w_2^3 would need to cross the boundary of each of the latter three faces exactly once, which is clearly impossible. Thus, w_1 can intersect at most two windows generated by u_2.

To show Claim 3, note that at least one endpoint of every window is a concave corner in surr(v), i.e., a vertex of surr(v) with an interior angle that is grater than 180°. Consider one concave corner x and let w_1 and w_2 be two windows with endpoint x. The window w_1 separates surr(v) into two parts, one of which cannot be part of the planarity region. If w_2 lies in this part, then w_2 yields no real restriction compared to w_1; see Fig. 5b. Thus, we say that w_2 is *dominated* by w_1. Clearly, removing all dominated windows does not alter the result of the algorithm. Moreover, it is not hard to see that there can be at most two non-dominated windows sharing an endpoint. Thus, Claim 3 follows, which concludes the proof. □

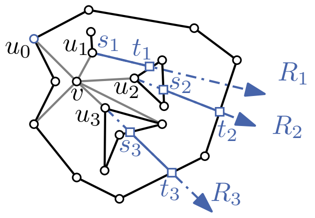


Figure 6: The blue segments are added as edges to G to ensure that v is not a cut-vertex.

Theorem 1 *The planarity region can be computed in $O(d_v^2 n_v \log n_v)$ time.*

Proof: If v is not a cut-vertex, we can apply Lemma 1. Hence, consider the case that v is a cut-vertex. Then the surrounding polygon surr(v) has holes. In the following, we show how to locally modify G such that v is not a cut-vertex anymore and such that the planarity region of v in the new graph coincides with the planarity region of v in G. Let $P_0, P_1, \dots, P_k$ be the poygons that describe the boundary of surr(v), i.e., P_0 is the outer polygon and $P_1, \dots, P_k$ the holes in the interior of P_0; see Fig. 6. Moreover, let u_i be a neighbor of v that lies on the boundary of P_i. Consider the ray R_i starting in u_i in the direction from u_0 towards u_i. Let $s_i t_i \in R_i$ be the segment of minimal length such that s_i lies on P_i and t_i on $P_j, j \neq i$. We subdivide the corresponding edges in G and add $s_i t_i$ as an edge to G.

Clearly the planarity region of v in the modified graph and the original graph coincide and v is not a cut-vertex anymore. To finish the proof of the theorem we have to prove the claimed running time. First, the polygonal chains $P_0, \dots, P_k$ and the neighbors u_i can be computed in $O(\sum_{i=0}^{k} |P_i|) = O(n_v)$ time. Each segment $s_i t_i$ can be computed in $O(\sum_{i=0}^{k} |P_i|) = O(n_v)$ time. Overall this yields a running time of $O(d_v n_v)$. Observe that the size if the surr(v) in the new graph is in $O(d_v n_v)$. Thus, we can compute the planarity region of v by Lemma 1 in $O(d_v^2 n_v \log n_v)$ time. □

4.2 Finding a Locally Optimal Position

In this section, we are given a vertex v together with its planarity region $\mathcal{PR}(v)$ and we want to compute a locally optimal position. We consider the two cases where v is a pure tail-vertex and the one where v is a pure dummy-vertex. These two cases can be combined to also handle hybrid vertices. For both cases, our approach is the following. For a given angle α, we show how to test whether $\mathcal{PR}(v)$ contains a point with cost less or equal to α. For any $\varepsilon > 0$ we can then apply $O(\log(1/\varepsilon))$ steps of a binary search over the domain $\alpha \in [0, 2\pi)$ to find a position in $\mathcal{PR}(v)$ whose cost is at most ε larger than the cost of a locally optimal position.

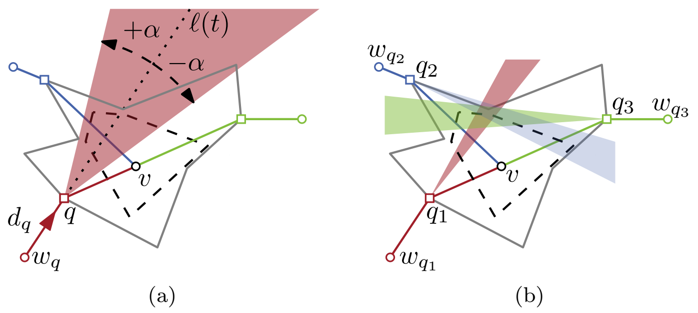


Figure 7: (a) A cone with respect to one neighbor q of v. (b) The intersection of all cones with the planarity region (dashed) includes possible positions for the vertex v.

4.2.1 Placing a Pure Tail Vertex.

Let v be a pure tail vertex and let $D(v) \subseteq N(v)$ be the set of dummy neighbors of v, where $N(v)$ is the neighborhood of v; see Fig. 7. For each dummy neighbor $q \in D(v)$ there is a dissected pair (w_q, q, v) whose angle is active. Note that these are the only active angles of a pure tail vertex. Consider the (oriented) line $\ell(t) = q + t \cdot d_q$ with the direction vector $d_q = q - w_q$. Clearly, placing v onto $\ell(t)$ (for $t > 0$) results in the deviation angle sd-$\alpha(w_q, q, v) = 0$. Moreover, all points in the plane that yield sd-$\alpha(w_q, q, v) \leq \alpha$ lie in a *cone*, i.e., in the intersection (union if $\alpha \geq \pi/2$) of two appropriately chosen half-planes.

It follows, that v can be moved to a position with cost α if and only if the intersection of all cones has a non-empty intersection with the planarity region $\mathcal{PR}(v)$; see for example Fig. 7. As v has at most d_v dummy neighbors (recall that d_v is the degree of v), the intersections of all cones can be computed in $O(d_v^2 \log d_v)$ time using a sweep-line algorithm [23]. Let $\mathcal{C}$ be the resulting intersection of the cones. Testing whether $\mathcal{C}$ and $\mathcal{PR}(v)$ have non-empty intersection can be done in $O((p_v + d_v^2) \log p_v)$ time, where p_v is the size of $\mathcal{PR}(v)$.

Lemma 2 *Let v be a pure tail vertex and assume $\mathcal{PR}(v)$ has already been computed. For any $\epsilon > 0$, an absolute ϵ-approximation of the locally optimal position can be computed in time $O(\log(1/\epsilon)(p_v + d_v^2) \log p_v)$.*

4.2.2 Placing a Pure Dummy Vertex.

A pure dummy vertex v has only two active deviation angles. Let $N(v) = \{a, p, b, q\}$ be the neighbors of v so that (a, v, b) and (p, v, q) are dissected pairs. Consider the angle $\beta = \angle avb$. By a generalization of *Thales' Theorem*, β does not change when moving v on a circular arc with endpoints a and b. Thus, to make sure that β is at least $\pi - \alpha$ (i.e., to ensure that sd-$\alpha(a, v, b) \leq \alpha$), one has to place v in the intersection of two discs (union if $\alpha > \pi/2$); see Fig. 8. These two disks must have a and b on their boundaries and basic geometry shows that their radii have to be $|ab|/(2\sin(\pi - \alpha))$ (which uniquely defines the two disks).

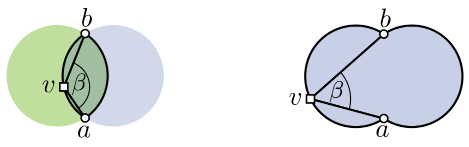


Figure 8: The angle $\angle avb$ is at least β for $\beta > 90°$ ($\beta < 90°$) if and only if v lies in the intersection (union) of two discs (including its boundary, but excluding a and b).

The same applies for $\angle pvq$. Thus, requiring both active deviation angles sd-$\alpha(a, v, b)$ and sd-$\alpha(p, v, q)$ to be at most α restricts the possible positions of the dummy vertex v either to the intersection of four disks, or to the intersection of the union of two disks with the union of two other disks. The check whether this intersection is empty requires time linear in the size p_v of the planarity region.

Lemma 3 *Let v be a pure dummy vertex and assume $\mathcal{PR}(v)$ has already been computed. For any $\epsilon > 0$, an absolute ϵ-approximation of the locally optimal position can be computed in time $O(\log(1/\epsilon)p_v)$.*

4.2.3 Placing a Hybrid Vertex.

Let v be a dummy vertex with at least one dummy neighbor. Combining the techniques from the two previous sections, we have to check whether $\mathcal{PR}(v)$ has a non-empty intersection with the intersection of up to four cones and up to four disks. This can again be done in time linear in the size p_v of the planarity region. We can thus conclude (for all three types of vertices) with the following theorem.

Theorem 2 *Let v be a vertex and assume $\mathcal{PR}(v)$ has already been computed. For any $\epsilon > 0$, an absolute ϵ-approximation of the locally optimal position can be computed in time $O(\log(1/\epsilon)(p_v + d_v^2) \log p_v)$.*

Overall Running Time. We have seen that the planarity region for a vertex v can be computed in $O(d_v^2 n_v \log n_v)$ time (Theorem 1) and that a locally optimal position can be approximated in $O(\log(1/\varepsilon)(n_v + d_v^2) \log p_v)$ time. Note that if v is not a cut-vertex $p_v \in O(n_v)$ otherwise it is in $O(d_v n_v)$. In the following, we assume that ε is a small constant and omit it from the running time.

As the degree d_v of a vertex v is a lower bound for the size n_v of its surrounding, the running time of computing the planarity region dominates the time for computing the locally optimal position. Each iteration thus needs $O(\sum_{v \in V} d_v^2 n_v \log n_v)$ time. In the worst case, this yields the running time stated in the following theorem.

Theorem 3 *One iteration of the Geometric Planarization Drawing approach takes $O(n^4 \log n)$ time.*

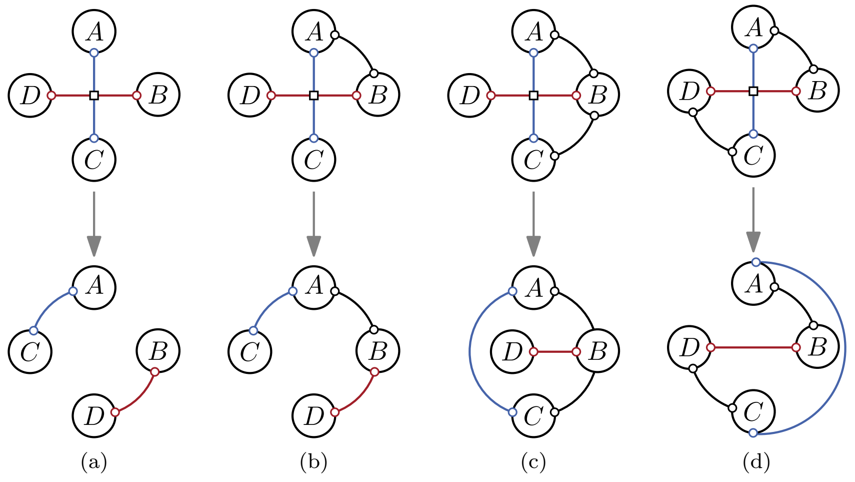


Figure 9: Removing a crossing in case that G_p is not biconnected and a dummy vertex is a cut-vertex.

Observe that since we assume that G has a small number of crossings, a cut-vertex v can not be a dummy vertex; compare Fig. 9. Thus if consider only biconnected graphs the running time reduces to $O(n^3 \log n)$. The running time improves further to $O(n^2 \log n)$ if the face degrees are bounded by a constant and even to $O(n)$ if additionally the vertex degrees d_v are bounded.

Corollary 1 *If G is biconnected, one iteration of the Geometric Planarization Drawing approach takes $O(n^3 \log n)$ time.*

5 Evaluation

We present an empirical evaluation of our planarization drawing methods. We first discuss the remaining degrees of freedom in our Geometric Planarization Drawing framework. Afterwards, we describe our experimental setup and the statistical tests we use for the evaluation. The first part of our evaluation focuses on the quality of different configurations of our Geometric Planarization Drawing approach. The second set of experiments focuses on the running time. We evaluate three benchmark sets. We give an extensive evaluation of the *rome graphs*. Based on the insights obtained from these graphs, we report the results for the *north* and *community graphs* for a limited number of configurations. We conclude the section with a presentation of a few sample drawings.

5.1 Degrees of Freedom in the Geometric Framework

As pointed out above, our algorithmic framework offers quite a number of degrees of freedom and possibilities for tweaking the outcome of the algorithm.

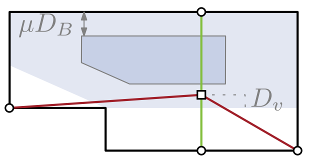


Figure 10: Moving the square dummy vertex towards the boundary of the planarity region decreased the deviation angle of the red dissected pair.

Initial Drawing We consider two sets of initial drawings ImPrEd and Gc, are both obtained from a planar straight-line drawing computed by PlanarStraightLayout [19] computed with OGDF [7]. For the first set of initial drawings we applied 100 iterations of ImPrEd, without the forces to optimize the planarization, to the drawings obtained by the PlanarStraightLayout algorithm. For the second set, we iteratively select a vertex and move the vertex to the *geometric center* of its planarity region, i.e., the planarity region is shrunken to a single point. As before, we repeated this process 100 times.

Vertex Orders We propose different orders for processing the vertices. An Outer Shell is obtained by iteratively removing the vertices of the outer face. An Inner Shell order is the reverse of an Outer Shell, and an Alternating Shell order is obtained by alternating between the two orders. Path Repair is a sequence of vertices where every vertex occurs d_v times. Each edge of the graph G, corresponds to a sequence of vertices of the planarization G_p, namely the vertices on the corresponding planarization path (or an edge) ordered according to their appearance on that path (or the sequence of the two end-vertices if the edge has no crossings). To obtain the Path Repair order, we concatenate these sequences in an order based on a breadth-first search.

Placement of Independent Vertices For an independent vertex v, every position in the planarity region $\mathcal{PR}(v)$ is equally good since all deviation angles are inactive. To reduce the restrictions imposed by independent vertices on their neighbors, we place v in the geometric center of $\mathcal{PR}(v)$.

Shrinking the Planarity Region As mentioned before, a locally optimal position for a vertex v might not exists as $\mathcal{PR}(v)$ is an open set; see Fig. 10. Moreover, it is visually unpleasant when vertices are placed too close to non-incident edges. We thus shrink $\mathcal{PR}(v)$ as follows. Let D_B be the length of the smallest side of the planarity region's bounding box and let $\mu > 0$ be a parameter. Let D_v be the smallest distance from v to a point on the boundary of $\mathcal{PR}(v)$. On one hand, the polygon obtained from shrinking $\mathcal{PR}(v)$ by μD_B may not contain v and therefore can yield a worse deviation angle. On the other hand, if v lies close to the geometric center of $\mathcal{PR}(v)$, shrinking $\mathcal{PR}(v)$ by D_v restricts the movement of v to a small region around v. Hence, we choose to

Table 1: Configurations for our Geometric Graph Drawing approach.

Configuration	**Vertex Order**	**Angle Relax. Weight**
Alternating Shell	Alternating-Shell	0.0
Shell	Outer-Shell	0.0
Path-Repair	Path-Repair	0.0
Relax-x	Alternating-Shell	$x \cdot 10^{-1}$
		$x \in \{1, 2, 4, 6, 8\}$

shrink $\mathcal{PR}(v)$ by the minimum of the values μD_B and D_v. In our experiments we used $\mu = 0.1$.

Angle Relaxation While the placement of the tail and hybrid vertices introduced in Sec. 4.2 works independently from the vertex order, it is natural to require that *unplaced* vertices (i.e., vertices that will be moved later in the same iteration) should have a smaller influence on positioning decisions. Hence, we alter the binary search in the cone construction: we replace the opening angle α of the cones of unplaced vertices by $(1 - \gamma)\alpha + \gamma\pi$, where $\gamma \in [0, 1]$ is the *angle relaxation weight*, thus widening their cone depending on the value of γ.

Drawing Region The drawing region is always limited by an axis-aligned rectangle whose side-length is twice as large as the corresponding side-length of the smallest axis-aligned rectangle that entirely contains the initial drawing.

Termination We consider two possibilities to terminate the execution of our algorithm, (i) after a fixed number of iterations, and (ii) after a fixed period of time. In order to allow a fair comparison between all algorithms in Sec. 5.3, each algorithm gets exactly $5n$ seconds to optimize the drawings. For experiments regarding the running time in Sec. 5.4, we measure the time until convergence limited by 100 iterations.

Configurations The presented degrees of freedom allow for many different configurations of our algorithm. Table 1 lists a set of configurations of our heuristic that we consider in our evaluation. Moreover, we compare these configurations with the baseline algorithm Initial, which simply outputs the initial drawing, and with our modification ImPrEd++ of the force-directed algorithm ImPrEd. The node-node repulsion force and the edge-attraction force used in ImPrEd are parametrized by a value δ. The node-edge repulsion force has a parameter γ. We set both values to $(\log n)^{-1}\sqrt{A/n}$, where A is area of the drawing region and n the number of vertices of the graph. We set the damping factor λ to of the dummy force to 0.1 and the damping factor κ of the force for tail vertices to 0.05.

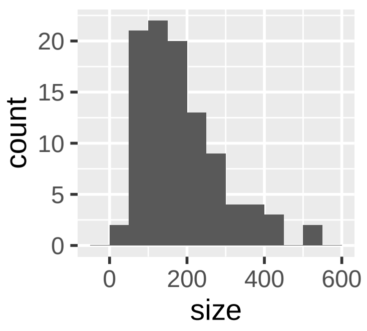


Figure 11: The distribution of the *size* of the selected Rome graphs, i.e., the sum of the number of vertices the sum of the number of vertices and edges of a graph.

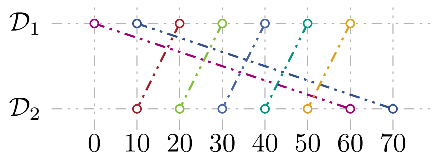


Figure 12: Two sets of deviations angles $\mathcal{D}_1$ and $\mathcal{D}_2$. The number on the bottom denotes the deviation angle of one point, i.e., the red point in $\mathcal{D}_2$ corresponds to a deviation angle of 2. The edges indicate pairings of values in $\mathcal{D}_1$ and $\mathcal{D}_2$, i.e, each dissected pair has a deviation angle in $\mathcal{D}_1$ and in $\mathcal{D}_2$. For example the red edge corresponds to a dissected pair with a deviation angle of 20 in $\mathcal{D}_1$ and a deviation angle of 10 in $\mathcal{D}_2$.

5.2 Experimental Setup and Methodology

We ran the algorithms on 100 randomly selected non-planar Rome graphs[1]. For each of them, we used the largest non-planar biconnected component, Fig. 11 shows the size distribution of these graphs. To take the lengths of the planarization paths into account, we a priori define three classes of instances: LOW ($\mathcal{L}$), MEDIUM ($\mathcal{M}$) and HIGH ($\mathcal{H}$). The partitioning is chosen such that the difference between the minimum and maximum length of the planarization is (roughly) equal for all classes. A planarization belongs to $\mathcal{L}$ and to $\mathcal{H}$ if it is at most 4- and at least 9-planar, respectively. Instances in the class $\mathcal{M}$ are k-planar with $4 < k < 9$. There are 68 graphs with in total 604 dummy vertices in $\mathcal{L}$, 26 graphs with in total 959 dummy vertices in $\mathcal{M}$, and 6 graphs with in total 443 dummy vertices in $\mathcal{H}$. Thus, there are in total 4012 dissected-pairs $\mathcal{D}$ (twice the number of dummy vertices).

In general, we are interested in whether the deviation angles sd-α_1 computed by one algorithm A_1 are smaller than the deviation angles sd-α_2 computed by

[1] `graphdrawing.org/data.html`

another algorithm A_2, e.g., we ask whether the configuration Shell computes smaller deviation angles than ImPrEd++. Denote by $\mathcal{D}_i = \{\text{sd-}\alpha_i(u, v, w) \mid (u, v, w) \in \mathcal{D}\}$ the set of deviation angles of the dissected pairs $\mathcal{D}$ in the drawings computed by A_i. One possibility to compare $\mathcal{D}_1$ to $\mathcal{D}_2$ is to compare the mean, median or the quantiles of $\mathcal{D}_1$ and D_2 to each other. Since the underlying distribution of the angles is unknown and not likely to be, e.g., normal, this comparison can be misleading. Consider the two sets $\mathcal{D}_1$ and $\mathcal{D}_2$ depicted in Fig. 12. The median and mean value of the set $\mathcal{D}_1$ is each 30. The median and mean of $\mathcal{D}_2$ is 40. Thus, this two statistics (as well as the min, max and quantiles) indicate that the set $\mathcal{D}_1$ contains smaller deviation angles than $\mathcal{D}_2$. On the other hand, each dissected pair corresponds to a deviation angle in $\mathcal{D}_1$ and in $\mathcal{D}_2$, indicated by the edges in Fig. 12. Considering the dependence between $\mathcal{D}_1$ and $\mathcal{D}_2$, we see that there are five pairs that have a smaller deviation angle in $\mathcal{D}_2$ than in $\mathcal{D}_1$ and that there are only two pairs that have a smaller angle in $\mathcal{D}_1$ than in $\mathcal{D}_2$. Thus, the conclusion drawn from the statistics can depend on whether the dependence between $\mathcal{D}_1$ and $\mathcal{D}_2$ is considered.

This motivates the following approach to compare the deviation angles of $\mathcal{D}_1$ to $\mathcal{D}_2$. Let $\Delta > 0$ and $\mathcal{D}_1 + \Delta = \{\text{sd-}\alpha_1(u, v, w) + \Delta \mid (u, v, w) \in \mathcal{D}\}$. We say $\mathcal{D}_1$ has an *advantage* of Δ over $\mathcal{D}_2$ if the binomial sign test for two dependent samples [27] with probability 0.5 indicates that there are a significant number of values in $\mathcal{D}_1 + \Delta$ that are smaller than the corresponding values in $\mathcal{D}_2$, i.e., $\text{sd-}\alpha_1(u, v, w) + \Delta < \text{sd-}\alpha_2(u, v, w)$, at a significance level of $\alpha = 0.05$. Note that this binomial sign test is independent of the underlying distribution. Further, we are interested in the smallest angle $\delta \geq 0$ such that the angles in our drawings of a graph G_p are smaller than δ. We define a hypothetical drawing called *δ-drawing* where each deviation angle is δ. For each algorithm, we seek the smallest angle δ such that the resulting drawing has an advantage over the δ-drawing.

In order to further increase the reliability of our results we partitioned the set of samples $\mathcal{D}$ into a training set $\mathcal{T}$, containing 20% of the samples, and a verification set $\mathcal{V}$ containing the remaining 80%. We compute the maximum Δ such that $\mathcal{T}_1$ has an advantage of Δ over $\mathcal{T}_2$. We use the value $3/4 \cdot \Delta$ as the conjectured advantage of $\mathcal{V}_1$ over $\mathcal{V}_2$, i.e., we use the binomial sign test to evaluate whether $\mathcal{V}_1$ has an advantage of $3/4\Delta$ over $\mathcal{V}_2$. In case of the comparison of $\mathcal{D}_1$ to a set of δ-drawings, we compute the smallest δ such that $\mathcal{T}_1$ has an advantage ($\Delta = 0$). We compare $\mathcal{V}_1$ to δ'-drawings, with $\delta' = (4/3 \cdot \delta)$.

Implementation Details We use OGDF[2] to planarize the graphs [15] and to compute the initial drawing [19]. We use the libraries CGAL[3] to compute line arrangements, STALGO [16,17] to shrink polygons, and GMP[4] to represent coordinates.

[2] `ogdf.net` [3] `cgal.org` [4] `gmplib.org`

5.3 Quality of the Drawings

In this Section we discuss the quality of our drawings. The evaluation is guided by the following hypotheses.

I) Gc as an initial drawings yields smaller deviation angles compared to ImPrEd (since the deviation angles of the initial drawings are smaller).

II) The Geometric Planarization Drawing approach and ImPrEd++ each have an advantage over the initial drawing.

III) Geometric Planarization Drawing has an advantage over ImPrEd++.

IV) Relax-1 has an advantage over Relax-2, Relax-4, Relax-6 and Relax-8, respectively.

V) In class $\mathcal{H}$, Relax-1 has an advantage over Alternating-Shell (due to the weakened influence of unplaced vertices).

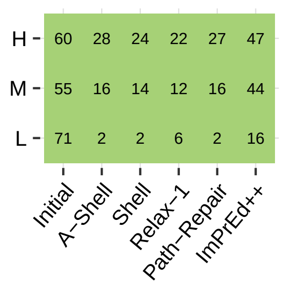
VI) In the presence of long planarization paths, the Path Repair order has an advantage over other vertex orderings (due to its ability to process all vertices of a planarization path consecutively).

We use Fig. 13 and Fig. 14 to show whether or not the binomial test supports our hypotheses. The figures are supplemented with the statistics in Tab. 2. A value Δ in a cell in Fig. 14 is the conjectured advantage of the algorithm over the algorithm on the y-axis computed on the training set. Note that the respective maximum advantage on the training set is $4/3 \cdot \Delta$. A green cell means that we can accept the hypothesis with a confidence of 95%, i.e., also has an advantage of Δ on our verification set. On the contrary, with a red cell we have to reject the hypothesis. An empty cell, indicates that the algorithm did not have an advantage on the training set. We rounded the values Δ to the largest integer Δ' such that $\Delta' < \Delta$. Therefore, a green cell that contains a 0 means that the algorithm on the x-axis has advantage of $\Delta < 1$ over the algorithm on the y-axis.

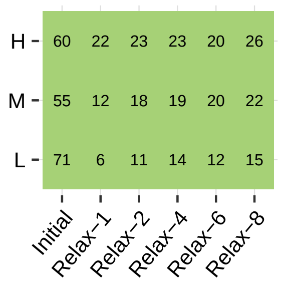
For example, in the class $\mathcal{H}$ (see Fig. 14d), we conjecture, based on the observation in the training set, that the drawings of the Shell configuration have an advantage of 7° over the drawings of ImPrEd++. Recall, that having an advantage means that 50% of the deviation angles, plus an additional buffer of 7°, of the first drawings are smaller than the deviation angles of the second. Since the cell is green, the binomial test on the verification set says that we can accept the hypothesis with a confidence of 95%.

By Fig. 13a, for class $\mathcal{L}$ we can say with 95% confidence that there are significant number of the deviation angles computed by the Shell configuration that are smaller than 2°. This is not necessarily true for $\delta = 1°$. We now discuss our hypotheses.

Hypothesis I) Good initial drawing For each configuration, we compared the deviation angles of the final layouts computed by the configuration applied to both sets of initial drawings (Gc and ImPrEd). For each configuration, our test indicated that Gc does not have an advantage over the ImPrEd drawings. In the contrary, the test indicates that ImPrEd has, for each configuration, an advantage 0.2° over Gc. The binomial test confirmed these advantages at

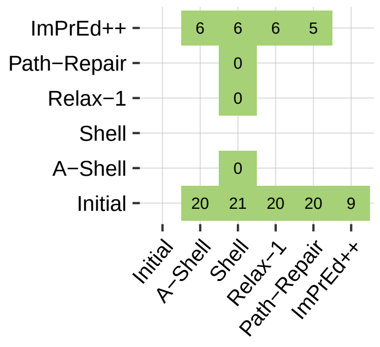


(a) Selection

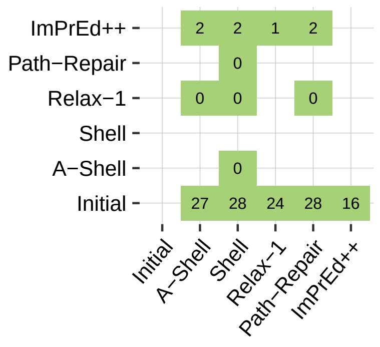


(b) Relax-x

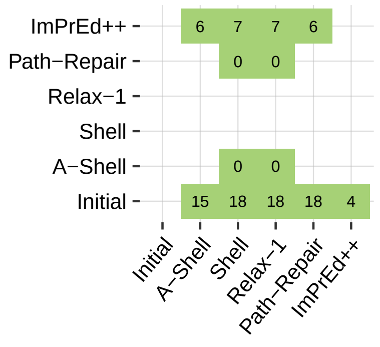
Figure 13: The minimum δ for each configuration (x-axis) such that it has an advantage over a δ-drawing, factored by the classes $\mathcal{L}$, $\mathcal{M}$, and $\mathcal{H}$ (y-axis).



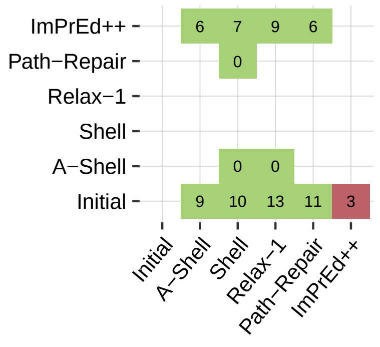
(a) $\mathcal{L} + \mathcal{M} + \mathcal{H}$



(b) $\mathcal{L}$



(c) $\mathcal{M}$



(d) $\mathcal{H}$

Figure 14: Advantage of each configuration (x-axis) compared to each configuration (y-axis), factored by the classes $\mathcal{L}$, $\mathcal{M}$, and $\mathcal{H}$.

Table 2: Median and mean values of the deviation angle for the different algorithms applied to ImPrEd as initial drawing.

	$\mathcal{L}+\mathcal{M}+\mathcal{H}$		$\mathcal{L}$		$\mathcal{M}$		$\mathcal{H}$	
	med	mean	med	mean	med	mean	med	mean
Initial	39.6	44.3	46.2	49.8	36.6	42.1	35.8	41.6
A-Shell	6.15	9.73	0.18	2.98	9.62	10.8	17.0	16.5
Shell	4.92	8.71	0.08	2.47	7.73	9.51	16.5	15.5
Relax-1	7.12	9.45	2.46	6.11	8.06	10	11.4	12.8
Relax-2	9.02	12.4	4.45	10.1	10.3	13.2	11.4	13.6
Relax-4	11.6	15.9	7.63	15.2	12.7	16.3	12.5	16.0
Relax-6	11.6	17.2	7.03	16.0	13.2	18.1	12.4	17.0
Relax-8	12.1	17.5	9.60	18.1	12.7	17.5	13.9	16.9
Path Repair	6.76	10.4	0.29	4.17	9.49	11.1	18.2	17.2
ImPrEd++	19.5	32.2	4.47	21.5	25.2	35.3	30.3	40.3

significance level of $\alpha = 0.05$. Hence, in the following we only consider ImPrEd for the initial drawings.

Hypothesis II) Advantage over the Initial drawing For each configuration, the binomial test supports this hypothesis, i.e., the advantage over the initial drawing, independent of the configuration, is at least 20°; see Fig. 14a. Note that the advantage over the Initial drawing decreases with the length of the longest planarization path in a drawing; refer to Fig 14b-14d. Moreover, Fig. 14d shows that for class $\mathcal{H}$ we were not able to verify that ImPrEd++ does have an advantage over the Initial drawing.

Hypothesis III) Advantage over ImPrEd++ Fig. 14a shows that for $\Delta = 5$ we can accept the hypothesis with high confidence for each configuration. Note that the advantage depends on the class, i.e, for class $\mathcal{L}$ the advantage decreases to 2°. For the configuration Relax-1 the advantage decreases even further to only 1°. Fig. 13a shows that ImPrEd has an advantage over 16°-drawings, i.e., a δ-drawing with $\delta = 16°$, but has no advantage over 15°-drawings. On the other hand, for example, Shell has an advantage over 2°-drawings.

Hypothesis IV) Relax-1 has an advantage over Relax-x. Fig. 15 confirms this hypothesis. Note that with respect to class $\mathcal{H}$, Relax-6 has an advantage over 20°-drawings, where as Relax-1 only has an advantage over 22°-drawings. The δ-values for the class $\mathcal{M}$ suggest that Relax-4 computes drawings with smaller angles than Relax-6, and Relax-6 drawings with smaller angle than Relax-8. Observe that this conclusion can not be drawn from the statistics listed in Tab. 2.

Hypothesis V) Angle relaxation helps with long planarization paths Fig. 14d shows for instances of the class $\mathcal{H}$ that the Relax-1 configuration has

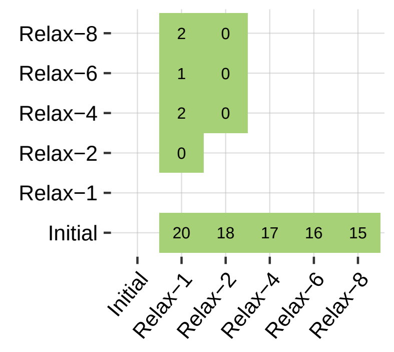


Figure 15: Advantages of the Relax-x configurations.

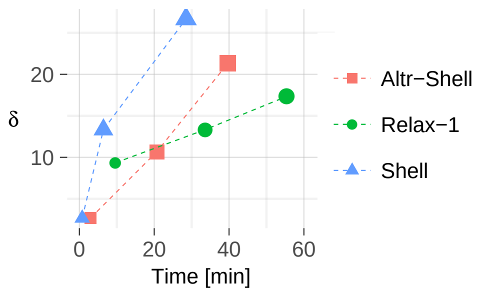


Figure 16: Time until convergence versus the δ-value. Symbol sizes indicate the classes $\mathcal{L}$, $\mathcal{M}$, and $\mathcal{H}$. Note: the δ-values of both figures are not coincident due to different experimental setups. The setup for the quality assessment does not allow a running time analysis.

a (small) advantage over the Alternating Shell. Further, Fig. 13a shows that this configuration tends to produce smaller deviation angle in instances of $\mathcal{M}$ and $\mathcal{H}$ in comparison to the remaining configurations, i.e., Relax-1 has an advantage over a 12°-drawing and 22°-drawing, respectively. The remaining configuration we were only able to show that they have an advantage over δ-drawings for larger values of δ.

Hypothesis VI) Path Repair helps with long planarization paths Figure 14d shows that the test on the training set does not conjecture a significant advantage of the Path Repair order over the remaining configurations. Hence, we have to reject this hypothesis.

5.4 Running Time

Force-directed methods have been engineered over the past decades. Hence, it is reasonable that the running time of ImPrEd++ is much faster in comparison

Table 3: Mean running time measurements for each configuration.

Configuration	**Time per Iteration**			**Total Time**		
	$\mathcal{L}$	$\mathcal{M}$	$\mathcal{H}$	$\mathcal{L}$	$\mathcal{M}$	$\mathcal{H}$
A-Shell	8.3s	15.0s	24.8s	2.9min	20.6min	39.6min
Shell	8.1s	18.0s	25.1s	0.7min	6.4min	28.4min
Relax-1	8.2s	20.3s	33.5s	9.5min	33.6.min	55.3min

Table 4: Number of graphs and dummies vertices per class for the north and community graphs.

	north		community	
	n. of graphs	n. of dummies	n. of graphs	n.of dummies
$\mathcal{L}$	87	1696	37	1062
$\mathcal{M}$	8	1128	25	1103
$\mathcal{H}$	5	2904	38	2041

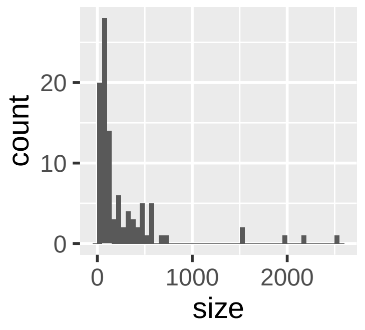
to our approach that heavily relies on geometric operations. On the other hand, the quality of the drawings obtained by our approach is significantly better than the quality of the drawings obtained by ImPrEd++. Therefore, we only evaluate the running time of our Geometric Planarization Drawing approach; see Table 3.

Running Time vs. Quality. We use the δ-values to compare the quality of the drawings with respect to the running time. Each point in Fig. 16 represents final drawings of a different configuration, divided into the introduced classes. The figure compares the average running time required to compute the final drawing against the smallest δ computed with the introduced methodology; all δ-values can be accepted with high confidence. For class $\mathcal{L}$ the configuration the (Alternating) Shell configurations achieves small angles and require only few minutes to finish. With increasing complexity of the drawings the relevance of the angle relaxation increases. For class $\mathcal{M}$ the Alternating Shell configuration has the smallest δ-value but is slower than the Shell configuration. For drawings of class $\mathcal{H}$, there is no clear dominance. In class $\mathcal{H}$ the Relax-1 configuration yields the best results but the Shell configuration requires less time. We suggest to use the Shell configuration for less complex drawings and when computing time is relevant and for drawings with increasing complexity the Relax-1 configuration.

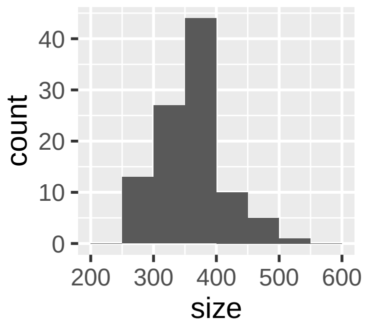
5.5 North and Community Graphs

In this section we augment our evaluation with a short analysis of two further benchmark datasets. The first dataset contains 100 randomly selected north graphs[5], and the second set contains 100 randomly generated community graphs, i.e., a set of graphs that resemble community structure. The community graphs

[5] `graphdrawing.org/data.html`



(a) North

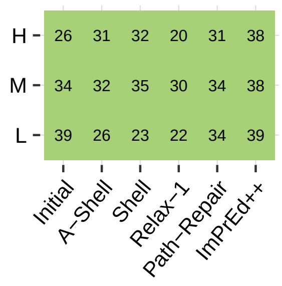


(b) Community

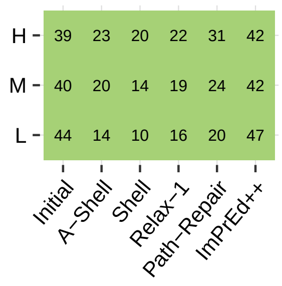
Figure 17: Size distribution of the north and community graphs.

Table 5: Median and mean values of the deviation angle for the different algorithms applied to ImPrEd as initial drawing.

	$\mathcal{L}+\mathcal{M}+\mathcal{H}$		$\mathcal{L}$		$\mathcal{M}$		$\mathcal{H}$	
	med	mean	med	mean	med	mean	med	mean
north								
Initial	20.1	29.5	25.2	34.7	20.9	30.9	17.4	26.0
A-Shell	20.4	21.8	15.6	18.4	22.1	24.4	21.9	22.7
Shell	19.8	21.4	14.4	16.5	23.0	24.4	21.7	23.0
Relax-1	14.4	18.2	12.8	17.2	18.7	22.3	14.0	17.1
Path Repair	21.0	26.5	21.0	30.3	20.9	30.9	21.1	22.6
ImPrEd++	24.7	32.3	24.3	31.1	24.8	33.2	24.8	32.6
community								
Initial	27.3	37.6	30.1	40.4	25.6	36.7	26.3	36.6
A-Shell	13.5	17.6	8.31	13.6	13.8	17.9	16.2	19.5
Shell	10.3	15.2	5.32	11.1	9.65	15.0	13.7	17.4
Relax-1	12.3	16.3	8.80	12.8	12.6	16.6	14.2	18.0
Path Repair	17.3	23.7	12.9	17.2	16.0	21.4	20.5	28.4
ImPrEd++	28.8	35.6	30.3	36.3	28.0	34.9	28.4	35.5



(a) North

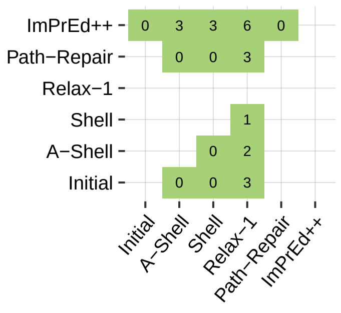


(b) Community

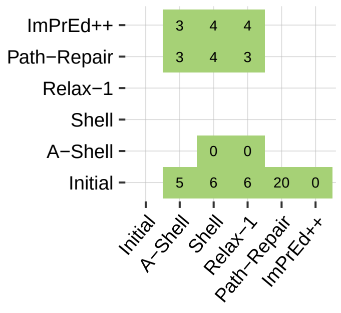
Figure 18: The minimum δ for each configuration (x-axis) such that it has an advantage over a δ-drawing, factored by the classes $\mathcal{L}$, $\mathcal{M}$, and $\mathcal{H}$ (y-axis).

have been used in the evaluation for heuristics to minimize crossings in straight-line drawings of graphs [24]. Fig. 17 shows the size distribution of the north and community graphs. The north graphs are at most 32-planar and the community graphs are at most 9-planar. For the north graphs the class $\mathcal{L}$ contains graphs with $k \leq 10$, $\mathcal{H}$ contains the graphs with $k > 20$, the remaining graphs belong to $\mathcal{M}$. In case of the community graphs the parameters are selected as follows, $\mathcal{L}$ and $\mathcal{H}$ contains graphs with $k < 6$ and $k \geq 7$, respectively, and $\mathcal{M}$ contains the remaining graphs. Tab. 4 lists the number of graphs and dummy vertices for each of the graph classes. The analysis of the rome graphs showed that using ImPrEd as an initial drawing resulted overall in drawings with smaller deviation angles compared to Gc. Therefore, we use ImPrEd to compute the initial layout of the north and community graphs as well. Moreover, Relax-1 computes drawings with significantly smaller deviation angles than the remaining Relax-x configurations on the rome graphs. Therefore, we abstain from evaluating the Relax-x configurations in this section for $x \neq 1$. Tab. 5 lists the mean and median values for the north and community graphs.

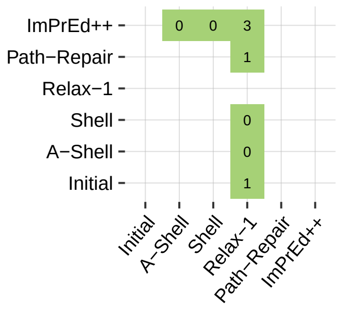
For the community graphs, we can confirm that all heuristics, except ImPrEd++, improve the deviation angles; see Fig. 20. Notice that this statement is only true for the class $\mathcal{L}$ of the north graphs but not for the classes $\mathcal{M}$ and $\mathcal{H}$; refer to Fig. 19. Recall that in case of the north graphs $\mathcal{M}$ contains graphs that are at least 11-planar and all community and rome graphs at most 13-planar. This confirms the observation that the deviation angle in drawings with many crossing per edge are difficult to optimize. The analysis of the rome graph already indicated that the Relax-1 configuration improves the deviation angle in graphs with long planarization graphs. Fig. 18a and Fig. 19 shows that the Relax-1 observation is the only configuration that is able to significantly improve the deviation angle compared to the initial drawing.



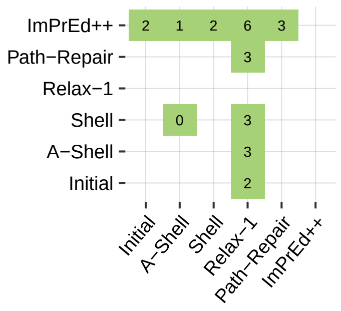
(a) $\mathcal{L} + \mathcal{M} + \mathcal{H}$



(b) $\mathcal{L}$



(c) $\mathcal{M}$

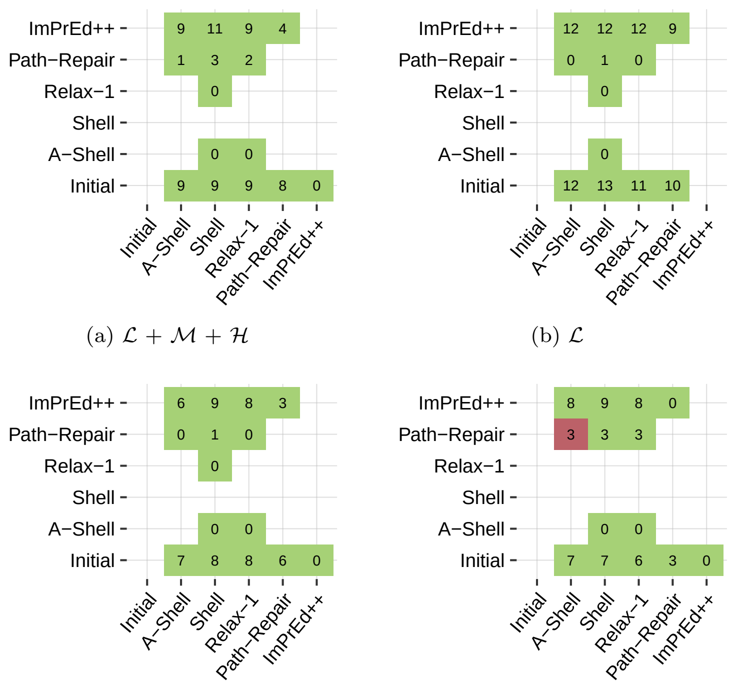


(d) $\mathcal{H}$

Figure 19: North: Advantage of each configuration (x-axis) compared to each configuration (y-axis), factored by the classes $\mathcal{L}$, $\mathcal{M}$, and $\mathcal{H}$.

5.6 Sample Drawings

For three graphs Fig. 21 shows the initial drawing and the drawing after the application of the Shell configuration of our Geometric Planarization Drawing approach. Observe that the optimization of the tail and dummy vertices in our Geometric Planarization Drawing approach can force a vertex v to be close to an edge which is not incident to v. To increase the readability of the drawings ImPrEd can help resolve this issue, i.e., to increase the vertex-edge distances. In case that we want to guarantee, that the deviation angles in the drawings do not change, we apply forces only to independent vertices. We observed that this strategy can be too restrictive, i.e., the vertex-edge distance remains small, since tail and dummy vertices restrict the movement of the independent vertices. Thus, we propose the following post-processing strategy, that relies on the assumption that ImPrEd, with additional planarization forces, does not alter the deviation angles too much. (i) Replace all planarization paths with an edge from the source to the target vertex if this edge crosses exactly the same edges as the path does. (ii) Apply ImPrEd with planarization forces on remaining dummy and tail vertices on the new drawing. The third column in



(a) $\mathcal{L} + \mathcal{M} + \mathcal{H}$ (b) $\mathcal{L}$

(c) $\mathcal{M}$ (d) $\mathcal{H}$

Figure 20: Community: Advantage of each configuration (x-axis) compared to each configuration (y-axis), factored by the classes $\mathcal{L}$, $\mathcal{M}$, and $\mathcal{H}$.

Fig. 21 shows examples of drawings that are obtained by this post-processing strategy.

6 Conclusion

We presented two approaches for drawing planarizations such that the edges of the original (non-planar) graph are as straight as possible. Our experiments show that the Geometric Planarization Drawing approach has an significant advantage over our adaptation of the force-directed algorithm ImPrEd. For instances with short planarization paths, we get very good deviation angles. Even though the deviation angles are worse for instances with longer planarization paths, our Geometric Planarization Drawing approach still significantly improves the angles of the initial drawing. Concerning future research, it would be interesting to investigate the effect of different initial drawings and to see how our geometric approach in Sec. 4 performs when additional optimization criteria such as the angular resolution are incorporated.

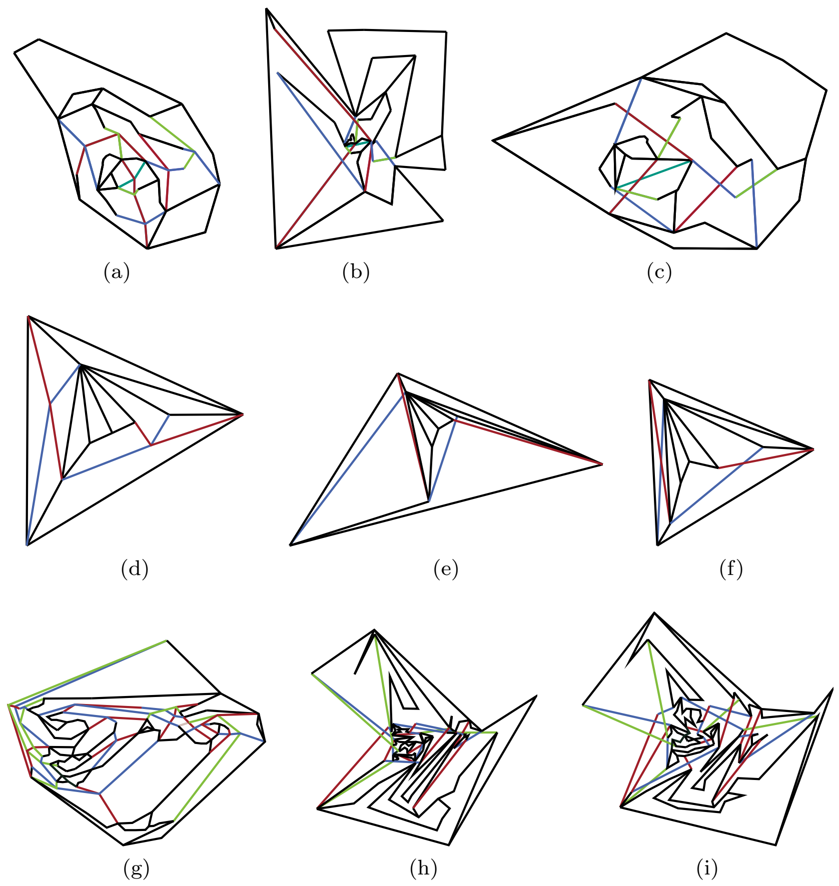


Figure 21: (a,d,g) Initial drawings, (b,e,h) Final drawing computed with the Shell configuration, (c,f,i) Drawing with the post processing step. Planarization paths are indicated by colors. (a,b,c) A rome graph. (d,e,f) A north graph. (g,h,j) A community graph.

References

[1] F. Bertault. A Force-Directed Algorithm that Preserves Edge-Crossing Properties. *Information Processing Letters*, 74(1–2):7–13, 2000. `doi:10.1016/S0020-0190(00)00042-9`.

[2] T. Bläsius, M. Radermacher, and I. Rutter. How to Draw a Planarization. In B. Steffen, C. Baier, M. van den Brand, J. Eder, M. Hinchey, and T. Margaria, editors, *Proceedings of the 43rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'17)*, pages 295–308. Springer International Publishing, 2017. `doi:10.1007/978-3-319-51963-0_23`.

[3] B. M. Ábrego, S. Fernández–Merchant, J. Leaños, and G. Salazar. The Maximum Number of Halving Lines and the Rectilinear Crossing Number of K_n for n ≤ 27. *Electronic Notes in Discrete Mathematics*, 30:261 – 266, 2008. `doi:10.1016/j.endm.2008.01.045`.

[4] C. Buchheim, M. Chimani, C. Gutwenger, M. Jünger, and P. Mutzel. Crossings and Planarization. In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, pages 43–85. Chapman and Hall/CRC, 2013.

[5] E. W. Chambers, D. Eppstein, M. T. Goodrich, and M. Löffler. Drawing Graphs in the Plane with a Prescribed Outer Face and Polynomial Area. *Journal of Graph Algorithms and Applications*, 16(2):243–259, 2012. `doi:10.7155/jgaa.00257`.

[6] T. M. Chan, F. Frati, C. Gutwenger, A. Lubiw, P. Mutzel, and M. Schaefer. Drawing Partially Embedded and Simultaneously Planar Graphs. *Journal of Graph Algorithms and Applications*, 19(2):681–706, 2015. `doi:10.7155/jgaa.00375`.

[7] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, and P. Mutzel. The open graph drawing framework (ogdf). In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, pages 543–569. Chapman and Hall/CRC, 2013.

[8] G. Da Lozzo, V. Dujmovic, F. Frati, T. Mchedlidze, and V. Roselli. Drawing Planar Graphs with Many Collinear Vertices. *Journal of Computational Geometry*, 9(1):94–130, 2018. `doi:10.20382/jocg.v9i1a4`.

[9] W. Didimo, G. Liotta, and F. Montecchiani. A Survey on Graph Drawing Beyond Planarity. *ACM Computating Surveys*, 52(1):4:1–4:37, 2019. `doi:10.1145/3301281`.

[10] W. Didimo, G. Liotta, and S. A. Romeo. Topology-Driven Force-Directed Algorithms. In U. Brandes and S. Cornelsen, editors, *Proceedings of the 18th International Symposium on Graph Drawing (GD'10)*, volume 6502 of *Lecture Notes in Computer Science*, pages 165–176. Springer, 2011. `doi:10.1007/978-3-642-18469-7\_15`.

[11] T. Dwyer, K. Marriott, and M. Wybrow. Topology Preserving Constrained Graph Layout. In I. G. Tollis and M. Patrignani, editors, *Proceedings of the 16th International Symposium on Graph Drawing (GD'08)*, volume 5417 of *Lecture Notes in Computer Science*, pages 230–241. Springer Berlin/Heidelberg, 2009. `doi:10.1007/978-3-642-00219-9\_22`.

[12] P. Eades, S.-H. Hong, G. Liotta, N. Katoh, and S.-H. Poon. Straight-Line Drawability of a Planar Graph Plus an Edge. In F. Dehne, J.-R. Sack, and U. Stege, editors, *Proceedings of the 14th International Symposium on Algorithms and Data Structures (WADS'15)*, volume 9214 of *Lecture Notes in Computer Science*, pages 301–313. Springer International Publishing, 2015. `doi:0.1007/978-3-319-21840-3\_25`.

[13] E. D. Giacomo, P. Eades, G. Liotta, H. Meijer, and F. Montecchiani. Polyline Drawings with Topological Constraints. In W.-L. Hsu, D.-T. Lee, and C.-S. Liao, editors, *Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC'18)*, volume 123 of *Leibniz International Proceedings in Informatics*, pages 39:1–39:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. `doi:10.4230/LIPIcs.ISAAC.2018.39`.

[14] L. Grilli, S.-H. Hong, J. Kratochvíl, and I. Rutter. Drawing Simultaneously Embedded Graphs with Few Bends. In C. Duncan and A. Symvonis, editors, *Proceedings of the 22nd International Symposium on Graph Drawing (GD'14)*, volume 8871 of *Lecture Notes in Computer Science*, pages 40–51. Springer Berlin/Heidelberg, 2014. `doi:10.1007/978-3-662-45803-7\_4`.

[15] C. Gutwenger, P. Mutzel, and R. Weiskircher. Inserting an Edge Into a Planar Graph. *Algorithmica*, 41(4):289–308, 2005. `doi:10.1007/s00453-004-1128-8`.

[16] S. Huber and M. Held. Motorcycle graphs: Stochastic Properties Motivate an Efficient yet Simple Implementation. *Journal of Experimental Algorithmics*, 16:1–3, 2011. `doi:10.1145/1963190.2019578`.

[17] S. Huber and M. Held. A Fast Straight-Skeleton Algorithm Based on Generalized Motorcycle Graphs. *International Journal of Computational Geometry & Applications*, 22(05):471–498, 2012. `doi:10.1142/S0218195912500124`.

[18] B. Joe and R. B. Simpson. Corrections to Lee's Visibility Polygon Algorithm. *BIT Numerical Mathematics*, 27(4):458–473, 1987.

[19] G. Kant. Drawing Planar Graphs Using the Canonical Ordering. *Algorithmica*, 16(1):4–32, 1996. `doi:10.1007/BF02086606`.

[20] T. Mchedlidze, M. Nöllenburg, and I. Rutter. Extending Convex Partial Drawings of Graphs. *Algorithmica*, 76(1):47–67, 2016. `doi:10.1007/s00453-015-0018-6`.

[21] T. Mchedlidze, M. Radermacher, and I. Rutter. Aligned Drawings of Planar Graphs. *Journal of Graph Algorithms and Applications*, 22(3):401–429, 2018. doi:10.7155/jgaa.00475.

[22] N. E. Mnëv. The Universality Theorems on the Classification Problem of Configuration Varieties and Convex Polytopes Varieties. In O. Y. Viro and A. M. Vershik, editors, *Topology and Geometry — Rohlin Seminar*, volume 1346 of *Lecture Notes in Mathematics*, pages 527–543. Springer Berlin/Heidelberg, 1988. doi:10.1007/BFb0082792.

[23] J. Nievergelt and F. P. Preparata. Plane-Sweep Algorithms for Intersecting Geometric Figures. *Communications of the ACM*, 25(10):739–747, 1982. doi:10.1145/358656.358681.

[24] M. Radermacher, K. Reichard, I. Rutter, and D. Wagner. Geometric Heuristics for Rectilinear Crossing Minimization. *Journal of Experimental Algorithmics*, 24(1):1.12:1–1.12:21, 2019. doi:10.1145/3325861.

[25] M. Radermacher and I. Rutter. Inserting an Edge into a Geometric Embedding. In T. C. Biedl and A. Kerren, editors, *Proceedings of the 26th International Symposium on Graph Drawing (GD'18)*, volume 11282 of *Lecture Notes in Computer Science*, pages 402–415. Springer International Publishing, 2018. doi:10.1007/978-3-030-04414-5_29.

[26] M. Schaefer. Complexity of Some Geometric and Topological Problems. In D. Eppstein and E. R. Gansner, editors, *Proceedings of the 17th International Symposium on Graph Drawing (GD'09)*, volume 5849 of *Lecture Notes in Computer Science*, pages 334–344. Springer Berlin/Heidelberg, 2010. doi:10.1007/978-3-642-11805-0_32.

[27] D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman and Hall/CRC, 2003.

[28] P. Simonetto, D. Archambault, D. Auber, and R. Bourqui. ImPrEd: An Improved Force-Directed Algorithm that Prevents Nodes from Crossing Edges. *Computer Graphics Forum*, 30(3):1071–1080, 2011. doi:10.1111/j.1467-8659.2011.01956.x.

[29] R. Tamassia. On Embedding a Graph in the Grid with the Minimum Number of Bends. *SIAM Journal on Computing*, 16(3):421–444, 1987. doi:10.1137/0216030.

[30] C. Thomassen. Rectilinear Drawings of Graphs. *Journal of Graph Theory*, 12(3):335–341, 1988. doi:10.1002/jgt.3190120306.

[31] W. T. Tutte. How to Draw a Graph. *Proceedings of the London Mathematical Society*, s3-13(1):743–767, 1963.