How to Draw a Planarization

Thomas Bläsius[1,2], Marcel Radermacher[1(✉)], and Ignaz Rutter[1]

[1] Faculty of Informatics, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{radermacher,rutter}@kit.edu

[2] Research Group Algorithm Engineering, Hasso Plattner Institute, Potsdam, Germany
thomas.blaesius@hpi.de

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 [18], the second approach, which we call *Geometric Planarization Drawing*, iteratively moves vertices to their locally optimal positions in the given initial drawing.

1 Introduction

In his seminal paper "How to Draw a Graph" [20], Tutte showed that every planar graph admits a planar straight-line drawing. His result has been strengthened in various ways, e.g., improving the running time and the required area [3]. 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 (rectilinear crossing number) is $\exists\mathbb{R}$-complete, i.e., as hard as the existential theory of the reals [16]. 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 $n \leq 27$ [1]. We thus follow the second approach, i.e., we insist on a small (though not necessarily minimal) 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 [2] for a survey. The output of these algorithms

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. Work was partially supported by grant WA 654/21-1 of the German Research Foundation (DFG).

DOI: 10.1007/978-3-319-51963-0_23

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 [19] and the fundamental ($\exists\mathbb{R}$-complete) problem Stretchability, which asks whether a given arrangement of pseudo-lines can be realized by geometric lines [14]. There have been several other works that consider the problem of realizing a given combinatorial description of a drawing geometrically. Hong et al. [9] give a characterization and testing algorithm for 1-planar graphs that admit a straight-line drawing. Grilli et al. [7] study the problem of realizing a given simultaneous planar embedding of two (or more) graphs with few bends per edge. Feng et al. [6] study trade-offs between straightness and area of drawings of clustered graphs where clusters are represented by convex drawings. The algorithm of Dwyer et al. [5] minimizes the stress of a layout while preserving the topology of the drawing. Didimo et al. [4] present an algorithm that is able to preserve the topology unless changing the topology improves the number of crossings. Simonetto et al. [18] improve a known force-directed layout algorithm for planar graphs that preserves the combinatorial embedding of the input drawing.

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. Throughout, we assume without loss of generality that G_p is biconnected; see Appendix. We present two approaches, one is based on an adaption of ImPrEd that includes additional forces to facilitate straightening the planarization paths (Appendix). 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 (Sect. 3). 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 (Sect. 4). 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,

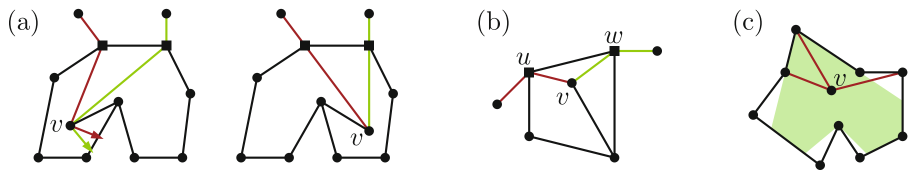


Fig. 1. (a) An initial drawing (left) that is difficult to repair using the force-directed algorithm although v could be moved to an optimal position without violating planarity (right). (b) The closer v lies to the edge uw, the better are the v-active angles. (c) The (green) planarity region of v. (Color figure online)

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 *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 $e = uw \in E \backslash E'$, 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 $e, e' \in E \backslash E'$ the paths p_e and $p_{e'}$ are edge-disjoint, and (v) the paths p_e, $e \in E \backslash E'$ cover all edges in E_p. We call the planarization G_p *k-planar* if the longest planarization path has k dummy vertices.

A *dissected pair* (u, v, w) is a pair $uv, vw \in E_p$ of edges that belong to the same planarization path. The *crossing angle* cr-$\alpha(u, v, w)$ of (u, v, w) is the angle cr-$\alpha(u, v, w) = \pi - \angle(u, v, w)$; A crossing angle is *active* with respect to v (also called *v-active*) if moving v can alter that angle. For a dissected pair (u, v, w), v is a dummy vertex and u and w are *tail* vertices. 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* 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 [10]. A *geometric center* of a polygon P is obtained by shrinking P to a single point.

3 Geometric Planarization Drawing

The spring embedder described in the appendix 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. 1a. 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 $\mathcal{PR}(v)$. The (non-convex) planarity region is independent of the geometric position of v within it surrounding. We show in Sect. 3.1 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 crossing angles when placing v to p. A point in $\mathcal{PR}(v)$ is a *locally optimal* position for v if $\mathcal{PR}(v)$ contains no other point with strictly smaller cost. In Sect. 3.2, 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 up to a limited number of iterations.

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 Sect. 4.

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. 1b. We therefore shrink $\mathcal{PR}(v)$ slightly and consider it to be a closed set; see Sect. 4 for more details.

3.1 Planarity Region

Let G_p be a planarization with a given drawing and let v be a vertex of G_p. Let $N(v)$ be the neighbors of v and 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 simple polygon $\operatorname{surr}(v)$, 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. 1c. Thus, the planarity region $\mathcal{PR}(v)$ is the intersection of all visibility regions in $\operatorname{surr}(v)$ 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 $\operatorname{surr}(v)$, and then intersecting these visibility polygons. Let n_v be the number of vertices of the surrounding polygon $\operatorname{surr}(v)$ and let d_v be the degree of v. Computing the d_v visibility polygons takes $O(d_v n_v)$ time [12]. To intersect these d_v visibility polygons (each having size $O(n_v)$), one can use a sweep-line algorithm [15] 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 can improve this running time as stated in the following theorem; see Appendix for a proof.

Theorem 1. *$\mathcal{PR}(v)$ has size $O(n_v)$ and can be computed in $O(d_v n_v \log n_v)$ time.*

Now assume $\mathrm{surr}(v)$ is not a simple polygon. As we assume G_p to be biconnected, $\mathrm{surr}(v)$ has a single connected component. It may, however, have cutvertices with multiple incidences to the interior of $\mathrm{surr}(v)$. We eliminate this issue by slightly shrinking $\mathrm{surr}(v)$, yielding a simple polygon. Another special case is the outer face. However, we can treat it like an interior face by basically placing the whole drawing in a box.

3.2 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.

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; see Fig. 2a. 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 $\ell(t)$

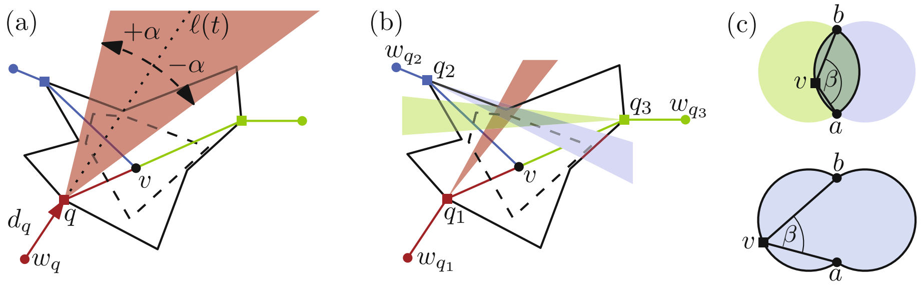


Fig. 2. (a) A cone with respect to one neighbor q of v. (b) The intersection of all cones with the planarity region (dashed) includes possible positions for the vertex v. (c) The angle $\angle avb$ is at least β for $\beta > 90°$ ($\beta < 90°$) if and only if v lies in the intersection (union) of two discs (including its boundary, but excluding a and b).

(for $t > 0$) results in the crossing angle cr-$\alpha(w_q, q, v) = 0$. Moreover, all points in the plane that yield cr-$\alpha(w_q, q, v) \leq \alpha$ 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. 2b. 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 [15]. 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 $O((n_v + d_v^2) \log n_v)$ time.

Lemma 1. *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 $O(\log(1/\epsilon)(n_v + d_v^2) \log n_v)$.*

Placing a Pure Dummy Vertex. A pure dummy vertex v has only two active crossing 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 cr-$\alpha(a, v, b) \leq \alpha$), one has to place v in the intersection of two discs (union if $\alpha > \pi/2$); see Fig. 2c. These two disks must have a and b on their boundary and basic geometry shows that their radii has to be $|ab|/(2\sin(\pi - \alpha))$ (which uniquely defines the two disks).

The same applies for $\angle pvq$. Thus, requiring both active crossing angles cr-$\alpha(a, v, b)$ and cr-$\alpha(p, v, q)$ 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 of the planarity region.

Lemma 2. *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 $O(\log(1/\epsilon) n_v)$.*

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 linear time in the size 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 $O(\log(1/\epsilon)(n_v + d_v^2) \log n_v)$.*

Overall Running Time. We have seen that the planarity region for a vertex v can be computed in $O(d_v n_v \log n_v)$ time (Theorem 1) and that a locally optimal position can be approximated in $O(\log(1/\varepsilon)(n_v + d_v^2) \log n_v)$ time. 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 surounding, the running time of computing the planarity region dominates the time for computing the locally optimal position. Each iteration thus needs $O(\sum_{v \in V} d_v n_v \log n_v)$ time. Bounding vertex face degrees improve the running time; see appendix.

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

4 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. The first set of experiments has a limited time contingent and the second runs until convergence limited by 100 iterations.

4.1 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.

Initial Drawing. Both, our geometric approach and our implementation of ImPrEd, improve an initial drawing of a planarization. While in principle an arbitrary planar straight-line drawing may be used for creating the initial drawing, we restrict ourselves to algorithms implemented within OGDF[1], which offers two algorithms: TutteLayout [20] and PlanarStraightLayout [13]. The former may generate drawings with exponentially bad resolution (creating problems with the floating point arithmetic). Hence, we cannot use these layouts as initial drawing. To gain a broader set of initial drawings we applied 100 iterations of the following two algorithms to the PlanarStraightLayout: (i) ImPrEd without the forces to optimize the planarization, (ii) the Geometric Center heuristic places every vertex in the geometric center of its planarity region. Due to space constraints we only present the results with ImPrEd as initial drawing. For these drawings we observe the worst initial crossing-angles but result in the potentially best overall quality. The results for the other initial drawings.

[1] The Open Graph Drawing Framework: ogdf.net.

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.

Placement of Independent Vertices. For independent vertices, every position in the planarity region is equally good since all crossing angles are inactive. To reduce the restrictions imposed by independent vertices on their neighbors, we suggest two placement strategies for them: Randomized Placement, which puts v at a random position in $\mathcal{PR}(v)$, and Geometric Center, where v is placed in the geometric center of $\mathcal{PR}(v)$.

Shrinking the Planarity Region. As mentioned before, a locally optimal position for a vertex v may not exists as $\mathcal{PR}(v)$ is an open set; see Fig. 1b. 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 D_v be the distance of v from the boundary of $\mathcal{PR}(v)$. We offset by the minimum of μD_B and D_v, where μ is a parameter. In our experiments we used $\mu = 0.1$. Note that shrinking by at most D_v ensures that the previous position of v remains valid. Thus, we do not have to move v to a worse position due to the shrinking.

Angle Relaxation. While the placement of the tail and hybrid vertices introduced in Sect. 3.2 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. When performing 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.

Configurations. The presented degrees of freedom allow for many different configurations of our algorithm. Due to space constraints, we focus on the three configurations shown in Table 1 (see Appendix for additional configurations).

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
Relax-1	Alternating-Shell	0.1

The drawing area is always limited by a box that is twice as large as the bounding box of the initial drawing and use the Geometric Center heuristic for independent vertices.

To allow a fair comparison between all algorithms, each algorithm gets exactly $5n$ s to optimize the drawings. For experiments regarding the running

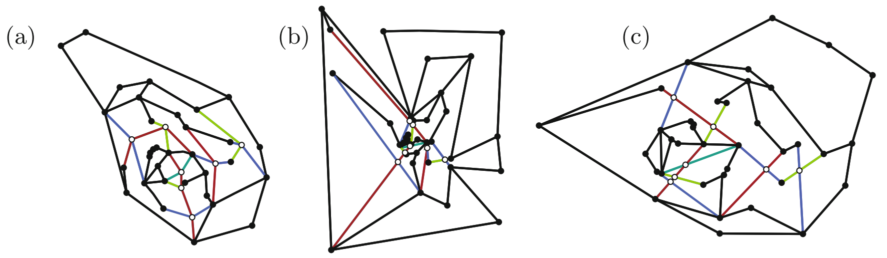


Fig. 3. (a): Initial drawing, (b): Final drawing computed with the Shell configuration (c) Drawing with the optional post processing step; see Appendix. Unfilled disks represent dummies.

time, we measure the time until convergence limited by 100 iterations. Figure 3 shows an example, where our geometric algorithm finds a nearly optimal solution; also see Appendix.

4.2 Experimental Setup

For a set of graphs $\mathcal{G}$ we want to compare the quality of two sets of drawings Γ_1, Γ_2 of these graphs. We use the crossing-angles to measure the quality of a drawing. Aggregating the crossing-angles per graph yields a loss of information, thus we compare the crossing-angles directly. Let D be the set of all dissected-pairs in $\mathcal{G}$ and $\{\text{cr-}\alpha_i(u, v, w) \mid (u, v, w) \in D\}$ the set of all crossing-angles in drawing $\Gamma_i, i = 1, 2$. The drawings Γ_1 have an *advantage* of $\Delta \in \mathbb{N}_0$ over the drawing Γ_2 if for more then 50% of the dissected pairs (u, v, w) the inequality $\text{cr-}\alpha_1(u, v, w) + \Delta < \text{cr-}\alpha_2(u, v, w)$ holds.

Further, we are interested in the smallest angle $\delta \in \mathbb{N}_0$ such that the angles in our drawings of a graph G_p are smaller then δ. We define a hypothetical drawing called δ-*drawing* where each crossing angle is δ. For each algorithm, we seek the smallest angle δ such that the resulting drawing has an advantage over the δ-drawing.

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($\mathcal{H}$). 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 $4 < k < 9$.

We ran the algorithms on 100 randomly selected non-planar Rome graphs[2]. For each of them, we used the (single) non-planar biconnected component. 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 $\mathcal{H}$. We compare the crossing angles directly and do not aggregate them per graph. Thus, we have 4012 samples in total (twice the number of dummy vertices). We partitioned the set of samples into a training set, containing 20% of the samples, and a verification set containing the remaining 80%.

[2] graphdrawing.org/data.html.

We use OGDF[3] to planarize the graphs [8] and to compute the initial drawing [13]. We use the libraries CGAL[4] to compute line arrangements, STALGO [10,11] to shrink polygons, and GMP[5] to represent coordinates.

4.3 Statistical Test

Our evaluation focuses on the comparison of crossing angles in different drawings of the same graph, e.g., the initial drawing vs. the final drawing of some algorithm. Since the underlying distribution of the angles is unknown and not likely to be, e.g., normal, the median and quantiles are not useful to compare two drawings. Instead we use a binomial test, which compares two dependent samples and is independent of the underlying distribution [17].

For each dissected pair (u, v, w) we compare the crossing angles cr-$\alpha_1(u, v, w)$ and cr-$\alpha_2(u, v, w)$ generated by two different algorithms. The comparison cr-$\alpha_1(u, v, w) + \Delta <$ cr-$\alpha_2(u, v, w)$ yields a sequence of 0 s and 1 s. With the binomial test we check whether 1 s occur significantly more often than 0 s at a significance level of $\alpha = 0.05$.

In order to formulate our hypothesis we compute the maximum Δ such that the binomial test shows significance on the training set. In order to get a robust and likely hypothesis we choose $3/4 \cdot \Delta$ as the conjectured value. Hypothesis regarding the δ drawings conjecture that the angles are smaller then $4/3 \cdot \delta$, where δ was computed on the training set.

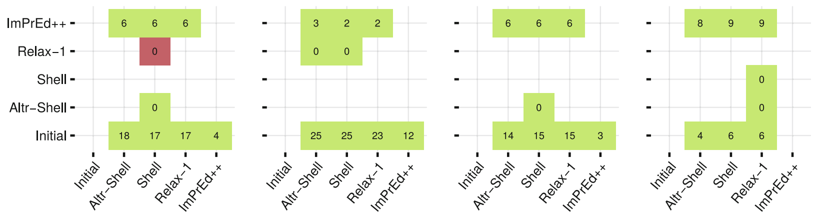
4.4 Quality of the Drawings

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

(I) Geometric Planarization Drawing approach and ImPrEd++ advantage of at least 4° over the initial drawing.
(II) Geometric Planarization Drawing has an advantage of at least 6° over ImPrEd++.
(III) In class $\mathcal{H}$, Relax-1 has an advantage over Alternating-Shell (due to the weakened influence of unplaced vertices).

We use Figs. 4 and 5a to show whether or not the binomial tests support our hypotheses. A value Δ in a cell in Fig. 4 shows that the algorithm on the x-axis has an hypothetical advantage of Δ over the algorithm on the y-axis. These values are computed on the training set. A green cell means that we can accept the hypothesis with a confidence of 95%. 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.

[3] ogdf.net.
[4] cgal.org.
[5] gmplib.org.



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

Fig. 4. Advantage of each configuration (x-axis) compared to each configuration (y-axis), factored by the classes $\mathcal{L}$, $\mathcal{M}$, and $\mathcal{H}$.

For example, in the class $\mathcal{H}$ (see Fig. 4d), we conjecture, based on the observation in the training set, that the drawings of the SHELL configuration have an advantage of 9° over the drawings of IMPRED++. Recall, that having an advantage means that 50% of the crossing angles, plus an additional buffer of 9°, of the first drawings are smaller then the crossing 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. 5a, for class $\mathcal{L}$ we can say with 95% confidence that 50% of the crossing angles of the SHELL configuration are smaller then the crossing-angles of a drawing where each crossing angle is 2°. We now discuss our hypotheses.

Hypothesis (I) Advantage over the INITIAL *drawing.* The binomial tests support this hypothesis for every configuration and for IMPRED++; see Fig. 4. Note that the advantage over the INITIAL drawing decreases with the length of the longest planarization path in a drawing. The Figure indicates, that IMPRED++ does not have an advantage over the INITIAL drawing on long planarization paths. Figure 4a indicates that there is support for the hypothesis when considering all instances (not separated into classes).

Hypothesis (II) Advantage over IMPRED++. Figure 4 shows that for $\Delta = 6$ we have can accept the hypothesis with high confidence for every configuration and class.

Hypothesis (III) Angle relaxation helps with long planarization paths. Figure 4d shows for instances of the class $\mathcal{H}$ that the RELAX-1 configurations has a (small) advantage. Figure 5a further shows that this configuration tends to produce smaller crossing angle in instances of $\mathcal{H}$ in comparison to the other configurations.

4.5 Running Time

We conclude the Section with a running time analysis. Table 2 shows a descriptive evaluation of the running time of our Geometric Planarization Drawing approach.

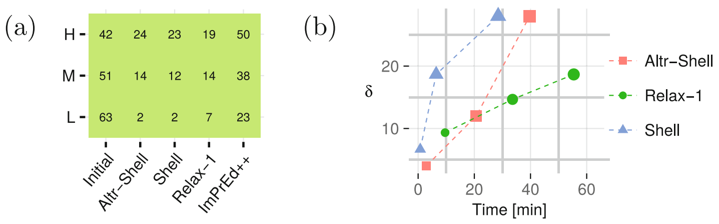


Fig. 5. (a) The minimum δ for each configuration (x-axis) such that it has an advantage over a δ-drawing, factored by the classes $\mathcal{L}$, $\mathcal{M}$, and $\mathcal{H}$ (y-axis). (b) Time until convergence versus the δ-value. Symbol sizes indicate the classes $\mathcal{L}$, $\mathcal{M}$, and $\mathcal{H}$. 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.

Table 2. Running time measurements for each configuration.

Configuration	Time per iteration			# iterations			Total time		
	$\mathcal{L}$	$\mathcal{M}$	$\mathcal{H}$	$\mathcal{L}$	$\mathcal{M}$	$\mathcal{H}$	$\mathcal{L}$	$\mathcal{M}$	$\mathcal{H}$
ALTERNATING SHELL	8.3 s	15.0 s	24.8 s	20.2	82.4	93.0	2.9 min	20.6 min	39.6 min
SHELL	8.1 s	18.0 s	25.1 s	5.5	22.4	66.6	0.7 min	6.4 min	28.4 min
RELAX-1	8.2 s	20.3 s	33.5 s	59.8	100.0	100.0	9.5 min	33.6 min	55.3 min

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. 5b 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-value can be accepted with high confidence. For class $\mathcal{L}$ the configuration the (ALTERNATING) SHELL configurations have 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 then 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 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 adaption of the force-directed algorithm ImPrEd. For instances with short planarization paths, we get very good crossing angles. Even though the crossing 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 Sect. 3 performs when additional optimization criteria such as the angular resolution are incorporated.

References

1. Ábrego, B.M., Fernández-Merchant, S., Leaños, J., Salazar, G.: The maximum number of halving lines and the rectilinear crossing number of kn for n $\leq$ 27. Electron. Notes Discrete Math. **30**, 261–266 (2008)
2. Buchheim, C., Chimani, M., Gutwenger, C., Jünger, M., Mutzel, P.: Crossings and planarization. In: Handbook of Graph Drawing and Visualization, pp. 43–85. Chapman and Hall/CRC (2013)
3. Chambers, E.W., Eppstein, D., Goodrich, M.T., Löffler, M.: Drawing graphs in the plane with a prescribed outer face and polynomial area. J. Graph Alg. Appl. **16**(2), 243–259 (2012)
4. Didimo, W., Liotta, G., Romeo, S.A.: Topology-driven force-directed algorithms. In: Brandes, U., Cornelsen, S. (eds.) GD 2010. LNCS, vol. 6502, pp. 165–176. Springer, Heidelberg (2011). doi:10.1007/978-3-642-18469-7_15
5. Dwyer, T., Marriott, K., Wybrow, M.: Topology preserving constrained graph layout. In: Tollis, I.G., Patrignani, M. (eds.) GD 2008. LNCS, vol. 5417, pp. 230–241. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00219-9_22
6. Feng, Q.-W., Cohen, R.F., Eades, P.: How to draw a planar clustered graph. In: Du, D.-Z., Li, M. (eds.) COCOON 1995. LNCS, vol. 959, pp. 21–30. Springer, Heidelberg (1995). doi:10.1007/BFb0030816
7. Grilli, L., Hong, S.-H., Kratochvíl, J., Rutter, I.: Drawing simultaneously embedded graphs with few bends. In: Duncan, C., Symvonis, A. (eds.) GD 2014. LNCS, vol. 8871, pp. 40–51. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45803-7_4
8. Gutwenger, C., Mutzel, P., Weiskircher, R.: Inserting an edge into a planar graph. Algorithmica **41**(4), 289–308 (2005)
9. Hong, S.-H., Eades, P., Liotta, G., Poon, S.-H.: Fáry's theorem for 1-planar graphs. In: Gudmundsson, J., Mestre, J., Viglas, T. (eds.) COCOON 2012. LNCS, vol. 7434, pp. 335–346. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32241-9_29
10. Huber, S., Held, M.: Motorcycle graphs: stochastic properties motivate an efficient yet simple implementation. J. Exper. Algo. **16**, 1–3 (2011)
11. Huber, S., Held, M.: A fast straight-skeleton algorithm based on generalized motorcycle graphs. Int. J. Comput. Geom. Appl. **22**(05), 471–498 (2012)
12. Joe, B., Simpson, R.B.: Corrections to Lee's visibility polygon algorithm. BIT Num. Math. **27**(4), 458–473 (1987)
13. Kant, G.: Drawing planar graphs using the canonical ordering. Algorithmica **16**(1), 4–32 (1996)
14. Mnev, N.E.: The universality theorems on the classification problem of configuration varieties and convex polytopes varieties. In: Viro, O.Y., Vershik, A.M. (eds.) Topology and Geometry — Rohlin Seminar. LNM, vol. 1346, pp. 527–543. Springer, Heidelberg (1988). doi:10.1007/BFb0082792

15. Nievergelt, J., Preparata, F.P.: Plane-sweep algorithms for intersecting geometric figures. Commun. ACM **25**(10), 739–747 (1982)
16. Schaefer, M.: Complexity of some geometric and topological problems. In: Eppstein, D., Gansner, E.R. (eds.) GD 2009. LNCS, vol. 5849, pp. 334–344. Springer, Heidelberg (2010). doi:10.1007/978-3-642-11805-0_32
17. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures. Chapman and Hall/CRC (2003)
18. Simonetto, P., Archambault, D., Auber, D., Bourqui, R.: ImPrEd: an improved force-directed algorithm that prevents nodes from crossing edges. Comput. Graph. Forum (EuroVis 2011) **30**(3), 1071–1080 (2011)
19. Tamassia, R.: On embedding a graph in the grid with the minimum number of bends. SIAM J. Comput. **16**(3), 421–444 (1987)
20. Tutte, W.T.: How to draw a graph. Proc. Lond. Math. Soc. **s3–13**(1), 743–767 (1963)