Geometric Heuristics for Rectilinear Crossing Minimization

MARCEL RADERMACHER, Karlsruhe Institute of Technology, Germany
KLARA REICHARD,
IGNAZ RUTTER, Universität Passau, Germany
DOROTHEA WAGNER, Karlsruhe Institute of Technology, Germany

In this article, we consider the *rectilinear crossing minimization problem*, i.e., we seek a straight-line drawing Γ of a graph $G = (V, E)$ with a small number of edge crossings. Crossing minimization is an active field of research [1, 10]. While there is a lot of work on heuristics for topological drawings, these techniques are typically not transferable to the rectilinear (i.e., straight-line) setting. We introduce and evaluate three heuristics for rectilinear crossing minimization. The approaches are based on the primitive operation of moving a single vertex to its crossing-minimal position in the current drawing Γ, for which we give an $O((kn + m)^2 \log(kn + m))$-time algorithm, where k is the degree of the vertex and n and m are the number of vertices and edges of the graph, respectively. In an experimental evaluation, we demonstrate that our algorithms compute straight-line drawings with fewer crossings than energy-based algorithms implemented in the Open Graph Drawing Framework [11] on a varied set of benchmark instances. Additionally, we show that the difference of the number of crossings of topological drawings computed with the edge insertion approach [10, 13] and the number of crossings in straight-line drawings computed by our heuristic is relatively small. All experiments are evaluated with a statistical significance level of $\alpha = 0.05$.

CCS Concepts: • **Theory of computation → Computational geometry**; • **Mathematics of computing → Graph algorithms**;

Additional Key Words and Phrases: Rectilinear crossing minimization, vertex movement, vertex insertion, edge insertion, experimental evaluation

ACM Reference format:
Marcel Radermacher, Klara Reichard, Ignaz Rutter, and Dorothea Wagner. 2019. Geometric Heuristics for Rectilinear Crossing Minimization. *J. Exp. Algorithmics* 24, 1, Article 1.12 (July 2019), 21 pages.
https://doi.org/10.1145/3325861

1 INTRODUCTION

The empirical study of Purchase et al. [30] indicates that a drawing of a graph with a small number of crossings is easier to comprehend than a drawing of the same graph with a large number of crossings. Consequently, the minimization of crossings has received considerable attention in

Work was partially supported by grants RU 1903/3-1 and WA 654/21-1 of the German Research Foundation (DFG). A preliminary version [31] of this paper appeared in the *Proceedings of the 20th Workshop on Algorithm Engineering and Experiments (ALENEX'18)*.
Authors' addresses: M. Radermacher, Karlsruhe Institute of Technology, Karlsruhe, Germany; email: radermacher@kit.edu; K. Reichard; I. Rutter, Universität Passau, Passau, Germany; email: rutter@fim.uni-passau.de; D. Wagner, Karlsruhe Institute of Technology, Karlsruhe, Germany; email: dorothea.wagner@kit.edu.

1084-6654/2019/07-ART1.12 $15.00
https://doi.org/10.1145/3325861

theory and in practice; the bibliography of Vrt'o is an impressive list of over 700 references [37]. A *topological drawing* of a graph is a drawing where each edge is a Jordan arc and in a *straight-line drawing* each edge is restricted to be a straight-line segment. The *crossing number* $\mathrm{cr}(G)$ of a graph G is the minimum crossing number of all possible topological drawings of G. The *rectilinear crossing number* $\overline{\mathrm{cr}}(G)$ of G is the minimum number of crossings over all possible straight-line drawings of G. Indeed, there is a family of graphs with a constant crossing number but an unbounded rectilinear crossing number [6]. Moreover, there is a difference in the algorithmic complexity of the respective minimization problem. The minimization of the crossing number is $\mathcal{NP}$-complete [21]. For the minimization of the rectilinear crossing number, only $\mathcal{NP}$-hardness is known; more precisely, the problem is $\exists\mathbb{R}$-complete [5]. Due to these gaps, we can either insist on a small number of crossings or on straight-line edges. In case of topological drawings iteratively inserting edges into a (planar) graph with a small number of crossings proved to be effective in practice [10, 13]. Unfortunately, deciding whether there is a straight-line drawing homeomorphic to a given drawing is $\exists\mathbb{R}$-complete [32, 34]. Based on the topological drawings with a small number of crossings, Bläsius et al. [7] heuristically straighten the edges. In general it is not possible to transfer the results on topological drawings to the geometric setting. Thus, if we insist on straight-line drawings, there is need for a geometric approach.

Several surveys [1, 10] show that the estimation of the (rectilinear) crossing number of complete graphs has received considerable attention. Most recently Fox et al. [16] introduced an $n^{2+o(1)}$-time algorithm that computes a straight-line drawing of a graph G with at most $\overline{\mathrm{cr}}(G) + o(n^4)$ pairs of crossing edges. This is a $1 + O(1)$ approximation for dense graphs but rather of theoretical interest for sparse graphs. A considerable number of known upper bounds for the rectilinear crossing number of the complete graphs K_n for $n \leq 100$ [2] is due to Fabila-Monroy and López [15].

Energy-based algorithms are a common way to compute straight-line drawings of arbitrary graphs. For a detailed description, we refer to the survey of Kobourov [26]. Energy-based algorithms are often designed to compute drawings with, e.g., uniform edge length or small stress. Kobourov claims that these algorithms tend to produce crossing-free drawings for planar graphs. The force-directed approach by Davidson and Harel [14] actively reduces the number of crossings among other optimization criteria. Apart from that, we are not aware of any algorithms that compute straight-line drawings with a small number of crossings.

Contribution and Outline. Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E and let Γ be a straight-line drawing of G. For a vertex $v \in V$ and a point $p \in \mathbb{R}^2$, we denote by $\Gamma[v \mapsto p]$ the straight-line drawing obtained from Γ by moving v to the point p. Based on the assumption that we are able to compute a drawing $\Gamma[v \mapsto p^\star]$ with a small number of crossings, we introduce in Section 2 three heuristics to compute drawings with few crossings. In Section 3, we show that a drawing $\Gamma[v \mapsto p^\star]$ with a minimum number of crossings can be computed in $O((kn + m)^2 \log(kn + m))$ time for a graph with n vertices, m edges, and a vertex v of degree k. In Section 4, we experimentally evaluate our algorithms and show that we achieve fewer crossings than energy-based algorithms implemented in the *Open Graph Drawing Framework* [11] with a statistical significance of $\alpha = 0.05$. Additionally, we compare our algorithm to topological drawings with a small number of crossings. We show that there is only a small gap between the number of crossings in topological and straight-line drawings of our benchmark instances. Throughout the remainder of this article, a *drawing* of a graph is a straight-line drawing.

2 A FRAMEWORK FOR RECTILINEAR CROSSING MINIMIZATION

Let v be a vertex of the graph $G = (V, E)$ and let Γ be a drawing of G. Recall that the drawing $\Gamma[v \mapsto p]$ is obtained from Γ by moving v to p. Assume that we are able to efficiently compute a

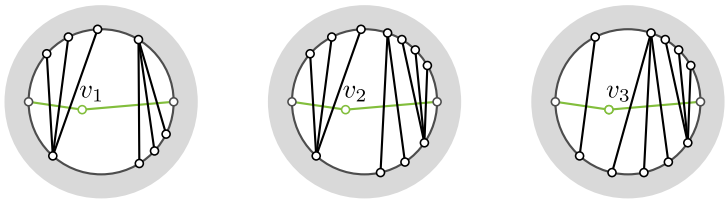


Fig. 1. Assume that structures in (a)–(c) are substructures of a common drawing Γ. Depending on the function $\mathrm{cr}_x(\Gamma,\cdot)$, the vertices are moved in a different ascending order. For $x = \textsc{Log}$, we have that $\mathrm{cr}_{\textsc{Log}}(\Gamma, v_3) < 3.81 \leq \mathrm{cr}_{\textsc{Log}}(\Gamma, v_1) = 4 < 4.5 < \mathrm{cr}_{\textsc{Log}}(\Gamma, v_2)$. For $x = \textsc{Sum}$, we have that $\mathrm{cr}_{\textsc{Sum}}(\Gamma, v_1) = 6 < \mathrm{cr}_{\textsc{Sum}}(\Gamma, v_3) = 7 < \mathrm{cr}_{\textsc{Sum}}(\Gamma, v_2) = 8$. For $x = \textsc{Sq}$, we have that $\mathrm{cr}_{\textsc{Sq}}(\Gamma, v_1) = 18 < \mathrm{cr}_{\textsc{Sq}}(\Gamma, v_2) = 34 < \mathrm{cr}_{\textsc{Sq}}(\Gamma, v_3) = 37$.

position $p^\star$ so the number of crossings is minimized over all drawings $\Gamma[v \mapsto p], p \in \mathbb{R}^2$. With this operation at hand, several possibilities arise to compute a drawing of G with a small rectilinear crossing number. We introduce three approaches. The *vertex movement approach* iteratively moves the vertices in some order to their locally optimal position. The *vertex insertion approach* starts from a large induced planar subgraph and inserts vertices at their locally optimal position. The *edge insertion approach* starts with a maximal planar subgraph and iteratively inserts edges into the drawing and locally modifies the drawing to reduce the number of crossings.

2.1 Vertex Movement Approach

Let $S = \langle v_1, v_2, \ldots, v_k\rangle$, $k \in \mathbb{N}$ be a sequence of vertices of G, and let Γ_0 be an arbitrary straight-line drawing of G. The drawing Γ_i is obtained from Γ_{i-1} by moving vertex v_i to its locally optimal position.

The number of crossings in Γ_n may depend on the order S. Hence, we introduce the following possibilities to choose S. As a baseline, we use a random permutation of V for S. We refer to this sequence as Random. To obtain other sequences S, we order the vertices V in descending or ascending order with respect to the number of crossings of v in the initial drawing Γ_0 of G. Denote by $E(v)$ the set of edges incident to v, and by $\mathrm{cr}(\Gamma, e)$ the number of crossings of an edge e in the drawing Γ. We propose the following ways to count the number of crossings incident to a vertex v. Figure 1 illustrates that these can yield different orders of the same vertex set.

$$\mathrm{cr}_{\textsc{Log}}(\Gamma, v) = \sum_{e\in E(v)} \log(\mathrm{cr}(\Gamma, e) + 1), \tag{1}$$

$$\mathrm{cr}_{\textsc{Sum}}(\Gamma, v) = \sum_{e\in E(v)} \mathrm{cr}(\Gamma, e), \tag{2}$$

$$\mathrm{cr}_{\textsc{Sq}}(\Gamma, v) = \sum_{e\in E(v)} \mathrm{cr}(\Gamma, e)^2. \tag{3}$$

2.2 Vertex Insertion Approach

In the vertex insertion approach, we identify a subset $V' \subset V$ so the induced subgraph G_P of $V \setminus V'$ is a planar subgraph of G. Starting from a planar drawing Γ_p of G_p, we iteratively insert the vertices in V' at their locally optimal position into Γ_p. Since the respective decision problem of deciding whether there is set V' of at most k vertices is known to be $\mathcal{NP}$-complete [27, 29], we take the following greedy approach:

Let Γ be a non-planar straight-line drawing of G. Let $T' = \langle v_1, v_2, \ldots v_n\rangle$ be an ascending (or descending) order of the vertices of G with respect to their number of crossing cr_x in Γ with

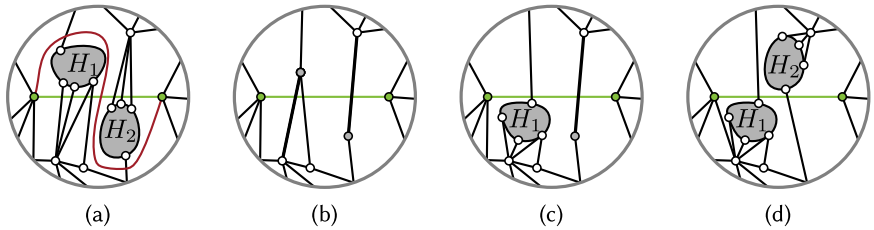


Fig. 2. (a) Crossing minimal curve C_{uv} with dense subgraphs H_1 and H_2. (b) Graph with the contracted subgraphs H_1 and H_2. (c) H_1 unpacked. (d) H_1 and H_2 unpacked.

x = Log, Sum, Sq. Let i be the smallest index such that the sub-drawing Γ_i of Γ induced by the vertices $v_i, \ldots, v_n$ is planar, i.e., the vertices $V' = \{v_j \mid j = 1, 2, \ldots, i-1\}$ are removed from Γ. We obtain a drawing Γ_j from Γ_{j+1} by inserting v_j at its locally optimal position in Γ_{j+1} for $j = 1, \ldots i-1$.

2.3 Edge Insertion Approach

The following heuristic is inspired by the topological edge-insertion algorithm introduced by Gutwenger et al. [22]. We start with a maximal planar subgraph of G and iteratively reinsert edges e into the previous drawing. We modify each drawing so we can add the edge e with a small number of crossings. It is $\mathcal{NP}$-complete to decide whether there is a set E' of k edges such that the graph $G' = (V, E \setminus E')$ is planar [20]. Fortunately, there are exact and heuristic approaches known [12, 24]. For further details, we refer to Section 4.6.

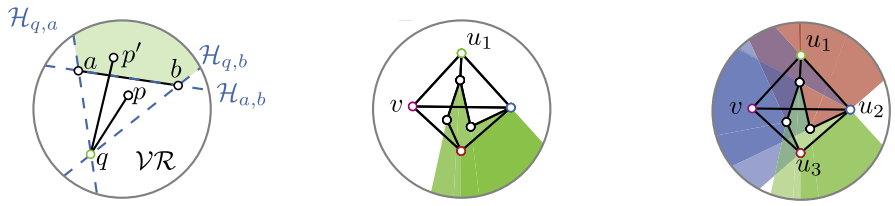
Note that we assume all vertices to be in general position. More formally, let $e = uv$ be an edge of a graph G and Γ_{-e} be a straight-line drawing of $G - e$. We obtain a drawing Γ_{+e} of G by inserting e into Γ_{-e} as a straight-line segment. In the following, we discuss strategies to locally modify the drawing Γ_{+e} to obtain a drawing Γ with a small number of crossings. Let C_{uv} be a *crossing minimal curve* from u to v, i.e., a Jordan arc in Γ_{-e} with u and v as its endpoints, only intersecting edges in its interior and with a minimal number of edge crossings; see Figure 2(a). Ideally, we can rearrange Γ_{-e} such that the edges crossed by e in Γ_{+e} are the same as the edges crossed by C_{uv}. Note that this problem is closely related to the stretchability of pseudolines, which is known to be $\exists\mathbb{R}$-complete [34].

Endpoint. The Endpoint strategy solely moves the endpoints u and v of the inserted edge e in an arbitrary order to their locally optimal position.

Crossed Neighborhood. For a vertex x and an edge e, denote the number of edges xy that cross e in Γ_{+e} by $\mathrm{cr}(\Gamma_{+e}, e, x)$. Let C_e be the set of vertices with $\mathrm{cr}(\Gamma_{+e}, e, x) > 0$. In addition to the endpoints of e, the Crossed Neighborhood strategy moves the vertices in C_e in an order depending on the crossing number cr_a, a = Log, Sum, Sq to their locally optimal position.

Subgraph. Let C_{uv} be a *crossing-minimal curve* from u to v in Γ_{-e} and let E' be the edges crossing C_{uv}. Let R be the (not necessarily simple) region enclosed by e and C_{uv}; see Figure 2. The region R partitions G into a set of subgraphs $H_1, H_2, \ldots, H_k$ of G with drawings $\Gamma_{H_1}, \Gamma_{H_2}, \ldots, \Gamma_{H_k}$ in the interior of R. Let E_j be the set of edges uv with $u \in V \setminus V(H_j)$ and $v \in V(H_j)$.

Let Γ_0 be the drawing obtained from Γ_{+e} by contracting every subgraph H_j to a vertex c_j and placing the vertex in the barycenter of the vertices of H_j. To obtain a drawing Γ_j from Γ_{j-1}, consider a connected region f_j such that moving the vertex c_j within f_j in Γ_{j-1} yields the same number of crossings, i.e., $\mathrm{cr}(\Gamma_{j-1}[c_j \mapsto p]) = \mathrm{cr}(\Gamma_{j-1}[c_j \mapsto p'])$ for every pair of points $p, p' \in f_j$. Let $f_j^\star$ be the region containing the crossing minimal position $p_j^\star$ of the vertex c_j in the drawing Γ_{j-1} (we prove the existence of such a region in Section 3). We obtain a drawing Γ_j by placing a scaled drawing



(a) (b) (c)

Fig. 3. The figures highlight the complements of the visibility regions. (a) The visibility region $\mathcal{VR}(q, s)$ of a point q and a segment $s = \mathcal{S}[a, b]$. (b) All regions $\mathcal{VR}(u_1, e)$ for a neighbor u_1 of v. (c) All regions $\mathcal{VR}(u_j, e)$ for all neighbors u_j of a vertex v.

Γ_{H_j} in the interior of $f_j^\star$ and reinserting the edges E_j and deleting c_j and its edges. This operation can introduce new crossings of the edges E_j with Γ_{H_j}. We resolve these crossings by repositioning every vertex $w \in V(H_j)$ to its locally optimal position with respect to the drawing Γ_j.

3 LOCALLY OPTIMAL VERTEX MOVEMENT

Let Γ be a drawing of a graph G and v be a vertex of G. The algorithms introduced in Section 2 are based on the assumption that we can efficiently compute a position $p^\star$ so the number of crossings in the drawing $\Gamma[v \mapsto p^\star]$ is minimized. In this section, we show that this is possible in $O((kn + m)^2 \log(kn + m))$ time for a degree-k vertex.

In the following, we refer to the edges incident to the vertex v as *active*. The remaining edges are called *inactive*. Let uv be an active edge and let e be an inactive edge. We characterize the set of points p such that moving v to p introduces a crossing between uv and e. Based on the resulting region, we define an arrangement $\mathcal{A}(\Gamma, v)$. Moving the vertex v within a face of this arrangement does not change the number of crossings. Thus, computing an optimal position $p^\star$ reduces to finding a particular face in $\mathcal{A}(\Gamma, v)$.

The mentioned characterization is based on the notion of *visibility*. Let $q \in \mathbb{R}^2$ be the position of u and let $s = \mathcal{S}[a, b] \subset \mathbb{R}^2$ be a closed segment between two points a and b. Let $\mathcal{VR}(q, s) \subset \mathbb{R}^2$ be the *visibility region* of q with respect to s, i.e., the set of points $p \in \mathcal{VR}(q, s)$, so the segments s and $\mathcal{S}[q, p]$ do not intersect. Clearly, $\mathcal{VR}(q, s)$ is the union of three half-planes $\mathcal{H}_{q,a}, \mathcal{H}_{q,b}$, and $\mathcal{H}_{a,b}$ as depicted in Figure 3(a). We denote the boundary of $\mathcal{VR}(q, s)$ by $\mathcal{BD}(q, s)$. Let $\mathcal{A}(\Gamma, v)$ be the arrangement obtained from intersecting the boundaries $\mathcal{BD}(u, e)$ for all pairs of active edges $uv \in E$ and inactive edges e; see Figures 3(b) and 3(c). We show that moving the vertex v within a face of this arrangement does not change the number of crossings in the drawing $\Gamma[v \mapsto p]$. Thus, it is sufficient to compute this arrangement and determine the face $f^\star$ inducing the smallest number of crossings. To avoid special cases, we assume that all vertices are in general position.

Lemma 1. *Let $G = (V, E)$ be a graph with a vertex $v \in V$, and let Γ be a straight-line drawing of G. Let f be a face of $\mathcal{A}(\Gamma, v)$, and let p and p' be two points in the interior of f. Then p and p' have the same crossing number, i.e.,* $\mathrm{cr}(\Gamma[v \mapsto p]) = \mathrm{cr}(\Gamma[v \mapsto p'])$.

Proof. For the sake of a contradiction, assume that there are two distinct points p and p' in the interior of f, so $\mathrm{cr}(\Gamma[v \mapsto p]) < \mathrm{cr}(\Gamma[v \mapsto p'])$. This implies that there is a pair of an active edge e_1 and an inactive edge e_2 that cross in $\Gamma[v \mapsto p']$ but not in $\Gamma[v \mapsto p]$; see Figure 4. Thus, p' is not contained in $\mathcal{VR}(v, e_2)$ but p is. This contradicts the assumption that both p and p' lie in the interior of the same face of $\mathcal{A}(\Gamma, v)$. □

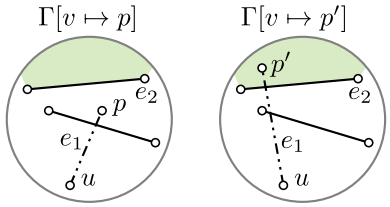


Fig. 4. Moving the vertex v within a face of $\mathcal{A}(\Gamma, v)$ does not change the number of crossings. Illustration for the contradiction.

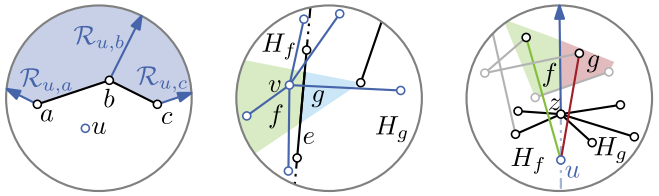


Fig. 5. (a) The boundary $\mathcal{BD}(q, ab)$ is bounded by the two rays $\mathcal{R}_{q,a}, \mathcal{R}_{a,b}$ and the edge ab. The ray $\mathcal{R}_{q,b}$ lies on the boundary of $\mathcal{BD}(q, ab)$ and $\mathcal{BD}(a, bc)$. (b) The faces f and g share a segment incident to an edge e. (c) The faces f and g share a segment on $R_{q,u}$.

Due to Lemma 1, it is sufficient to consider only one point p in the interior of a face f to evaluate the crossing number $\mathrm{cr}(\Gamma[v \mapsto q])$ for an arbitrary point q in f. Thus, in the following, we denote with $\Gamma[v \mapsto f]$ a drawing, where v is moved to an arbitrary point in f.

Theorem 1. *Let Γ be a straight-line drawing of a graph $G = (V, E)$, and let v be a degree-k vertex of G. A point $p^\star \in \mathbb{R}^2$ with the property that $\mathrm{cr}(\Gamma[v \mapsto p^\star]) = \min_{q \in \mathbb{R}^2} \mathrm{cr}(\Gamma[v \mapsto q])$ can be computed in $O((kn+m)^2 \log(kn+m))$ time.*

Proof. The proof relies on the following claims:

Claim 1. *The arrangement $\mathcal{A}(\Gamma, v)$ has $O((kn+m)^2)$ vertices. Moreover, it can be computed in $O((kn+m)^2 \log(kn+m))$ time.*

For each active edge uv, we obtain $O(m)$ visibility regions. The boundary $\mathcal{BD}(q, ab)$ with respect to an edge ab can be represented by two rays $\mathcal{R}_{q,a}$, $\mathcal{R}_{q,b}$ and the edge ab; see Figure 5(a). Observe that the two edges ab, bc share a common ray $\mathcal{R}_{q,b}$. Thus, there are in total $O(kn+m)$ geometric entities ($O(kn)$ rays and $O(m)$ edges) with at most $O((kn+m)^2)$ intersections. Thus, we can compute $\mathcal{A}(\Gamma, v)$ with a sweep-line algorithm [3] in $O((kn+m)^2 \log(kn+m))$ time.

Claim 2. *For all faces f and g of $\mathcal{A}(\Gamma, v)$ that share a segment s the values $\Delta_{f,g}$ such that $\mathrm{cr}(\Gamma[v \mapsto g]) = \mathrm{cr}(\Gamma[v \mapsto f]) + \Delta_{f,g}$ can be computed in $O((kn+m)^2)$ time.*

We distinguish whether the segment s lies on an edge e or on a ray $\mathcal{R}_{u,z}$ for a neighbor u of v and $z \in V$. In both cases, we show that the value $\Delta_{f,g}$ is equal for all pairs of faces f, g that share a segment on e or $\mathcal{R}_{u,z}$, respectively.

First, consider the case that s lies on an edge $e = xy$ in Γ; see Figure 5(b). Denote by H_f and H_g the half-planes of the line that contains s such that H_f contains f and H_g contains g. Let p_f and p_g be points in f and g, respectively, that are sufficiently close to s. Note that, since we assume the vertices to be in general position, there is no vertex $z \neq x, y$ that lies on the line that contains s, Thus, an edge uv and s cross in $\Gamma[v \mapsto p_f]$ if and only if $u \in H_g$. Correspondingly, uv and s cross in

$\Gamma[v \mapsto p_g]$ if and only if $u \in H_f$. Let n_f and n_g be the number of vertices incident to v contained in H_f and H_g, respectively. Hence, we have that $\Gamma[v \mapsto p_g] = \Gamma[v \mapsto p_f] + n_f - n_g$. Due to Lemma 1, it follows that $\Delta_{f,g} = n_f - n_g$. Moreover, the number n_g and n_f are equal for all segments on e, i.e., it is sufficient to compute n_g and n_f with respect to Γ and not for each segment s in $\mathcal{A}(\Gamma, v)$. Overall, the counting requires $O(km)$ time, and mapping these values to differences $\Delta_{f,g}$ requires additional time linear in the size of the arrangement, i.e., $O((kn+m)^2)$ time.

Second, consider the case that s lies on a ray $\mathcal{R}_{u,z}$, i.e., the ray originates in a vertex z and the direction is determined by a neighbor u of v; see Figure 5(c). As before, let H_f and H_g be the half-planes that contain f and g, respectively. Since all vertices lie in general position, we have the following. Each edge wv with $w \neq u$ crosses the same edges in $\Gamma[v \mapsto f]$ as in $\Gamma[v \mapsto g]$. The edges uv and xy cross in $\Gamma[v \mapsto f]$, with $z \neq x, y$, if and only if uv and xy cross in $\Gamma[v \mapsto g]$. Moreover, the edges uv and xz cross in $\Gamma[v \mapsto f]$ if and only if x lies in H_f. Correspondingly, uv and xz cross in $\Gamma[v \mapsto g]$ if and only if x lies in H_g. Let n'_f and n'_f be the number of neighbors of z that lie in H_f and H_g, respectively. Thus, $\Delta_{f,g} = n'_g - n'_f$.

The values n'_f and n'_g can be computed in $O(d_u)$ time, where d_u is the degree of u. Since all differences $\Delta_{f,g}$ are equal for all pair of faces f, g that have a common segment that lies on $\mathcal{R}_{u,z}$, all differences can be computed in $O(\sum_{q \in N_v} \sum_{u \in V} d_u) = O(km)$ time. In time linear in the size of $\mathcal{A}(\Gamma, v)$, these values can be mapped to segments in $\mathcal{A}(\Gamma, v)$. This finishes the proof of the second claim.

In the following, v_f denotes the dual vertex of a face f of $\mathcal{A}(\Gamma, v)$:

Claim 3. *Let s and t be two faces of $\mathcal{A}(\Gamma, v)$, and let s contain v in its interior. Let Π be a simple path from v_s to v_t in the dual graph of $\mathcal{A}(\Gamma, v)$. Then* $\mathrm{cr}(\Gamma[v \mapsto t]) = \mathrm{cr}(\Gamma) + \sum_{(v_f, v_g) \in \Pi} \Delta_{f,g}$.

Since s contains v in its interior, the number of crossings in Γ and $\Gamma[v \mapsto s]$ coincide, i.e., $\mathrm{cr}(\Gamma[v \mapsto s]) = \mathrm{cr}(\Gamma)$. Secondly, for two adjacent faces f and g, we can express the number of crossings in $\Gamma[v \mapsto g]$, depending on the number of crossings in $\Gamma[v \mapsto f]$ and $\Delta_{f,g}$, i.e., $\mathrm{cr}(\Gamma[v \mapsto g]) = \mathrm{cr}(\Gamma[v \mapsto f]) + \Delta_{f,g}$. This proves the claim.

Let f be the face of $\mathcal{A}(\Gamma, v)$ containing v. To find a face $f^\star$ with the minimum number of crossings $\mathrm{cr}(\Gamma[v \mapsto f^\star])$, we determine the number of crossing $\mathrm{cr}(\Gamma[v \mapsto g])$ for every face g in the arrangement $\mathcal{A}(\Gamma, v)$. First, we compute the differences $\Delta_{f,g}$ for all adjacent faces. According to Claim 2, this requires $O((kn+m)^2)$ time.

In time linear in the size of $\mathcal{A}(\Gamma, v)$, the values $\Gamma[v \mapsto g]$ can be accumulated as described in Claim 3 with a breadth-first search in the dual of $\mathcal{A}(\Gamma, v)$ starting at the dual vertex of f. Note that to determine the face $f^\star$, the term $\mathrm{cr}(\Gamma)$ can be omitted from the statement of Claim 3, and thus, does not need to be computed. According to Claim 1, the size of $\mathcal{A}(\Gamma, v)$ is in $O((kn+m)^2)$ and the arrangement can be computed in $O((kn+m)^2 \log(kn+m))$ time. This concludes the proof. □

4 EVALUATION

In the following evaluation, we consider three approaches (i) our geometric heuristic to minimize the number of crossings, (ii) commonly used algorithms to compute straight-line drawings of arbitrary graphs, i.e., energy-based algorithms, and (iii) an approach to minimize the number of crossings in topological drawings.

We use synthetic and real-world instances to evaluate the performance of the algorithms. Section 4.1 contains a brief description of our benchmark instances. The evaluation is based on descriptive statistics and the statistical test described in Section 4.2. Our evaluation is structured as follows. First, we identify a representative for each type of heuristic, i.e., in Section 4.3, we consider

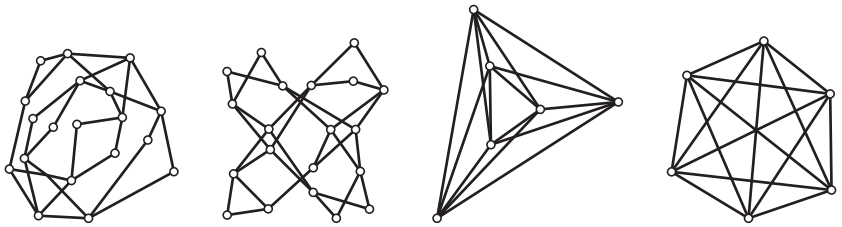


Fig. 6. (a, c) Example drawings computed by our Edge Insertion heuristic with a repositioning with PrEd. (b, d) Drawings computed with Stress. (a, b) North graph 20.47. (c, d) K_6. Number of crossings: (a) 5, (b) 11, (c) 3, (d) 15.

energy-based layouts, in Section 4.4, Section 4.5 and Section 4.6, we consider several configurations of the vertex-movement, vertex-insertion and edge-insertion approach, respectively.

Starting from Section 4.7, we compare the representatives to each other. In particular, in Section 4.7, we focus on the vertex-movement, vertex-insertion and the edge-insertion approach. Section 4.8 compares stress minimization [19], i.e., the representative of the energy-based layouts, to our heuristics. In Section 4.9, we compare our heuristics to a topological crossing minimization approach. We conclude the evaluation with an analysis of the running time in Section 4.10.

The drawings in Figure 6 give a first impression of the effectiveness of our algorithm compared to stress minimization. Figure 6(a) and Figure 6(c) are obtained by one of our heuristics with additional runs of PrEd [4] to optimize the aesthetics of the drawing. The remaining two drawings are computed by stress minimization.

All experiments were conducted on a single core of an Intel Xeon(tm) E5-2670 processor clocked at 2.6GHz. The server is equipped with 64GB RAM. All algorithms were compiled with g++ version 7.3.1 with optimization mode -O3. The operation system was openSUSE Leap 15.0. For geometric operations, we rely on CGAL [36] (v4.10) and GMP[1] to represent coordinates. The usage of CGAL and GMP allows us to evaluate our heuristics without dealing with geometric edge cases. We use snapshot 2017-07-23 of OGDF.

4.1 Benchmark Instances

We evaluated our algorithms on four classes of graphs, either purely synthetic or with a structure resembling real-world data. The classes North and Rome (AT&T)[2] are the non-planar subsets of the corresponding well-known benchmark sets, respectively. The Triangulation+X dataset contains maximal planar graphs with 64 vertices (generated using Reference [9]) and 10 additional random edges. Note that 64 is the maximal number of vertices the generator of Brinkmann et al. can handle. The Community graphs are generated with the LFR-Generator [28] implemented in NetworKit [35]. They resemble social networks with a community structure. From each of these datasets, we selected 100 graphs uniformly at random. Figure 7 shows the size distribution of these graphs.

For each graph G, we generated a *random drawing* on an $m \times m$ integer grid, i.e., the x- and y-coordinates of each vertex is an integer between 0 and m chosen uniformly at random, where m is the number of edges of G. In case the drawing contains three collinear vertices, we assign a

[1] gmplib.org.
[2] http://graphdrawing.org/data.html.

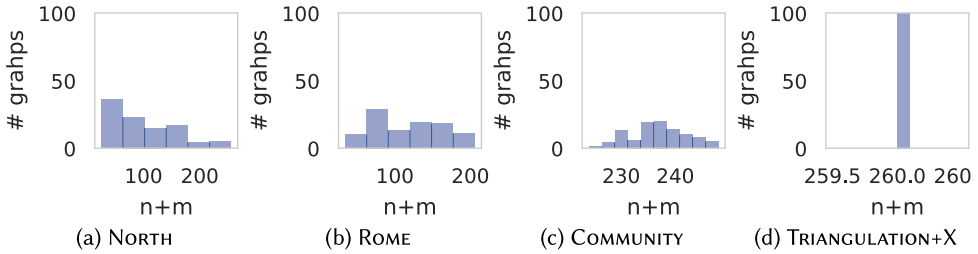


Fig. 7. Distribution of number of vertices plus number of edges for each dataset.

Table 1. Energy-based Graph Drawing Algorithms Implemented in OGDF

Name	OGDF	Ref.
DH	ogdf::DavidsonHarel	[14]
FMMM	ogdf::FMMMLayout	[23]
FR	ogdf::SpringEmbedderFR	[18]
GEM	ogdf::GEMLayout	[17]
KK	ogdf::SpringEmbedderKK	[25]
PMDS	ogdf::PivotMDS	[8]
Stress	ogdf::StressMinimization	[19]

new random position to one of the three vertices. We repeat this process until all vertices are in general position. The resulting drawing is then used as input for all evaluated algorithms.

4.2 Binomial Advantage Test

The *binomial advantage test* is based on a binomial sign test [33]. We follow the terminology of Bläsius et al. [7]. In the following, let $p \in (0, 1]$ and $\alpha \in (0, 1)$ be fixed. A sequence Σ over $\{0, 1\}$ is *good* if the binomial test indicates that the sequence Σ contains more than $p \cdot |\Sigma|$ ones at a significance level α. Note that the binomial test can be used regardless of the distribution of Σ.

We denote by $\Gamma\{G\}$ the set of all drawings of G. Let $\mathcal{G} = \{G_1, G_2, \ldots, G_k\}$ be a family of (non-planar) graphs. We refer to a set $\Lambda = \{\Gamma_1, \ldots, \Gamma_k\}$ with $\Gamma_i \in \Gamma\{G_i\}$ as a *family of drawings* of $\mathcal{G}$. Let Λ^1 and Λ^2 be two families of drawings of $\mathcal{G}$. The comparison $\mathrm{cr}(\Gamma_i^1) \cdot \Delta < \mathrm{cr}(\Gamma_i^2)$ with $1 \leq i \leq k$ yields a sequence Σ_Δ. We say that Λ^1 *has an advantage over* Λ^2 if Σ_1 is good. If Λ^1 has an advantage over Λ^2, then we refer to the maximum value $\Delta^\star$ such that $\Sigma_{\Delta^\star}$ is good, as *advantage of* Λ^1 *over* Λ^2.

4.3 Energy-based Layouts

In this section, we evaluate the energy-based layouts implemented in the Open Graph Drawing Framework (OGDF) (compare Table 1) with respect to the rectilinear crossing number. Some drawings computed by FMMM, KK, and PMDS are *not valid*, i.e., distinct vertices have the same coordinates or a vertex lies in the interior of an edge. We resolve this issue by iteratively perturbing vertices that lie on the interior of an edge.

According to Table 2, drawings computed by DH have a considerably higher number of crossings than drawings computed by GEM, FR, or Stress. The table indicates that FR computes drawings with a slightly higher number of crossings compared to Stress and GEM. A comparison of Stress and GEM is not conclusive, e.g., Stress has larger mean but a smaller median. Observe that FMMM

Table 2. Descriptive Statistics of the Number of Crossings

Algorithm	Mean	Min	.25-Percentile	Median	.75-Percentile	Max
DH	651.59	6	248.50	570.0	993.25	2,210
FMMM	156.58	1	35.00	89.0	289.00	1,369
FR	163.75	2	46.75	109.0	281.25	1,115
GEM	153.43	1	34.25	94.0	259.75	1,174
KK	202.20	1	35.75	86.0	327.00	2,503
PMDS	198.84	1	37.75	99.0	307.25	2,449
Stress	155.78	1	32.75	82.5	288.75	1,220

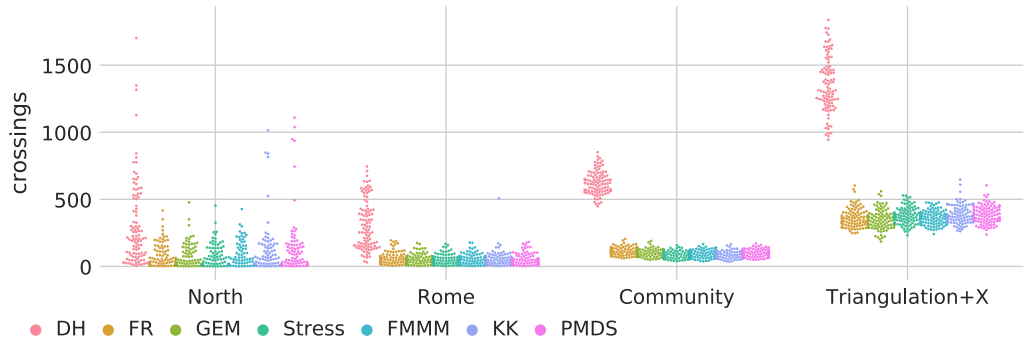


Fig. 8. Comparison of drawings obtained from algorithms implemented in OGDF. The number of crossings of a drawing for each graph in the class indicated on the x-axis clustered by the algorithms. Outliers have been removed.

computes drawings with only a small number of crossings more than Stress. Note that the objective function of DH is explicitly configured to minimize the number of crossings. The remaining algorithms do not have explicit mechanisms to reduce the crossings.

Each point in the plot in Figure 8 corresponds to the number of crossings of one drawing computed by the algorithm indicated by the color. The measurements are categorized by the respective graph class. We removed *outliers* from the plot, i.e., the plot shows all measurements that differ by at most three times the standard deviation from the mean of the respective datasets. The plot confirms our observation that DH computes drawings with the highest number of crossings. For the remaining algorithms, the plot does not show a clear preference. Comparing the graph classes to each other, the plot indicates that the drawings of graphs in the class Triangulation+X computed by energy-based algorithms tend to have a larger number of crossings in comparison to the remaining classes.

The observations drawn from Table 2 and Figure 8 neglect the fact that the algorithms compute drawings of the same graphs, i.e., we are able to directly compare the number of crossings of the drawings. The statistical test introduced in Section 4.2 uses the mapping between the drawings to obtain a statistically reliable comparison of the drawings. Note that the binomial test does not use any assumption about the distribution of the measurements. Figure 9 shows the advantages of the algorithms on the x-axis over the algorithms on the y-axis. For example, Stress has an advantage of 3.5 over DH on 75 out of 100 drawings ($p = 0.75$), i.e., the number of crossings of at least 75% of the drawings computed by DH are larger by a factor of 3.5 than in the corresponding drawings computed by Stress. This and the following advantages are significant at significance

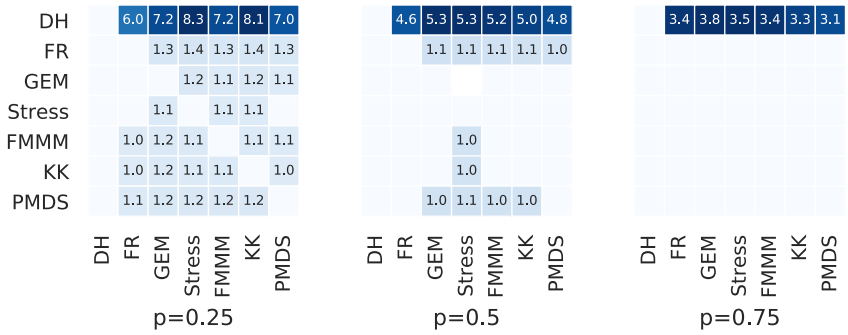


Fig. 9. Advantages of pairs of algorithms.

Table 3. Different Possibilities to Order the Vertices

Name	Counting for $v \in V$	Order
Asc_Log	$\mathrm{cr}_{\mathrm{Log}}(v)$	ascending
Asc_Sum	$\mathrm{cr}_{\mathrm{Sum}}(v)$	ascending
asc_Sq	$\mathrm{cr}_{\mathrm{Sq}}(v)$	ascending
Desc_Log	$\mathrm{cr}_{\mathrm{Log}}(v)$	descending
Desc_Sum	$\mathrm{cr}_{\mathrm{Sum}}(v)$	descending
Desc_Sq	$\mathrm{cr}_{\mathrm{Sq}}(v)$	descending
Rnd		random

level of $\alpha = 0.05$. For $p = 0.5$, we observe that Stress has an advantage over all algorithms except GEM. The advantages over FR, FMMM, and KK lie in between 1.0 and 1.1. We conclude that Stress computes drawings with a slightly but significantly smaller number of crossings in comparison to the other energy-based layouts. Thus, in the following, we use Stress as a representative for the class of energy-based algorithms.

4.4 Vertex Movement

For the vertex movement approach described in Section 2.1, we are free to choose a vertex order. In this section, we evaluate how the choice of the vertex order affects the number of crossings of the final drawings.

In Section 2.1, we introduced three possibilities to count the number of crossings for a vertex v of G. Moreover, we can decide to order the vertices in ascending or in descending order. Table 3 lists all configurations of the vertex movement approach that we evaluate. It contains additionally a random permutation of the vertex set.

As in the evaluation of the energy-based layouts, we use the descriptive statistics in Table 4, the plot in Figure 10, and the advantages (Figure 11) to compare the configurations of the vertex-movement approach to each other. The statistics in Table 4 and the plot in Figure 10 indicate that configurations that move the vertices in ascending order (Asc_⋆) compute drawings with a considerably higher number of crossings compared to configurations using a descending order (Desc_⋆) or the random order (Rnd).

The plots of the advantages in Figure 11 confirm this observation for $p = 0.75$. Moreover, for $p = 0.5$, Desc_Sq has a small advantage over Desc_Log, i.e., the advantage is between 1.0 and 1.1. For $p = 0.25$, each configuration using a descending order (Desc_⋆) has an advantage of 1.1

Table 4. Descriptive Statistics of the Number of Crossings Obtained by the *Vertex-movement* Approach with Different Vertex Orders

Algorithm	Mean	Min	.25-Percentile	Median	.75-Percentile	Max
Asc_Log	282.35	1	44.00	214.5	495.50	1147
Asc_Sum	303.83	1	47.00	233.0	504.50	1303
Asc_Sq	310.28	1	46.75	246.5	510.25	1184
Desc_Log	182.28	1	32.75	167.5	267.25	1074
Desc_Sum	176.24	1	29.00	157.5	258.75	970
Desc_Sq	174.46	1	30.75	157.0	266.00	910
Rnd	238.54	1	35.75	185.5	387.00	1070

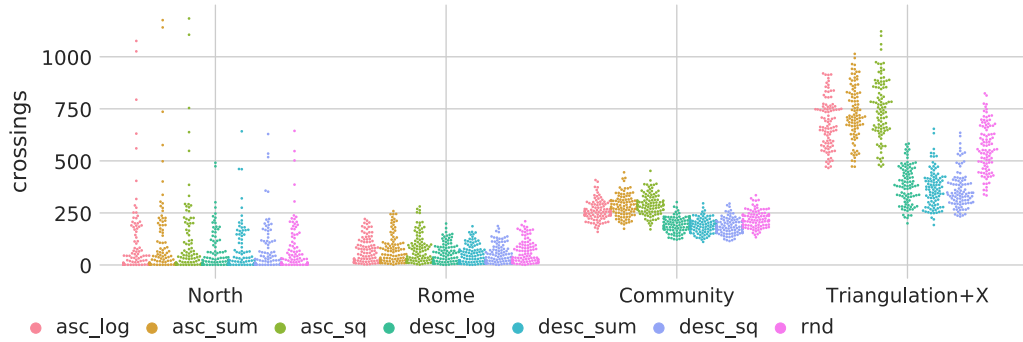


Fig. 10. Comparison of drawings obtained from different configurations of the vertex movement approach. The number of crossings of a drawing for each graph in the class indicated on the x-axis clustered by the configurations. Outliers have been removed.

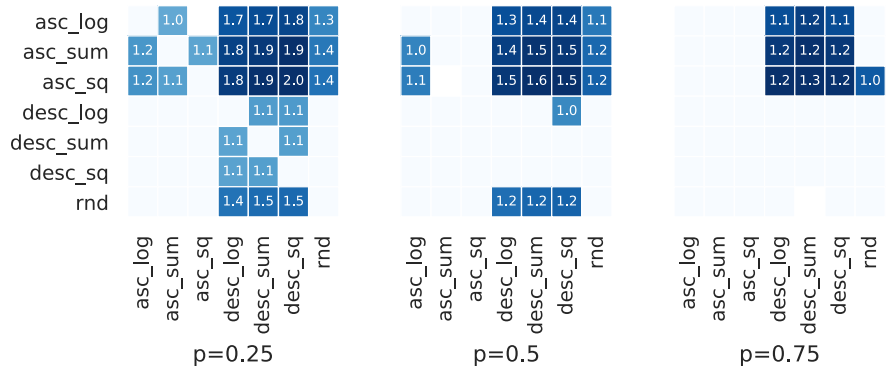


Fig. 11. Advantages of pairs of configurations of the vertex movement approach.

over the other configurations using a descending order. Thus, the advantages do not show a clear preference to one of these configurations.

To reduce the complexity of the rest of the evaluation, we choose a single configuration of the vertex movement approach. Therefore, we consider the average number of crossings as a tie breaker. Since the Desc_Sq computes the smallest number of crossings with respect to this statistic, we use this configuration as the representative for the vertex-movement approach.

Table 5. Descriptive Statistics of the Number of Crossings Obtained by the *Vertex-insertion* Approach with Different Vertex Orders

Algorithm	Mean	Min	.25-Percentile	Median	.75-Percentile	Max
Asc_Log	126.18	1	37.00	138.5	185.25	1,217
Asc_Sum	131.16	1	34.00	142.0	188.00	1,187
Asc_Sq	140.57	1	37.00	155.5	204.00	1,316
Desc_Log	406.32	1	76.75	310.0	778.50	1,905
Desc_Sum	386.56	1	72.75	276.5	739.75	1,712
Desc_Sq	360.68	1	62.00	252.0	680.25	1,652
Rnd	237.01	1	55.50	227.5	369.25	1,080

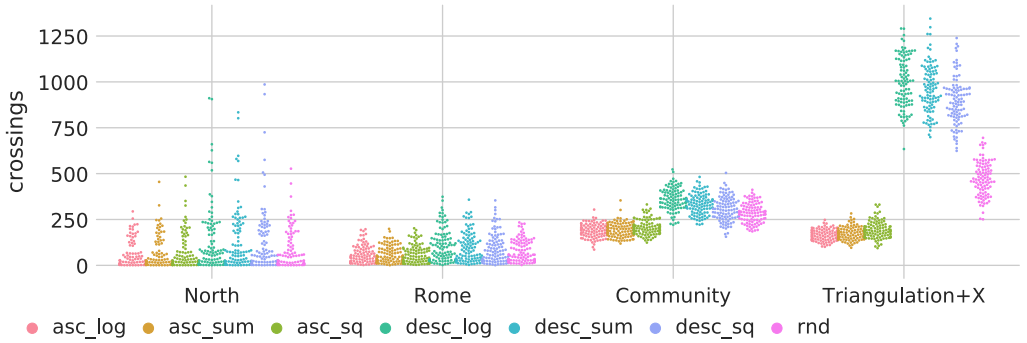


Fig. 12. Comparison of drawings obtained from different configurations of the vertex insertion approach. The number of crossings of a drawing for each graph in the class indicated on the x-axis clustered by the configurations. Outliers have been removed.

4.5 Vertex Insertion

Similar to the vertex-movement approach, the order in which we remove and insert vertices in the vertex-insertion approach (Section 2.2) can affect the number of crossings of the final drawing. In this section, we evaluate the vertex-insertion approach with different vertex orders (see Table 3). Note that in case of an ascending order (Asc_$\star$), the vertices are removed in this order and inserted in the reversed (descending) order. Vice versa, in a descending order (Desc_$\star$), the vertices are removed in descending order and reinserted in ascending order. Preliminary experiments indicated that reinserting the vertices in the same order instead of the reversed order yields a larger number of crossings. To reduce the complexity of the evaluation, we decided to omit these configurations.

The descriptive statistics in Table 5 and the plot in Figure 12 show that the descending vertex orders yield drawings with a considerably higher number of crossings compared to the ascending orders. The statistics indicate that the vertex insertion approach with the Asc_Log order computes drawings with the smallest number of crossings. The advantages in Figure 13 confirm this observation. For $p = 0.75$, Asc_Log has higher advantages as the corresponding advantages of Asc_Sum and Asc_Sq. Moreover, for $p = 0.25$, the Asc_Log order has an advantage over both Asc_Sum and Asc_Sq with an advantage of 1.2. However, Asc_Sum and Asc_Sq each have only an advantage of 1.1 over Asc_Log. Hence, for the following evaluations, we consider the vertex-insertion approach with the Asc_Log order.

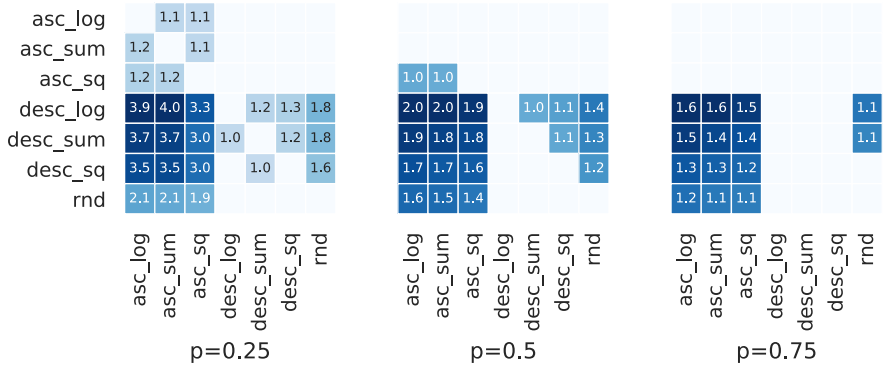


Fig. 13. Advantages of pairs of configurations of the vertex insertion approach.

Table 6. Configuration of the Reference Algorithms

	Algorithm	Vertex Order	Termination
VM	Vertex Movement	Desc_Sq	
VI	Vertex Insertion	Asc_Log	
EP	Edge Insertion	Endpoints	
EI	Edge Insertion	EP + Crossed Neighbors (Desc_Sq)	
Stress	ogdf::StressMinimization		convergence
Tpl	ogdf::SubgraphPlanarizer	-	-

4.6 Edge Insertion

The edge insertion approach as described in Section 2.3 has several degrees of freedom: (i) the computation of the maximal planar subgraph, (ii) the initial drawing of the maximal planar subgraph, (iii) the order in which the edges are reinserted, and (iv) the order in which the vertices are moved after each edge insertion. We use the PlanarSubgraphFast algorithm [24] implemented in OGDF to compute a large planar subgraph and the PlanarStraightLayout method to compute an initial drawing of the planar subgraph. In both cases, we use the default configuration of OGDF. We reinsert the edges in the order they are returned by the PlanarSubgraphFast routine. The configuration EP only moves the endpoints of an edge. The configuration EI moves in addition to the endpoints the crossed neighborhood of the newly inserted edge e, i.e., vertices that are incident to an edge that crosses e. In Section 4.4, we selected the Desc_Sq order as a representative for the vertex-movement approach. Hence, we use this vertex order to move the crossed neighborhood of an edge.

Since EI moves a superset of the vertices of EP, we expect that EI further reduces the number of crossings, compared to EP. Table 7 and the plots in Figures 14 and 15 confirm this observation.

In comparison to the conference version of the article [31], we reimplemented the geometric operation of moving a single vertex and the heuristics (VM, VI, EI, EP). In the experiments on the old code base, we observed that the edge insertion heuristic with the additional movements of subgraphs introduced a significant number of new crossings. Since moving entire subgraphs did not seem promising, we decided to not reimplement this particular heuristic.

4.7 Comparison of Our Heuristics

In the following, we compare our heuristics, i.e., VM, VI, EP, and EI, to each other (Table 6). For the comparison of the heuristics to Stress and Tpl,refer to Section 4.8 and Section 4.9,

Table 7. Descriptive Statistics of the Number of Crossings of Drawings Computed by the Final Configuration of the Heuristics, Stress, and Tpl

Algorithm	Mean	Min	.25-Percentile	Median	.75-Percentile	Max
Tpl	43.30	1	7.00	29.0	66.25	610
EI	55.43	1	9.00	41.0	87.25	601
EP	69.41	1	9.00	49.0	107.75	630
VI	126.18	1	37.00	138.5	185.25	1,217
VM	174.46	1	30.75	157.0	266.00	910
Stress	155.51	1	32.75	82.5	288.75	1,220

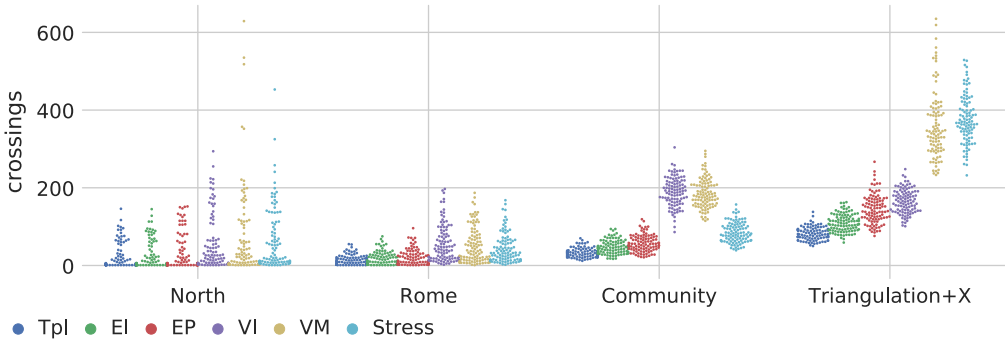


Fig. 14. Comparison of the final configurations of each heuristic, Stress, and Tpl. The number of crossings of a drawing for each graph in the class indicated on the x-axis clustered by the heuristic. Outliers have been removed.

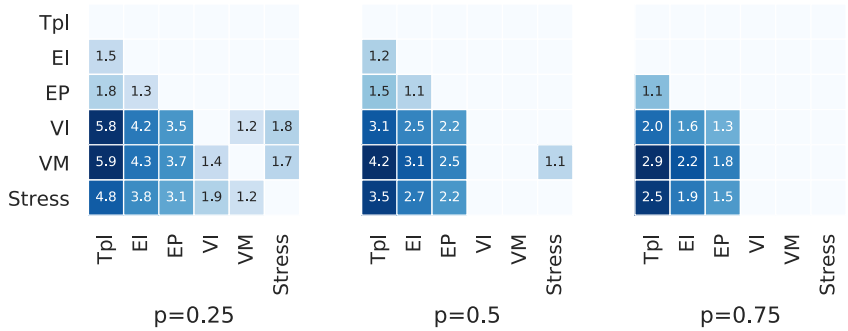


Fig. 15. Advantages of pairs of the final configurations of the heuristics, Stress, and Tpl.

respectively. Table 7 suggests that EI computes drawings with fewer crossings than EP, EP fewer than VI and VM. Recall that a point in Figure 14 corresponds to the number of crossings of one drawing computed by the algorithm indicated by the color. The plot confirms that the edge-insertion approaches compute drawings with less crossings than VI and VM. Moreover, VI computes drawings of the Triangulation+X graphs with less crossings than VM.

For $p = 0.75$, we observe that EI and EP compute drawings with significantly less crossings than VI and VM; refer to Figure 15. Thus, we can confirm the above observation at a significance level of $\alpha = 0.05$. Considering the graph classes independently, the statement remains true for the North, Rome, and Community graphs; see Figures 16, 17, and 18. For the Triangulation+X graphs, only

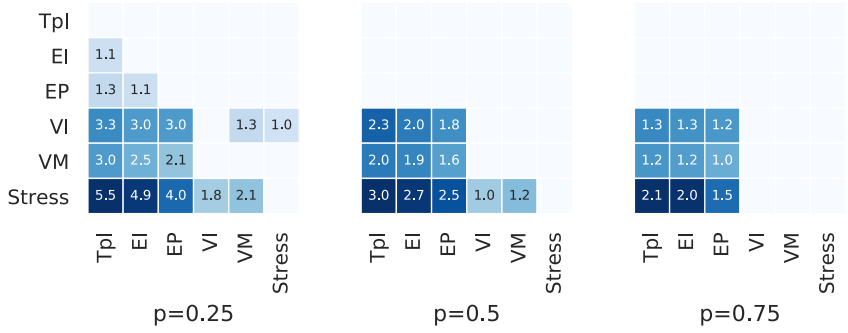


Fig. 16. North.

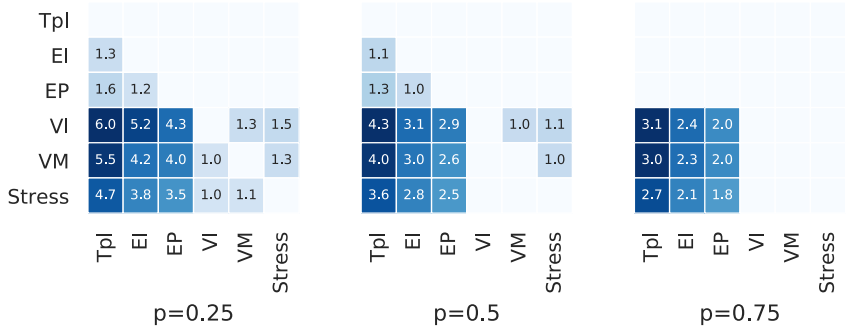


Fig. 17. Rome.

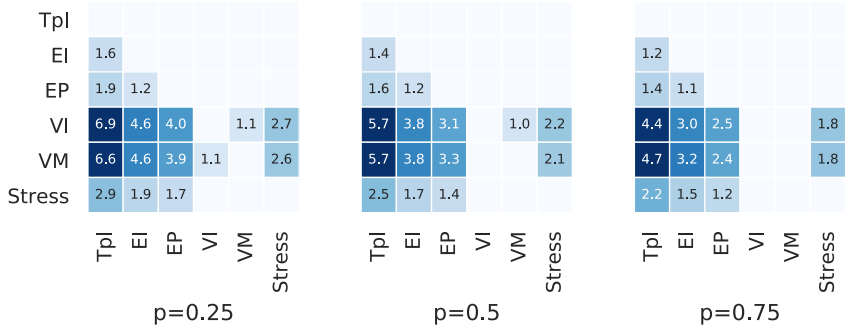


Fig. 18. Community.

EI has an advantage over both VI and VM, for $p = 0.75$ (Figure 19). For $p = 0.5$, EP has an advantage of 2.3 and 1.1 over VM and VI, respectively.

Only for Triangulation+X graphs and the Community graphs, EI has an advantage of 1.1 over EP on at least 75% of the graphs. For $p = 0.5$, EI has an advantage between 1.0 and 1.1, on the Rome graphs; for $p = 0.25$, this advantage increases to 1.2. For the North graphs and $p = 0.25$, EI has an advantage of 1.1 over EP.

Comparing VI and VM, neither heuristic has an advantage over the other for $p = 0.5$ (Figure 15). For $p = 0.25$, VI has an advantage of 1.4 over VM, and VM has an advantage of 1.1 over VI. Considering the graph classes independently, we see that on the North, Rome, and Community graphs,

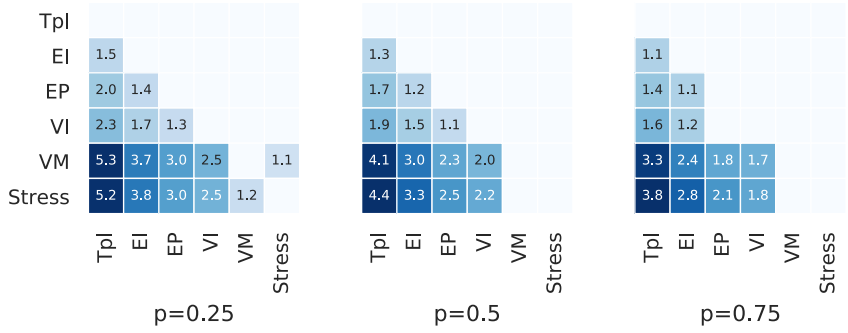


Fig. 19. Triangulation+X.

VM has a small advantage over VI. On the Triangulation+X graphs, VI computes drawings with significantly less crossings than VM, i.e., for $p = 0.75$, VI has an advantage of 1.7 over VM.

Overall, we conclude that the edge-insertion approach (EI and EP) computes drawings with significantly fewer crossings than its competitors. It depends on the graph class whether the additional movement of vertices (EI) significantly decreases the number of crossings compared to EP.

4.8 Comparison to Stress

We compare the drawings computed by our heuristics with drawings computed by stress minimization (Stress), i.e., to an algorithm commonly used to compute straight-line drawings of general graphs. In Section 4.3, we showed that this algorithm computes drawings with less crossings than other energy-based heuristics implemented in OGDF. We configured Stress to stop after convergence, thus we cannot expect Stress to compute drawings with a smaller number of crossings if we increase the computing time.

Table 7 suggests that Stress computes drawings with at least a factor of two more crossings than EI and EP. A comparison between Stress and VI is inconclusive. On average, VI computes drawings with a smaller number of crossings; however, Stress has a smaller median value.

In addition to the above observations, Figure 14 shows that on a large subset of the Triangulation+X graphs, Stress computes drawings with a considerably larger amount of crossings than EI, EP, and VI. On the Community graphs, Stress achieves a smaller number of crossings than VI and VM. For the remaining graph classes, the plot provides no clear distinction between VI, VM, and Stress. Although Table 7 and Figure 14 do not provide a conclusive distinction between Stress and VM, Figure 15 shows an advantage of 1.1 for Stress over VM, for $p = 0.5$.

The advantages in Figure 15 show that Stress computes drawings with a factor of 1.9 and 1.5 more crossings than EI and EP, respectively, for 75% of our benchmark instances. Further, considering only the Community graphs (Figure 18), the advantage of EI over Stress drops from 1.9 to 1.5, for $p = 0.75$. However, for the Triangulation+X graphs, the advantage increases to 2.8. We conclude that the edge-insertion approach computes drawings with significantly less crossings than Stress.

4.9 Comparison to Tpl

We investigate how close the number of crossings in drawings computed by EI are to the number of crossings in topological drawings. Note that Tpl as well as EI start from a large planar subgraph and iteratively insert the remaining edges. The drawings obtained by Tpl are not necessarily realizable as straight-line drawings with the same number of crossings.

Table 8. Descriptive Statistics of the Running Time in Seconds Per Graph Class

Algorithm	Mean	Min	.25-Percentile	Median	.75-Percentile	Max
North						
Tpl	0.90	<0.01	<0.01	0.01	0.16	42.07
EI	35.77	0.04	0.46	2.61	30.64	934.79
EP	4.37	0.01	0.10	0.57	4.60	95.53
VI	1.49	0.02	0.17	0.69	2.09	9.23
VM	3.86	0.04	0.32	1.48	5.87	29.60
Rome						
Tpl	0.06	<0.01	0.01	0.03	0.08	0.55
EI	10.86	0.20	1.14	4.99	15.81	61.11
EP	2.12	0.07	0.32	1.11	2.94	10.67
VI	5.62	0.09	0.66	1.85	3.24	178.40
VM	4.20	0.27	1.12	2.70	6.19	15.31
Community						
Tpl	0.24	0.04	0.13	0.20	0.29	0.90
EI	50.81	24.66	38.80	50.41	61.11	88.73
EP	10.28	6.58	8.26	10.43	11.64	20.29
VI	27.40	7.05	8.60	10.25	17.53	514.47
VM	21.24	17.34	20.10	21.27	22.26	25.61
Triangulation+X						
Tpl	0.67	0.10	0.40	0.57	0.83	2.79
EI	391.40	200.23	348.75	393.31	428.81	566.14
EP	52.20	22.23	45.28	50.56	60.47	89.74
VI	5.56	4.98	5.39	5.53	5.71	6.52
VM	34.17	31.28	33.06	34.06	34.99	50.46

Table 7 shows that the maximum number of crossings computed by EI is smaller than the corresponding number computed by Tpl. The Tpl approach iteratively inserts edges into a planar graph. After each edge insertion, the crossings are replaced by degree-four vertices. This fixes the crossings for future edge insertions. Our edge insertion approach (EI) at least moves the vertices v incident to a new edge e. Since the vertex movement minimizes the number of crossings of all edges incident to v, it is possible that two edges that cross in Γ do not cross in Γ_{+e}. Apparently, this flexibility helps in some cases to find drawings with less crossings compared to Tpl. Indeed, there are 60 out of 400 instances in which the number of crossings computed by EI is smaller or equal to the number of crossings computed by Tpl. On 35 instances, EI achieves a strictly smaller number of crossings than Tpl.

For at least 75% of graphs, Tpl has an advantage of 1.1 over EP; see Figure 15. For the same number of graphs, Tpl does not have an advantage over EI. However, there is a subset containing at least 25% of graphs such that Tpl has an advantage of 1.5 over EI and 1.8 over EP. Considering the Community and the Triangulation+X graphs, Tpl has an advantage over all other algorithms for $p = 0.75$, but the advantage over EI is only 1.2 and 1.1, respectively.

4.10 Running Time

In this section, we analyze the running time of our algorithms. We abstain from a comparison to Stress, since Stress is very well engineered and requires at most 10^{-2}s per instance on our graphs.

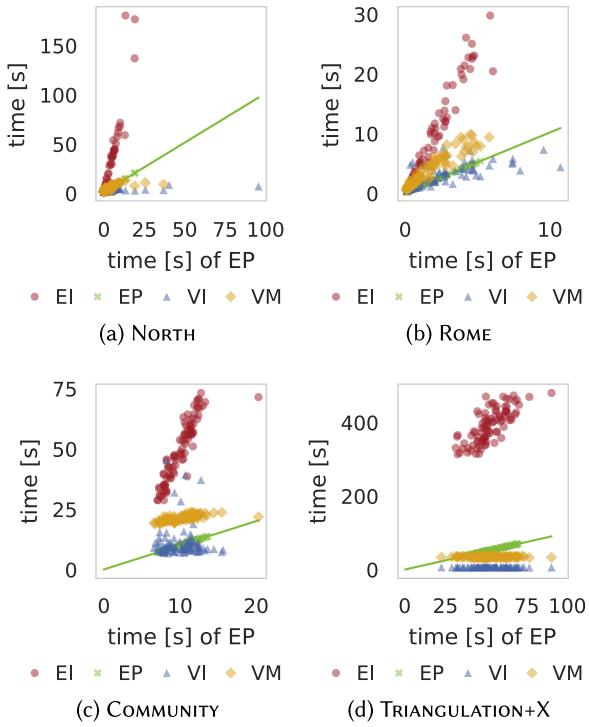


Fig. 20. The running time of each algorithm as a function of the running time of EP, i.e., each data point (t_x, t_y) corresponds to graph G and an algorithm $\mathcal{A}$, where t_x and t_y is the running time of EP and $\mathcal{A}$ on G, respectively. We removed outliers to increase readability.

We compare the remaining algorithms listed in Table 6. Table 8 shows several statistics of the running time grouped by graph class. Figure 20 shows the running time of each instance for all graph classes. Since the running time of Tpl is less than one second for most instances (compare Table 8), we omit these measurements in Figure 20 to increase readability. A data point p_G below the green diagonal indicates that the algorithm that corresponds to p_G uses less time to finish on the graph than EP. For example, Figure 20(d) shows that there are many instances where VM and VI consume less time than EP. However, on every Triangulation+X instance, the running time of EI is considerably higher. Note that on the Triangulation+X graphs, EI only has a small advantage over VI; compare Figure 19. However, VI is significantly faster on this graph class.

The observation that EI has the longest running time is true for all graph classes. Recall that EI moves a superset of vertices compared to EP. Thus, this observation is expected. Moreover, the figures show that the edge-insertion approach that only moves endpoints of an edge (EP) and VI profit from the incremental growth of the drawing, whereas the vertex-movement approach has to deal with the entire graph in each iteration.

5 CONCLUSION

In this article, we introduced several heuristics that are based on moving a vertex to its crossing minimal position. This position can be computed in $O((kn+m)^2 \log(kn+m))$ time. Our evaluation in Section 4 shows that the approach yields drawings with a smaller number of crossings in comparison to the well-established stress-minimization algorithm.

The edge-insertion approach in combination with the crossed-neighborhood strategy computes drawings with the smallest number of crossings. We compared our heuristic to an approach computing topological drawings with a small number of crossings. Our experimental evaluation showed that there is only a relatively small difference between the number of crossings. Especially, we could show that we are able to match the number of crossings in about 15% of our instances. For future work, it is desirable to further engineer the implementation to cope with larger instances.

REFERENCES

[1] Bernardo M. Ábrego, Silvia Fernández-Merchant, and Gelasio Salazar. 2013. The rectilinear crossing number of K_n: Closing in (or are we?). In *Thirty Essays on Geometric Graph Theory*, János Pach (Ed.). Springer New York, 5–18. DOI: https://doi.org/10.1007/978-1-4614-0110-0_2

[2] Oswin Aichholzer. 2017. On the Rectilinear Crossing Number. Retrieved from: http://www.ist.tugraz.at/staff/aichholzer/research/rp/triangulations/crossing.

[3] John L. Bentley and Thomas A. Ottmann. 1979. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.* C-28, 9 (1979), 643–647.

[4] François Bertault. 2000. A force-directed algorithm that preserves edge-crossing properties. *Inform. Process. Lett.* 74, 1–2 (2000), 7–13.

[5] Daniel Bienstock. 1991. Some provably hard crossing number problems. *Disc. & Comput. Geo.* 6, 1 (1991), 443–459. DOI: https://doi.org/10.1007/BF02574701

[6] Daniel Bienstock and Nathaniel Dean. 1993. Bounds for rectilinear crossing numbers. *J. Graph Theor.* 17, 3 (1993), 333–348. DOI: https://doi.org/10.1002/jgt.3190170308

[7] Thomas Bläsius, Marcel Radermacher, and Ignaz Rutter. 2017. How to draw a planarization. In *Proceedings of the 43rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'17)*, Bernhard Steffen, Christel Baier, Mark van den Brand, Johann Eder, Mike Hinchey, and Tiziana Margaria (Eds.). Springer International Publishing, 295–308. DOI: https://doi.org/10.1007/978-3-319-51963-0_23

[8] Ulrik Brandes and Christian Pich. 2007. Eigensolver methods for progressive multidimensional scaling of large data. In *Proceedings of the 14th International Symposium on Graph Drawing (GD'06)*, Michael Kaufmann and Dorothea Wagner (Eds.). Springer, Berlin, 42–53. DOI: https://doi.org/10.1007/978-3-540-70904-6_6

[9] Gunnar Brinkmann and Brendan D. McKay. 2007. Fast generation of planar graphs. *MATCH-Commun. Math. Comput. Chem.* 58, 2 (2007), 323–357.

[10] Christoph Buchheim, Markus Chimani, Carsten Gutwenger, Michael Jünger, and Petra Mutzel. 2013. Crossings and planarization. In *Handbook of Graph Drawing and Visualization*, Roberto Tamassia (Ed.). Chapman and Hall/CRC, Chapter 2, 43–85.

[11] Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Karsten Klein, and Petra Mutzel. 2013. The open graph drawing framework (OGDF). In *Handbook of Graph Drawing and Visualization*, Roberto Tamassia (Ed.). Chapman and Hall/CRC, Chapter 17, 543–569.

[12] Markus Chimani, Ivo Hedtke, and Tilo Wiedera. 2018. Exact algorithms for the maximum planar subgraph problem: New models and experiments. In *Proceedings of the 17th International Symposium on Experimental Algorithms (SEA'18) (Leibniz International Proceedings in Informatics (LIPIcs))*, Gianlorenzo D'Angelo (Ed.), Vol. 103. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 22:1–22:15. DOI: https://doi.org/10.4230/LIPIcs.SEA.2018.22

[13] Markus Chimani and Petr Hlinený. 2016. Inserting multiple edges into a planar graph. In *Proceedings of the 32nd Symposium on Computational Geometry (SoCG'16)* (Leibniz International Proceedings in Informatics (LIPIcs)), Sándor Fekete and Anna Lubiw (Eds.), Vol. 51. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 30:1–30:15. DOI: https://doi.org/10.4230/LIPIcs.SoCG.2016.30

[14] Ron Davidson and David Harel. 1996. Drawing graphs nicely using simulated annealing. *ACM Trans. Graph.* 15, 4 (1996), 301–331. DOI: https://doi.org/10.1145/234535.234538

[15] Ruy Fabila-Monroy and Jorge López. 2014. Computational search of small point sets with small rectilinear crossing number. *J. Graph Algor. Appl.* 18, 3 (2014), 393–399. DOI: https://doi.org/10.7155/jgaa.00328

[16] Jacob Fox, János Pach, and Andrew Suk. 2016. Approximating the rectilinear crossing number. In *Proceedings of the 24th International Symposium on Graph Drawing (GD'16)*, Yifan Hu and Martin Nöllenburg (Eds.). Springer, Berlin, 413–426. DOI: https://doi.org/10.1007/978-3-319-50106-2_32

[17] Arne Frick, Andreas Ludwig, and Heiko Mehldau. 1995. A fast adaptive layout algorithm for undirected graphs (extended abstract and system demonstration). In *Proceedings of the 2nd International Symposium on Graph Drawing (GD'94)*, Roberto Tamassia and Ioannis G. Tollis (Eds.). Springer, Berlin, 388–403. DOI: https://doi.org/10.1007/3-540-58950-3_393

[18] Thomas M. J. Fruchterman and Edward M. Reingold. 1991. Graph drawing by force-directed placement. *Softw.: Pract. Exper.* 21, 11 (1991), 1129–1164. DOI: https://doi.org/10.1002/spe.4380211102

[19] Emden R. Gansner, Yehuda Koren, and Stephen North. 2005. Graph drawing by stress majorization. In *Proceedings of the 12th International Symposium on Graph Drawing (GD'04)*, János Pach (Ed.). Springer, Berlin, 239–250. DOI: https://doi.org/10.1007/978-3-540-31843-9_25

[20] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman.

[21] Michael R. Garey and David S. Johnson. 1983. Crossing number is NP-complete. *SIAM J. Alg. Disc. Meth.* 4, 3 (1983), 312–316.

[22] Carsten Gutwenger, Petra Mutzel, and René Weiskircher. 2005. Inserting an edge into a planar graph. *Algorithmica* 41, 4 (2005), 289–308. DOI: https://doi.org/10.1007/s00453-004-1128-8

[23] Stefan Hachul and Michael Jünger. 2005. Drawing large graphs with a potential-field-based multilevel algorithm. In *Proceedings of the 12th International Symposium on Graph Drawing (GD'04)*, János Pach (Ed.). Springer, Berlin, 285–295. DOI: https://doi.org/10.1007/978-3-540-31843-9_29

[24] Michael Jünger, Sebastian Leipert, and Petra Mutzel. 1998. A note on computing a maximal planar subgraph using PQ-trees. *IEEE Trans. Comput.-Aided Des. Integ. Circ. Syst.* 17, 7 (1998), 609–612. DOI: https://doi.org/10.1109/43.709399

[25] Tomihisa Kamada and Satoru Kawai. 1989. An algorithm for drawing general undirected graphs. *Inform. Process. Lett.* 31, 1 (1989), 7–15. DOI: https://doi.org/10.1016/0020-0190(89)90102-6

[26] Stephen G. Kobourov. 2013. Force-directed drawing algorithms. In *Handbook of Graph Drawing and Visualization*, Roberto Tamassia (Ed.). Chapman and Hall/CRC, Chapter 12.

[27] M. S. Krishnamoorthy and Narsingh Deo. 1979. Node-deletion NP-complete problems. *SIAM J. Comput.* 8, 4 (1979), 619–625. DOI: https://doi.org/10.1137/0208049

[28] Andrea Lancichinetti, Santo Fortunato, and Filippo Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* 78, 4 (2008), 046110.

[29] John M. Lewis and Mihalis Yannakakis. 1980. The node-deletion problem for hereditary properties is NP-complete. *J. Comput. Syst. Sci.* 20, 2 (1980), 219–230. DOI: https://doi.org/10.1016/0022-0000(80)90060-4

[30] Helen C. Purchase, Robert F. Cohen, and Murray James. 1996. Validating graph drawing aesthetics. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*, Franz J. Brandenburg (Ed.). Springer, Berlin, 435–446. DOI: https://doi.org/10.1007/BFb0021827

[31] Marcel Radermacher, Klara Reichard, Ignaz Rutter, and Dorothea Wagner. 2018. A geometric heuristic for rectilinear crossing minimization. In *Proceedings of the 20th Workshop on Algorithm Engineering and Experiments (ALENEX'18).* 129–138. DOI: https://doi.org/10.1137/1.9781611975055.12

[32] Marcus Schaefer. 2010. Complexity of some geometric and topological problems. In *Proceedings of the 17th International Symposium on Graph Drawing (GD'09) (Lecture Notes in Computer Science)*, David Eppstein and Emden R. Gansner (Eds.), Vol. 5849. Springer, Berlin, 334–344. DOI: https://doi.org/10.1007/978-3-642-11805-0_32

[33] David J. Sheskin. 2003. *Handbook of Parametric and Nonparametric Statistical Procedures.* Chapman and Hall/CRC.

[34] Peter W. Shor. 1991. Stretchability of pseudolines is NP-hard. In *Applied Geometry and Discrete Mathematics—The Victor Klee Festschrift*, Bernd Sturmfels and Peter Gritzmann (Eds.). Vol. 4. AMS, DIMACS, and ACM, 531–554.

[35] Christian L. Staudt, Aleksejs Sazonovs, and Henning Meyerhenke. 2016. NetworKit: A tool suite for large-scale complex network analysis. *Netw. Sci.* 4, 4 (2016), 508–530. DOI: https://doi.org/10.1017/nws.2016.20

[36] The CGAL Project. 2017. CGAL User and Reference Manual. CGAL editorial board. Retrieved from: http://doc.cgal.org/4.10/Manual/packages.html.

[37] Imrich Vrt'o. 2014. Bibliography on Crossing Numbers of Graphs. Retrieved from: ftp://ftp.ifi.savba.sk/pub/imrich/crobib.pdf.

Received August 2018; revised January 2019; accepted April 2019