A Geometric Heuristic for Rectilinear Crossing Minimization *

Marcel Radermacher†, Klara Reichard‡, Ignaz Rutter§, Dorothea Wagner†

Abstract

In this paper 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 [1,9]. 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 $O\left((kn+m)^2 \log(kn+m)\right)$-time algorithm, where k is the degree of the vertex and n and m are the numbers 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 [10] on a varied set of benchmark instances. All experiments are evaluated with a statistical significance level of $\alpha = 0.05$.

1 Introduction

The empirical study of Purchase et al. [24] 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 theory and in practice; the bibliography of Vrt'o is an impressive list of over 700 references [29]. A *topological drawing* of a graph is a drawing where each edge is a Jordan Curve 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 [17]. 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 [9, 11]. Unfortunately, deciding whether there is a straight-line drawing homeomorphic to a given drawing is $\exists\mathbb{R}$-complete [25, 27]. Based on the topological drawings with a small number of crossings, Bläsius et al. [7] heuristically straighten the edges. Overall it is in general 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, 9] show that the estimation of the (rectilinear) crossing number of complete graphs has received considerable attention. Most recently Fox et al. [14] 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 [13].

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 [20]. 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 [12] actively reduces the number of crossings among other optimization criteria. Apart from that we

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

†Department of Computer Science, Karlsruhe Institute of Technology ,Germany, {radermacher, dorothea.wagner}@kit.edu

‡Department of Algorithms in Bioinformatics, Universität Tübingen, Tübingen, Germany, klara.reichard@uni-tuebingen.de

§Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands, i.rutter@tue.nl

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 a undirected graph with vertex set V and edge set E and let Γ be a straight-line drawing of G. For a vertex $v \in V$ we denote by $\Gamma[v \mapsto p], p \in \mathbb{R}^2$, 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 in order 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 $O\left((kn+m)^2 \log(kn+m)\right)$ 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* [10] with a statistical significance of $\alpha = 0.05$. Throughout the remainder of this paper, 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 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 position $p^\star$ so that 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.

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.

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. Fig. 1 illustrates that these can yield different orderings.

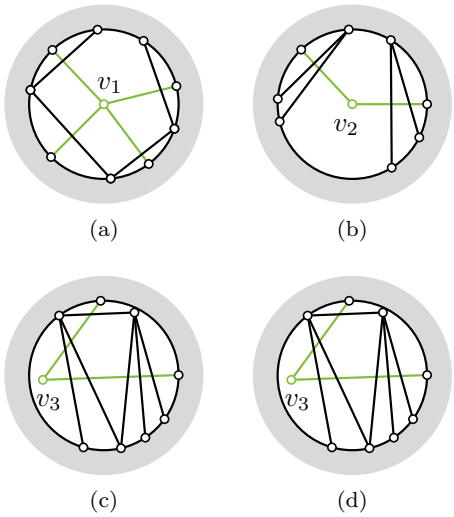


Figure 1: Assume that structures in (a)-(d) are substructures of a common drawing Γ. Depending on the function $\mathrm{cr}_i(\Gamma, \cdot)$ the vertices are moved in a different order. For $i = 1$ (a) is the first in order and (c) last. For $i = 2$ the (b) is first and (c) last. For $i = 3$, (a) is first and (d) last.

$$\mathrm{cr}_1(\Gamma, v) = \sum_{e \in E(v)} \log(\mathrm{cr}(\Gamma, e) + 1) \tag{2.1}$$

$$\mathrm{cr}_2(\Gamma, v) = \sum_{e \in E(v)} \mathrm{cr}(\Gamma, e) \tag{2.2}$$

$$\mathrm{cr}_3(\Gamma, v) = \sum_{e \in E(v)} \mathrm{cr}(\Gamma, e)^2 \tag{2.3}$$

Vertex Insertion Approach. In the vertex insertion approach we identify a sequence $T = \langle v_1, v_2, \ldots, v_k \rangle$, $k \leq n$, so that the induced subgraph G_P of $V \setminus V(T)$ is a planar subgraph of G. Since, the respective optimization problem of removing a small set $V(T)$ of vertices is known to $\mathcal{NP}$-complete [21, 23], we take the following greedy approach. We iteratively remove the vertex with the highest crossing number cr_i from a drawing Γ of G until Γ is planar. We reinsert the vertices in reversed order at their locally optimal positions.

Edge Insertion Approach. The following heuristic is inspired by a topological edge insertion algorithm introduced by Gutwenger et al. [18]. We start with a maximal planar subgraph of G and iteratively reinsert edges e into the previous drawing. We modify each

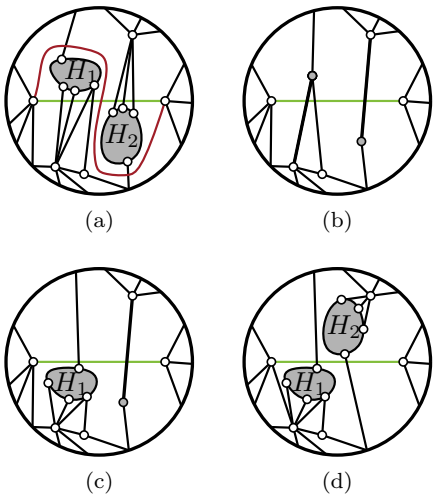


Figure 2: (a) Crossing Minimal Curve $\mathcal{C}_{uv}$ with dense Subgraphs H_1 and H_2. (b) Graph with the contracted Subgraphs H_1 and H_2. (c) H_1 unpacked. (d) H_1 and H_2 unpacked.

drawing so that we can add the edge e with a small number of crossings. We propose several strategies, either only moving the endpoints of e, or only moving the vertices incident to the edges crossing e, or moving entire subgraphs at once in order to escape local minima.

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.

Endpoint. The Endpoint strategy solely moves the endpoints u and v of the inserted edge e 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 by $\mathrm{cr}(\Gamma, e, x)$. Let C_e be the set of vertices with $\mathrm{cr}(\Gamma, e, x) > 0$. Additionally, to the endpoints of e, the Crossed Neighborhood strategy moves the vertices in C_e in some order to their locally optimal position. The Ascending Crossed Neighborhood order is an order of C_e with increasing values $\mathrm{cr}(\Gamma, e, x)$. Correspondingly, Descending Crossed Neighborhood is the respective decreasing order of C_e. Note that, we update the set C_e after each vertex movement, by maintaining C_e as a priority queue. The goal of these two strategies is to locally optimize the crossings of the edge e. The Random Crossed Neighborhood order is random permutation of C_e.

Subgraph. Let $\mathcal{C}_{uv}$ be a *crossing-minimal curve* from u to v in Γ_{-e}, i.e., a simple open Jordan Curve with u and v as its endpoints, only intersecting edges in its interior and with a minimal number of edge crossings. Let E' be the edges crossing $\mathcal{C}_{uv}$. Let R be the (not necessarily simple) region enclosed by e and $\mathcal{C}_{uv}$; see Fig. 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 $v \in V(H_j)$.

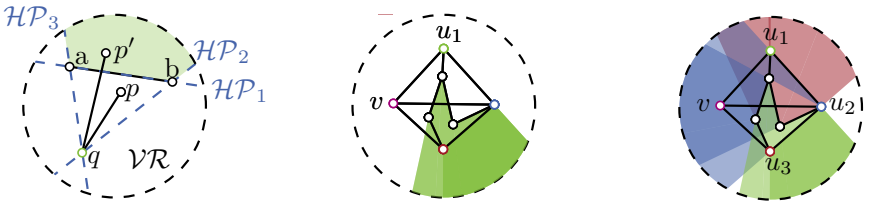
Let Γ_0 be the drawing obtained from Γ_{+e} by contracting every subgraph H_j to a vertex c_j and placing the vertex in the barycenter of the vertices of H_j. Let f_j be a connected region such that moving the vertex c_j within f in Γ_{j-1} yields the same number of crossings, i.e., $\mathrm{cr}(\Gamma_{j-1}[c_j \mapsto p]) = \mathrm{cr}(\Gamma_{j-1}[c_j \mapsto p'])$ for every pair of points $p, p' \in f$. 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 Γ_{H_j} in the interior of $f_j^\star$ and reinserting the edges E_j. 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 that the number of crossings in the drawing $\Gamma[v \mapsto p^\star]$ is minimized. In this section we show that this is possible in $O\left((kn+m)^2 \log(kn+m)\right)$ 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 that the segments s and $\mathcal{S}[q, p]$ do not intersect. Clearly, $\mathcal{VR}(q, s)$ is the union of three half-spaces; see Fig. 3a. 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



(a) (b) (c)

Figure 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.

of active edges $uv \in E$ and inactive edges e; see Fig. 3b and Fig. 3c. 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 3.1. *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 that $\mathrm{cr}(\Gamma[v \mapsto p]) < \mathrm{cr}(\Gamma[v \mapsto p'])$. 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 Fig. 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 $\mathcal{A}(\Gamma, v)$.

Due to Lemma 3.1 it is sufficient to consider only one point p in the interior of a face f in order 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.

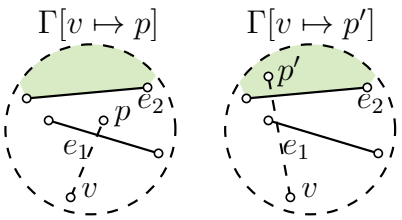


Figure 4: Moving the vertex v within a face of $\mathcal{A}(\Gamma, v)$ does not change the number of crossings. Illustration for the contradiction.

THEOREM 3.1. *Let Γ be straight-line drawing of graph $G = (V, E)$ and let $v \in V$. 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 $O\left((kn+m)^2 \log(kn+m)\right)$ time with $k = \deg v$.*

Proof. The proof relies on three helpful claims.

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

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

CLAIM 2. *Let f and g be two faces of $\mathcal{A}(\Gamma, v)$ sharing a segment s. The difference $\Delta_{f,g} = \mathrm{cr}(\Gamma[v \mapsto f]) - \mathrm{cr}(\Gamma[v \mapsto g])$ can be computed in $O(k)$ time.*

Note that due to Lemma 3.1 the difference $\Delta_{f,g}$ is well defined and it is sufficient to consider two arbitrary points $p \in f, p' \in g$. Therefore, the difference $\Delta_{f,g}$ only depends on s and v. Thus, it is sufficient to compute the number of crossings between all active edges and s in $\Gamma[v \mapsto p]$ and $\Gamma[v \mapsto p']$, respectively. This, can be done in time linear in the degree of v by checking for every neighbor u of v whether the segment $\mathcal{S}[u, v]$ intersects s in the drawings $\Gamma[v \mapsto p]$ and $\Gamma[v \mapsto p']$, respectively.

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 $\mathrm{cr}(\Gamma[v \mapsto t]) = \mathrm{cr}(\Gamma) + \sum_{(v_f, v_g) \in \Pi} \Delta_{f,g}$.*

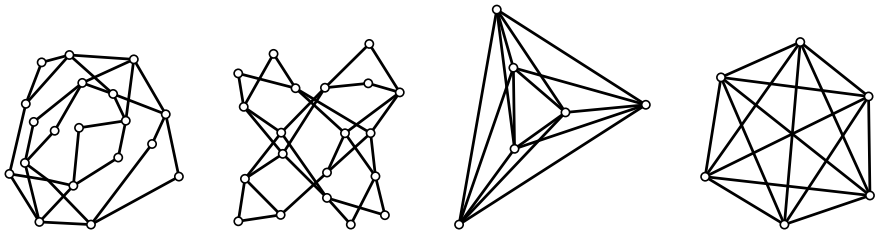


Figure 5: (a, c) Example drawings computed by our EDGE INSERTION heuristic with a repositioning with PrEd. (b, d) Drawings computed with STRESS. (a,b) North graph 20.47. (c,d) K_6. Number of crossings: (a) 5, (b) 11, (c) 3, (d) 15

Since s contains v in its interior, the crossing numbers of 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 crossing 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. In order 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 $\mathcal{A}(\Gamma, v)$. This can be achieved in time linear in the size of $\mathcal{A}(\Gamma, v)$ by performing a breadth-first search in the dual of $\mathcal{A}(\Gamma, v)$ starting at the dual vertex of f, accumulating the values $\Gamma[v \mapsto g]$ as described in Claim 3. Note that in order 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 $O\left((kn+m)^2\right)$ and the arrangement can be computed in $O\left((kn+m)^2 \log(kn+m)\right)$ time. This concludes the proof.

4 Evaluation

The goal of this section is not only to analyze the number of crossings of drawings computed by our heuristics, but also to compare it to commonly used energy-based algorithms. We use the energy-based algorithms implemented in the well-established OPEN GRAPH DRAWING FRAMEWORK[1] [10]. For our evaluation we consider synthetic and real-world instances. In Section 4.1 we give a brief description of these instances. We evaluate the energy-based algorithms and our algorithms in Section 4.3 and Section 4.4, respectively, based on descriptive characteristics and the statistical test described in Section 4.2. We conclude this Section with a running time analysis in Section 4.5.

[1] Snapshot 2017-07-23

The drawings in Fig. 5 give a first impression of the effectiveness of our algorithm compared to stress minimization. Figures 5a and 5c are obtained by our algorithm with additional runs of PrEd [4] in order to optimize the aesthetics of the drawing. The remaining two drawings are computed with stress minimization.

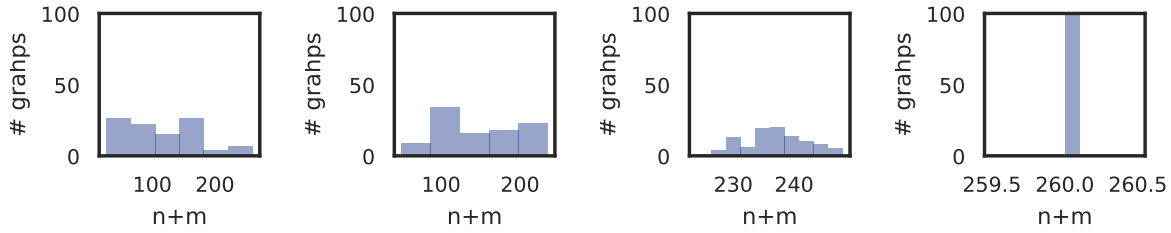
All experiments were conducted on a single core of an AMD Opteron Processor 6172 clocked at 2.1 GHz. The server is equipped with 256GB RAM. All algorithms were compiled with `g++` version 4.8.5 with optimization mode `-O3`. For geometric operations we rely on CGAL[2] and we use GMP[3] to represent coordinates.

4.1 Benchmark Instances We evaluated our algorithm on four classes of graphs, either purely synthetic or with a structure resembling real-world data. The classes ROME and NORTH (AT&T)[4] 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 [8]) and ten additional random edges. The COMMUNITY graphs are generated with the LFR-GENERATOR [22] implemented in NETWORKIT [28]. The last dataset resembles social networks with a community structure. From each of these datasets we selected 100 graphs uniformly at random. Fig. 6 shows the size distribution of these graphs.

4.2 Binomial Advantage Test The *binomial advantage test* is based on a binomial sign test [26]. 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 then $p \cdot |\Sigma|$ 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)

[2] `cgal.org`
[3] `gmplib.org`
[4] http://graphdrawing.org/data.html



(a) North (b) Rome (c) Community (d) Triangulation+X

Figure 6: 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	[12]
FMMM	ogdf::FMMMLayout	[19]
FR	ogdf::SpringEmbedderFR	[15]
Stress	ogdf::StressMinimization	[16]

graphs. We refer to a set $\Lambda = \{\Gamma_1, \ldots, \Gamma_k\}$ where $\Gamma_i \in \Gamma[G_i]$ as a *family of drawings* of $\mathcal{G}$. Let Λ^1 and Λ^2 be two families of drawings of $\mathcal{G}$. The comparison $\mathrm{cr}(\Gamma_i^1) \cdot \Delta < \mathrm{cr}(\Gamma_i^2)$ yields a sequence Σ_Δ. We say that Λ^1 *has an advantage over* Λ^2 if there exists a value $\Delta \geq 1$ such that Σ_Δ is good. If Λ^1 has an advantage over Λ^2, we refer to the maximum value $\Delta^\star$ such that $\Sigma_{\Delta^\star}$ is good, as *advantage of* Λ^1 *over* Λ^2. Let $\mathcal{L} = \{\Lambda^1, \Lambda^2, \ldots, \Lambda^l\}$ be a set of families of drawings of $\mathcal{G}$. We say a family of drawings Λ *dominates* $\mathcal{L}$ if and only if Λ has an advantage over every $\Lambda' \in \mathcal{L}$. On the other hand, we say Λ is *outperformed* by $\mathcal{L}$ if and only if every $\Lambda' \in \mathcal{L}$ has an advantage over Λ.

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. Before the execution of each algorithm the vertices are randomly positioned on an $n \times n$ grid.

Table 2 shows that both FMMM and Stress appear to compute drawings with a similar number of crossings. FR has a slightly and DH a noticeable number of crossings more in comparison to Stress and FMMM. We can confirm these observations with the plot in Fig. 7a. Each point in this plot 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. The plot indicates that drawings of the 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 plot does not show a clear preference for the drawings obtained from Stress, FMMM or FR.

The observations drawn from Table 2 and Fig. 7a neglect the fact 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. Fig. 7b shows the advantages of the algorithms on the x-Axis over the algorithms on the y-Axis. We used a significance level of $\alpha = 0.05$. For example, FMMM has an advantage of 2.2 over DH, i.e., the number of crossings in at least 50% of all drawings computed by DH are larger by a factor of 2.2 than in the corresponding drawings computed by FMMM. Further, we see that Stress dominates all other algorithms except FMMM. Table 2 show that Stress computes drawings with a sightly lower number of crossings compared to FMMM. Thus, in the following we use Stress as a representative for the class of

Table 2: Characteristics of the number of crossings.

Algorithm	⌊mean⌋	min	.25-quartile	median	.75-quartile	max
Stress	155	1	32	82	288	1220
FMMM	155	1	32	86	281	1227
FR	165	1	48	110	274	1236
DH	340	4	76	173	624	1343

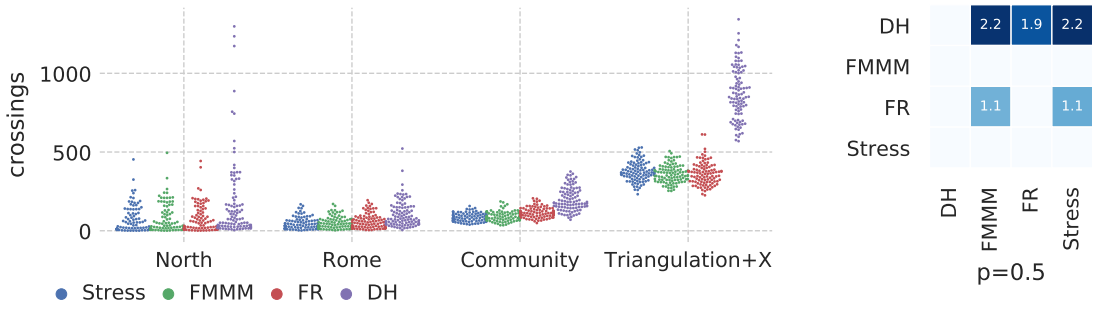


Figure 7: Comparison of drawings obtained from algorithms implemented in OGDF. (a) The number of crossings of a drawing for each graph in the class indicated on the x-Axis clustered by the algorithms. Outliers are removed. (b) Advantage between the drawings obtained from the binomial advantage.

Table 3: Configuration of the reference algorithms.

	Algorithm	**Vertex Order**	**Termination**
VM	Vertex Movement	Desc., Eq. 2.3	
VI	Vertex Insertion	Asc., Eq. 2.2	
EP	Edge Insertion	Endpoints	
EI	Edge Insertion	EP + Desc. crossed Neighbors	
Stress	ogdf::StressMinimization		convergence

energy-based algorithms. Note that, even though DH uses an energy function with the objective to reduce the number of crossings, it computes drawings with the highest number of crossings.

4.4 Geometric Heuristics In a preliminary evaluation we identified for each algorithm described in Section 2 a representative configuration listed in Table 3. The final configuration for each algorithm was selected depending on the binomial advantage test described in Section 4.2. Thus, the final configuration computes, with a high significance, drawings with a smaller number of crossings than the other configurations of the same algorithm.

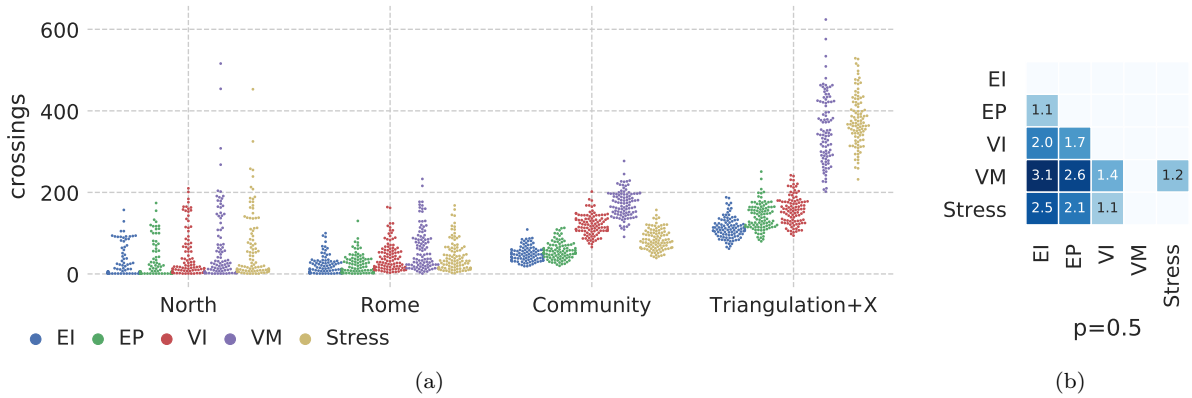
Table 4 shows that on average the drawings obtained by stress minimization have a factor of two more crossings than drawings obtained by the edge insertion approach. A comparison between Stress and VI is not that conclusive. On average VI computes drawings with a smaller number of crossings, on the other hand Stress has a smaller median value. A similar observation can be made for the comparison of Stress and VM.

Recall that a point in Fig. 8 corresponds to the number of crossings of one drawing computed by the algorithm indicated by the color. Additionally to the observations drawn from Table 4, it is apparent from this plot, that Stress computes drawings with a larger number of crossings in comparison to Edge and Vertex Insertion for graphs of the class Triangulation + X. For the remaining graph classes, the plot allows no clear distinction between Vertex Insertion and Stress. But it suggests that edge insertion computes drawings with the smallest number of crossings.

With the evaluation based on the binomial test with a significance level of $\alpha = 0.05$, depicted in Fig. 8b, we can confirm that edge insertion dominates the other algorithms. Thus, Edge Insertion computes drawings with significantly less drawings than its competitors.

Table 4: Characteristics of the number of crossings.

Algorithm	**⌊mean⌋**	**min**	**.25-quartile**	**median**	**.75-quartile**	**max**
EI	57	1	11	43	91	546
EP	70	1	11	50	111	706
VI	100	1	28	106	144	844
VM	172	1	37	152	248	839
Stress	155	1	32	82	288	1220



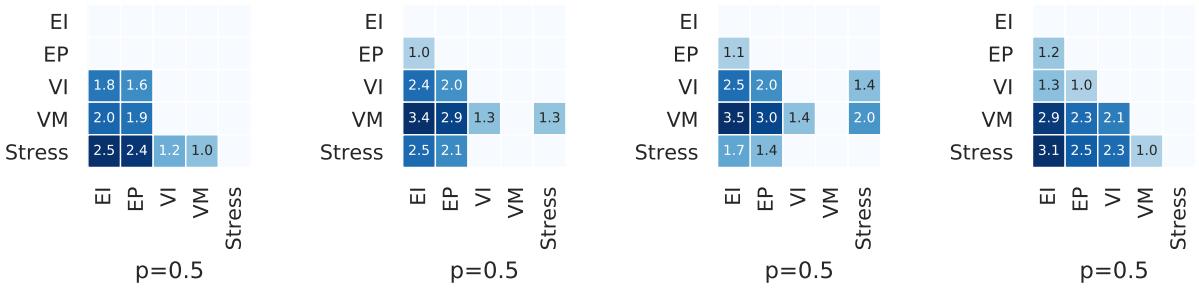
(a) (b)

Figure 8: Comparison of the final configurations of each algorithm. (a) The number of crossings of a drawing for each graph in the class indicated on the x-Axis clustered by the algorithms. Outliers are removed. (b) Advantages between the drawings obtained from the binomial advantage test.

Table 4 and Fig. 8 do not allow a conclusive distinction between STRESS and the VERTEX MOVEMENT approach. Indeed, the binomial test in Fig. 8b shows a slight advantage for STRESS over VERTEX MOVEMENT. Comparing the EI and EP heuristic shows that the movement of the crossed neighborhood in case of EDGE INSERTION further decreases the number of crossings by about 10%. Unfortunately, this comes with noticeable growth in running time as Section 4.5 shows. Further, Fig. 9 shows that there is a significant decrease in the number of crossings depending on the graph class. In particular, we can verify the hypothesis that STRESS has difficulties optimizing the number of crossing in the graph class TRIANGULATION+X.

4.5 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} seconds per instance on our graphs. We configured STRESS to stop after convergence, thus we can not expect STRESS to compute drawings with a smaller number of crossings if we increase the computing time.

We compare the remaining algorithms listed in Table 3. Fig. 10 shows the running time of each instance for all graph classes but TRIANGULATION+X. Since, all graphs in the class TRIANGULATION+X have the same size, we give numerical values in Table 5. Fig. 10 shows that EDGE INSERTION with the endpoint vertex order (EP) and VERTEX INSERTION profit from the incremental growth of the drawing, whereas VERTEX MOVEMENT has to deal with the entire graph in each iteration. Edge Insertion with the movement of the crossed neighborhood moves a significant amount of vertices in each iteration. This certainly decreases the number of crossings in a drawings but requires a considerable amount of time. Table 5 shows that the running time advantage of VERTEX INSERTION further increases for the TRIANGULATION + X graph class. With respect to the number of crossings EDGE INSERTION has only a small advantage over VERTEX INSERTION for this graph class; compare Fig. 9d.



(a) North (b) Rome (c) Community (d) Triangulation + X

Figure 9: Advantages according to the binomial advantage test for each graph class.

Table 5: Running time in seconds on the TRIANGULATION+X graph class.

Algorithm	mean	.25-quartile	median	.75-quartile	max
EI	1152.3	1568.1	1734.0	1920.5	2379.3
EP	338.1	290.1	327.4	384.1	569.8
VM	283.1	276.1	282.6	290.5	316.7
VI	28.5	27.3	28.3	29.6	34.1

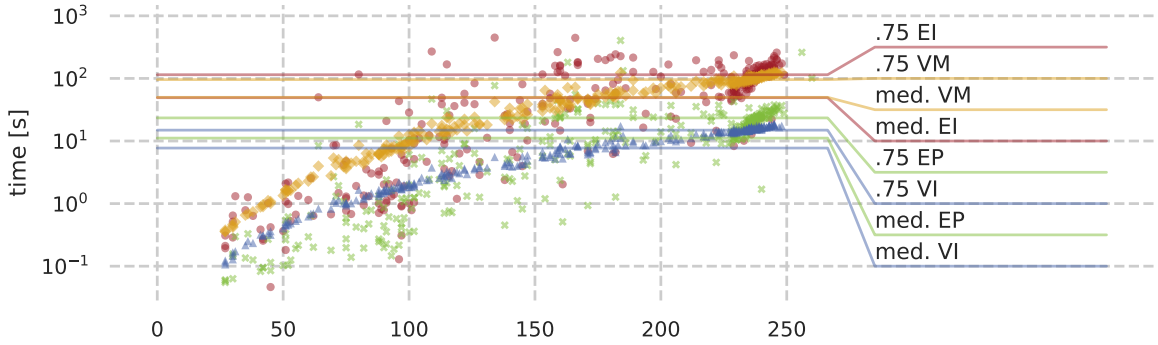


Figure 10: Size $n + m$ of the graph versus the running time. In order to increase readability the graph class TRIANGULATION+X and outliers are excluded from the plot.

5 Conclusion

In this paper we introduced several heuristics that are based on moving a vertex to its crossing minimal position. This position can be computed in $O\left((kn+m)^2 \log(kn+m)\right)$ 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. In the meantime we compared our edge insertion heuristic to an approach that minimizes the number of crossings in topological drawings. A preliminary evaluation indicates that the difference in the number of crossings of both approaches is rather small on our instances. Thus, in future work it would be interesting to see if this observation can be confirmed in an extensive evaluation. Additionally, it is desirable to further engineer our implementation to cope with larger instances.

References

[1] B. M. Ábrego, S. Fernández-Merchant, and G. Salazar. The rectilinear crossing number of K_n : Closing in (or are we?). In J. Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 5–18. Springer, 2013.

[2] O. Aichholzer. On the rectilinear crossing number. `http://www.ist.tugraz.at/staff/aichholzer/research/rp/triangulations/crossing`, 5 2017.

[3] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, C-28(9):643–647, 1979.

[4] F. Bertault. A force-directed algorithm that preserves edge-crossing properties. *Information Processing Letters*, 74(1–2):7–13, 2000.

[5] D. Bienstock. Some provably hard crossing number problems. *Discrete & Computational Geometry*, 6(3):443–459, 1991.

[6] D. Bienstock and N. Dean. Bounds for rectilinear crossing numbers. *Journal of Graph Theory*, 17(3):333–348, 1993.

[7] T. Bläsius, M. Radermacher, and I. Rutter. How to draw a planarization. In B. Steffen, C. Baier, M. van den Brand, J. Eder, M. Hinchey, and T. Margaria, editors, *Proceedings of the 43rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'17)*, pages 295–308. Springer, 2017.

[8] G. Brinkmann, B. D. McKay, et al. Fast generation of planar graphs. *MATCH - Communications in Mathematical and in Computer Chemistry*, 58(2):323–357, 2007.

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

[10] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, and P. Mutzel. *Handbook of Graph Drawing and Visualization*, chapter The Open Graph Drawing Framework (OGDF), pages 543–569. Chapman and Hall/CRC, 2013.

[11] M. Chimani and P. Hlinený. Inserting Multiple Edges into a Planar Graph. In S. Fekete and A. Lubiw,

editors, *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

[12] R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics (TOG)*, 15(4):301–331, 1996.

[13] R. Fabila-Monroy and J. López. Computational search of small point sets with small rectilinear crossing number. *Journal of Graph Algorithms and Applications*, 18(3):393–399, 2014.

[14] J. Fox, J. Pach, and A. Suk. Approximating the rectilinear crossing number. In Y. Hu and M. Nöllenburg, editors, *Proceedings of the 24th International Symposium on Graph Drawing (GD'16)*, pages 413–426. Springer, 2016.

[15] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.

[16] E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In J. Pach, editor, *Proceedings of the 12th International Symposium on Graph Drawing (GD'04)*, pages 239–250, Berlin, 2005. Springer.

[17] M. R. Garey and D. S. Johnson. Crossing number is np-complete. *SIAM Journal on Algebraic Discrete Methods*, 4(3):312–316, 1983.

[18] C. Gutwenger, P. Mutzel, and R. Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4):289–308, 2005.

[19] S. Hachul and M. Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In J. Pach, editor, *Proceedings of the 12th International Symposium on Graph Drawing (GD'04)*, pages 285–295. Springer, 2005.

[20] S. G. Kobourov. Spring embedders and force directed graph drawing algorithms. In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, pages 383–408. Chapman and Hall/CRC, 2013.

[21] M. S. Krishnamoorthy and N. Deo. Node-deletion NP-complete problems. *SIAM Journal on Computing*, 8(4):619–625, 1979.

[22] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.

[23] J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.

[24] H. C. Purchase, R. F. Cohen, and M. James. Validating graph drawing aesthetics. In F. J. Brandenburg, editor, *Proceedings of the Symposium on Graph Drawing (GD'95)*, pages 435–446. Springer, 1996.

[25] M. Schaefer. Complexity of some geometric and topological problems. In D. Eppstein and E. R. Gansner, editors, *Proceedings of the 17th International Symposium on Graph Drawing (GD'09)*, volume 5849 of *Lecture Notes in Computer Science*, pages 334–344. Springer, 2010.

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

[27] P. Shor. Stretchability of pseudolines is NP-hard. In *Applied Geometry and Discrete Mathematics–The Victor Klee Festschrift*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Socity.

[28] C. L. Staudt, A. Sazonovs, and H. Meyerhenke. Networkit: An interactive tool suite for high-performance network analysis. *arXiv preprint arXiv:1403.3005*, 2014.

[29] I. Vrt'o. Bibliography on crossing numbers of graphs. `ftp://ftp.ifi.savba.sk/pub/imrich/crobib.pdf`, 2014.