Drawing Planar Graphs with Few Geometric Primitives[*]

Gregor Hültenschmidt[1], Philipp Kindermann[1], Wouter Meulemans[2], and André Schulz[1]

[1]*LG Theoretische Informatik, FernUniversität in Hagen, Germany,*
gregorhueltenschmidt@gmx.de,
{philipp.kindermann | andre.schulz}@fernuni-hagen.de
[2]*TU Eindhoven, The Netherlands,* w.meulemans@tue.nl

July 31, 2017

Abstract

We define the *visual complexity* of a plane graph drawing to be the number of basic geometric objects needed to represent all its edges. In particular, one object may represent multiple edges (e.g., one needs only one line segment to draw two collinear edges of the same vertex). Let n denote the number of vertices of a graph. We show that trees can be drawn with $3n/4$ straight-line segments on a polynomial grid, and with $n/2$ straight-line segments on a quasi-polynomial grid. Further, we present an algorithm for drawing planar 3-trees with $(8n-17)/3$ segments on an $O(n) \times O(n^2)$ grid. This algorithm can also be used with a small modification to draw maximal outerplanar graphs with $3n/2$ edges on an $O(n) \times O(n^2)$ grid. We also study the problem of drawing maximal planar graphs with circular arcs and provide an algorithm to draw such graphs using only $(5n-11)/3$ arcs. This provides a significant improvement over the lower bound of $2n$ for line segments for a nontrivial graph class.

1 Introduction

The complexity of a graph drawing can be assessed in a variety of ways: area, crossing number, bends, angular resolution, etc. All these measures have their justification, but in general it is challenging to optimize all of them in a single drawing. More recently, the *visual complexity* was suggested as another quality measure for drawings [18]. The visual complexity denotes the number of simple geometric entities used in the drawing.

Typically, we consider as entities either straight-line segments or circular arcs. To distinguish these two types of drawings, we call the former *segment drawings* and the latter *arc drawings*. The idea is that we can use, for example, a single segment to draw a path of collinear edges. The hope is that a drawing that consists of only a few geometric entities is easy to perceive. It is a natural question to ask for the best possible visual complexity of a drawing of a given graph. Unfortunately, it is an NP-hard problem to determine the smallest number of segments necessary in a segment drawing [10]. However, we can still expect to prove bounds for certain graph classes.

[*]The work of P. Kindermann and A. Schulz was supported by DFG grant SCHU 2458/4-1. The work of W. Meulemans was supported by Marie Skłodowska-Curie Action MSCA-H2020-IF-2014 656741.

Table 1: Upper bounds on the visual complexity. Here, n is the number of vertices, ϑ the number of odd-degree vertices and e the number of edges. Constant additions or subtractions have been omitted.

Class	**Segments**		**Arcs**	
trees	$\vartheta/2$	[8]	$\vartheta/2$	[8]
maximal outerplanar	n	[8]	n	[8]
3-trees	$2n$	[8]	$11e/18$	[18]
3-connected planar	$5n/2$	[8]	$2e/3$	[18]
cubic 3-connected planar	$n/2$	[12, 15]	$n/2$	[12, 15]
triangulation	$7n/3$	[9]	$\mathbf{5n/3}$	**Thm. 5**
4-connected triangulation	$9n/4$	[9]	$\mathbf{3n/2}$	**Thm. 6**
4-connected planar	$9n/4$	[9]	$\mathbf{9n/2 - e}$	**Thm. 8**
planar	$16n/3 - e$	[9]	$\mathbf{14n/3 - e}$	**Thm. 7**

Related work. For a number of graph classes, upper and lower bounds are known for segment drawings and arc drawings; the upper bounds are summarized in Table 1. However, these bounds (except for cubic 3-connected graphs) do not require the drawings to be on the grid. In his thesis, Mondal [14] gives an algorithm for triangulations with few segments—but more than Durocher and Mondal [9] require—on an exponential grid.

There are three trivial lower bounds for the number of segments required to draw any graph $G = (V, E)$ with n vertices and e edges: (i) $\vartheta/2$, where ϑ is the number of odd-degree vertices, (ii) $\max_{v \in V} \lceil \deg(v)/2 \rceil$, and (iii) $\lceil e/(n-1) \rceil$. For triangulations and general planar graphs, a lower bound of $2n - 2$ and $5n/2 - 4$, respectively, is known [8]. Note that the trivial lower bounds are the same as for the slope number of graphs [20], that is, the minimum number of slopes required to draw all edges, and that the number of slopes of a drawing is upper bounded by the number of segments. Chaplick et al. [3, 4] consider a similar problem where all edges are to be covered by few lines (or planes); the difference to our problem is that collinear segments are only counted once.

Contributions. In this work, we present two types of results. In the first part (Sections 2, 3 and 4), we discuss algorithms for segment drawings on the grid with low visual complexity. This direction of research was posed as an open problem by Dujmović et al. [8], but only a few results exist; see Table 2. We present an algorithm that draws trees on an $O(n^2) \times O(n^{1.58})$ grid using $3n/4$ straight-line segments. For comparison, the drawings of Schulz [18] need also $3n/4$ arcs on a smaller $O(n) \times O(n^{1.58})$ grid, but use the more complex circular arcs instead. Our segment drawing algorithm for trees can be modified to generate drawings with an optimal number of $\vartheta/2$ segments on a quasi-polynomial grid. We also present algorithms to compute segment drawings of planar 3-trees and maximal outerplanar graphs, both on an $O(n) \times O(n^2)$ grid. In the case of planar 3-trees, the algorithm needs at most $(8n - 17)/3$ segments, and in the case of maximum outerplanar graphs the algorithm needs at most $3n/2$ segments.

Finally, in Section 5, we study arc drawings of triangulations and general planar graphs. In particular, we prove that $(5n - 11)/3$ arcs are sufficient to draw any triangulation with n vertices. We highlight that this bound is significantly lower than the $2n - 2$ *lower* bound known for segment drawings [8] and the so far best-known $2e/3 = 2n$ upper bound for circular arc drawings [18]. A straightforward extension shows that $(14n - 3e - 29)/3$ arcs are sufficient for general planar graphs with e edges.

Table 2: Same as in Table 1 but for grid drawings.

Class	Type	Vis.Compl.	Grid	Ref.
trees	arcs	$3n/4$	$O(n) \times O(n^{1.58})$	[18]
trees	segments	$\mathbf{3n/4}$	$\mathbf{O(n^2) \times O(n^{1.58})}$	**Thm. 1**
trees	segments	$\vartheta/\mathbf{2}$	**pseudo-polynom.**	**Thm. 2**
cubic planar 3-conn.	segments	$n/2$	$O(n) \times O(n)$	[12, 15]
maximal outerplanar	segments	$\mathbf{3n/2}$	$\mathbf{O(n) \times O(n^2)}$	**Thm. 4**
triangulation	segments	$8n/3$	$O((3.63n)^{4n/3})$	[14]
planar 3-tree	segments	$\mathbf{8n/3}$	$\mathbf{O(n) \times O(n^2)}$	**Thm. 3**

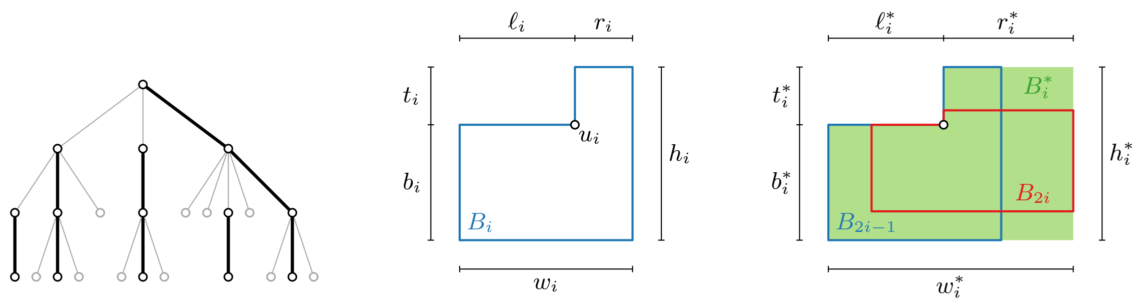
Preliminaries. Given a triangulated planar graph $G = (V, E)$ on n vertices, a *canonical order* $\sigma = (v_1, \dots, v_n)$ is an ordering of the vertices in V such that, for $3 \le k \le n$, (i) the subgraph G_k of G induced by $v_1, \dots, v_k$ is biconnected, (ii) the outer face of G_k consists of the edge (v_2, v_1) and a path C_k, called *contour*, that contains v_k, and (iii) the neighbors of v_k in G_{k-1} form a subpath of C_{k-1} [6, 7]. A canonical order can be constructed in reverse order by repeatedly removing a vertex without a chord from the outer face.

Most of our algorithms make ample use of *Schnyder realizers* [17]. Assume we selected a face as the outer face with vertices v_1, v_2 and v_n. We decompose the interior edges into three trees: T_1, T_2, and T_n rooted at v_1, v_2, and v_n, respectively. The edges of the trees are oriented to their roots. For $k \in \{1, 2, n\}$, we call each edge in T_k a *k-edge* and the parent of a vertex in T_k its *k-parent.* In the figures of this paper, we will draw 1-edges red, 2-edges blue, and n-edges green. The decomposition is a Schnyder realizer if at every interior vertex the edges are cyclically ordered as: outgoing 1-edge, incoming n-edges, outgoing 2-edge, incoming 1-edges, outgoing n-edge, incoming 2-edges. The trees of a Schnyder realizer are also called *canonical ordering trees*, as each describes a canonical order on the vertices of G by a (counter-)clockwise pre-order traversal [5]. There is a unique *minimal* realizer such that any interior cycle in the union of the three trees is oriented clockwise [2]; this realizer can be computed in linear time [2, 17]. The number of such cycles is denoted by Δ_0 and is upper bounded by $\lfloor (n-1)/2 \rfloor$ [21]. Bonichon et al. [1] prove that the total number of leaves in a minimal realizer is at most $2n - 5 - \Delta_0$.

2 Trees with segments on the grid

Let $T = (V, E)$ be an undirected tree. Our algorithm follows the basic idea of the circular arc drawing algorithm by Schulz [18]. We make use of the *heavy path decomposition* [19] of trees, which is defined as follows. First, root the tree at some vertex r and direct all edges away from the root. Then, for each non-leaf u, compute the size of each subtree rooted in one of its children. Let v be the child of u with the largest subtree (one of them in case of a tie). Then, (u, v) is called a *heavy edge* and all other outgoing edges of u are called *light edges.* The maximal connected components of heavy edges form the *heavy paths* of the decomposition.

We call the vertex of a heavy path closest to the root its *top node* and the subtree rooted in the top node a *heavy path subtree.* We define the *depth* of a heavy path (subtree) as follows. We treat each leaf that is not incident to a heavy edge as a heavy path of depth 0. The depth of each other heavy path P is by 1 larger than the maximum depth of all heavy paths that are connected from P by an outgoing light edge. Heavy path subtrees of common depth are disjoint.



(a) A heavy path decomposition.

(b) The heavy path box B_i with top node u_i.

(c) The merged box B_i^* for B_{2i-1} and B_{2i}.

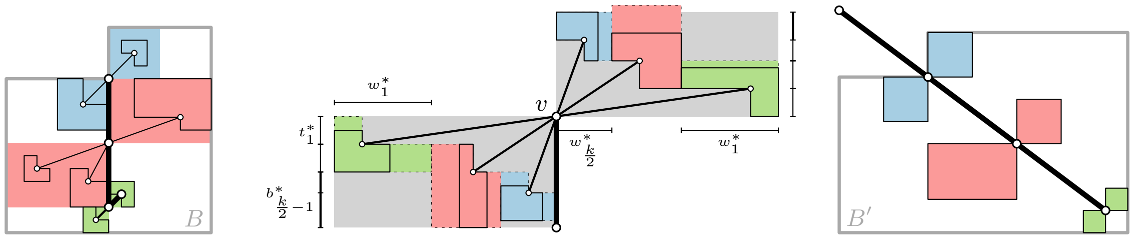
Figure 1: The heavy path boxes and the merged boxes with their respective lengths.

Boxes. We order the heavy paths nondecreasingly by their depth and then draw their subtrees in this order. Each heavy path subtree is placed completely inside an L-shaped box (*heavy path box*) with its top node placed at the reflex angle; see Fig. 1b for an illustration of a heavy path box B_i with top node u_i, width $w_i = \ell_i + r_i$, and height $h_i = t_i + b_i$. We require that (i) heavy path boxes of common depth are disjoint, (ii) u_i is the only vertex on the boundary, and (iii) $b_i \geq t_i$. Note that the boxes will be mirrored horizontally and/or vertically in some steps of the algorithm. We assign to each heavy path subtree of depth 0 a heavy path box B_i with $\ell_i = r_i = t_i = b_i = 1$.

Drawing. Assume that we have already drawn each heavy path subtree of depth d. When drawing the subtree of a heavy path $\langle v_1, \ldots, v_m \rangle$ of depth $d+1$, we proceed as follows. The last vertex on a heavy path has to be a leaf, so v_m is a leaf. If $\operatorname{outdeg}(v_{m-1})$ is odd, we place the vertices $v_1, \ldots, v_m$ on a vertical line; otherwise, we place only the vertices $v_1, \ldots, v_{m-1}$ on a vertical line and treat v_m as a heavy path subtree of depth 0 that is connected to v_{m-1}. For $1 \leq h \leq m-1$, all heavy path boxes adjacent to v_h will be drawn either in a rectangle on the left side of the edge (v_h, v_{h+1}) or in a rectangle on the right side of the edge (v_{h-1}, v_h) (a rectangle that has v_1 as its bottom left corner for $h = 1$); see Fig. 2a for an illustration with even $\operatorname{outdeg}(v_{m-1})$.

We now describe how to place the heavy path boxes $B_1, \ldots, B_k$ with top node $u_1, \ldots, u_k$, respectively, incident to some vertex v on a heavy path into the rectangles described above. First, assume that k is even. Then, for $1 \leq i \leq k/2$, we order the boxes such that $b_{2i} \leq b_{2i-1}$. We place the box B_{2i-1} in the lower left rectangle and box B_{2i} in the upper right rectangle in such a way that the edges (v, u_{2i-1}) and (v, u_{2i}) can be drawn with a single segment. To this end, we construct a *merged* box B_i^* as depicted in Fig. 1c with $\ell_i^* = \max\{\ell_{2i-1}, \ell_{2i}\}$, $r_i^* = \max\{r_{2i-1}, r_{2i}\}$, and $w_i^* = \ell_i^* + r_i^*$; the heights are defined analogously. The merged boxes will help us reduce the number of segments. We mirror all merged boxes horizontally and place them in the lower left rectangle (of width $\sum_{i=i}^{k/2} w_i^*$) as follows. We place B_1^* in the top left corner of the rectangle. For $2 \leq j \leq k/2$, we place B_j^* directly to the right of B_{j-1}^* such that its top border lies exactly t_{j-1}^* rows below the top border of B_{j-1}^*. Symmetrically, we place the merged boxes (vertically mirrored) in the upper right rectangle. Finally, we place each box B_{2i-1} (horizontally mirrored) in the lower left copy of B_i^* such that their inner concave angles coincide, and we place each box B_{2i} (vertically mirrored) in the upper right copy of B_i^* such that their inner concave angles coincide; see Fig. 2b. If k is odd, then we simply add a dummy box $B_{k+1} = B_k$ that we remove afterwards.

Analysis. We will now calculate the width w and the height h of this construction. For



(a) (b) (c)

Figure 2: (a) Placement of a heavy path, its box B, and areas for the adjacent heavy path boxes. (b) Placement of the heavy path boxes adjacent to v. (c) Further improvement on the visual complexity via increasing the size of the boxes.

the width, we have $w = 2\sum_{i=1}^{k/2} w_i^* = 2\sum_{i=1}^{k/2} \max\{w_{2i-1}, w_{2i}\} \le 2\sum_{i=1}^{k} w_i$. The height of each rectangle in the construction is at least $2\sum_{i=1}^{k/2} t_i^*$, but we have to add a bit more for the bottom parts of the boxes; in the worst case, this is $\max_{1\le j\le k/2} b_{2j-1}$ in the lower rectangle and $\max_{1\le h\le k/2} b_{2h}$ in the upper rectangle. Since we require $b_i \ge t_i$ for each i, we have

$$h \le 2\sum_{i=1}^{k/2} t_i^* + \max_{1\le j\le k/2} b_{2j-1} + \max_{1\le h\le k/2} b_{2h} \le 2\sum_{i=1}^{k} t_i + \sum_{j=1}^{k} b_i \le \frac{3}{2}\sum_{i=1}^{k} h_i.$$

Since all heavy path trees of common depth are disjoint, the heavy path boxes of common depth are also disjoint. Further, we place only the top vertex of a heavy path on the boundary of its box. Finally, since we order the boxes such that $b_{2i} \le b_{2i-1}$ for each i, for the constructed box B we have $b \ge t$.

Due to the properties of a heavy path decomposition, the maximum depth is $\lceil \log n \rceil$. Recall that we place the depth-0 heavy paths in a box of width and height 2. Hence, by induction, a heavy path subtree of depth d with n' vertices lies inside a box of width $2 \cdot 2^d \cdot n'$ and height $2 \cdot (3/2)^d \cdot n'$. Thus, the whole tree is drawn in a box of width $2 \cdot 2^{\lceil \log n \rceil} n = O(n^2)$ and height $2 \cdot (3/2)^{\lceil \log n \rceil} n = O(n^{1+\log 3/2}) \subseteq O(n^{1.58})$. Following the analysis of Schulz [18], the drawing uses at most $\lceil 3e/4 \rceil = \lceil 3(n-1)/4 \rceil$ segments.

Theorem 1. *Every tree admits a straight-line drawing that uses at most $\lceil 3e/4 \rceil$ segments on an $O(n^2) \times O(n^{1.58})$ grid.*

We finish this section with an idea of how to get a grid drawing with the best possible number of straight-line segments. Due to the limited space we give only a sketch. Observe that there is only one situation in which the previous algorithm uses more segments than necessary, that is the top node of each heavy path. This suboptimality can be "repaired" by *tilting* the heavy path as sketched in Fig. 2c. Note that the incident subtrees with smaller depth will only be translated. To make this idea work, we have to blow up the size of the heavy path boxes. We are left with scaling in each "round" by a polynomial factor. Since there are only $\log n$ rounds, we obtain a drawing on a quasi-polynomial grid. However, an implementation of the algorithm shows that some simple heuristics can already substantially reduce the drawing area, which gives hope that drawings on a polynomial grid exist for all trees.

Theorem 2. *Every tree admits a straight-line drawing with the smallest number of straight-line segments on a quasi-polynomial grid.*

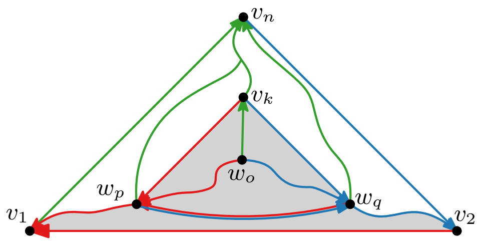


Figure 3: Proof that the edge (w_p, w_q) exists.

3 Planar 3-trees with few segments on the grid

In this section, we show how to construct drawings of planar 3-trees with few segments on the grid. We begin by introducing some notation. A *3-tree* is a maximal graph of treewidth k, that is, no edges can be added without increasing the treewidth. Each *planar 3-tree* can be produced from the complete graph K_4 by repeatedly adding a vertex into a triangular face and connecting it to all three vertices incident to this face. This operation is also known as *stacking*. Any planar 3-tree admits exactly one Schnyder realizer, and it is cycle-free [2], that is, directing all edges according to T_1, T_2, and T_n gives a directed acyclic graph.

Let $\mathcal{T}$ be a planar 3-tree. Let T_1, T_2, and T_n rooted at v_1, v_2, and v_n, respectively, be the canonical ordering trees of the unique Schnyder realizer of $\mathcal{T}$. Without loss of generality, let T_1 be the canonical ordering tree having the fewest leaves, and let $\sigma = (v_1, v_2, \ldots, v_n)$ be the canonical order induced by a clockwise pre-order walk of T_2. The following lemma holds for any $v_k, 1 \leq k \leq n$.

Lemma 1. *Let $C_{k-1} = (w_1, \ldots, w_r)$ be the contour of G_{k-1}, let w_p be the 1-parent of v_k, and let w_q be the 2-parent of v_k. Then,*

(a) either $(w_q, w_p) \in T_1$ or $(w_p, w_q) \in T_2$

(b) the subgraph inside the triangle (w_p, w_q, v_k) is a planar 3-tree

(c) if $q > p+1$, then $w_{p+1}, \ldots, w_{q-1}$ lie inside the triangle (w_p, w_q, v_k), $(w_{p+1}, w_p) \in T_1$, and $(w_{q-1}, w_q) \in T_2$.

Proof. Proof for (a): For $q = p + 1$ the statement holds trivially. So we assume that there exists a vertex w_o on C_k between w_p and w_q. Let $X = \{v_n\}$ and $Y = \{w_o\}$. Obviously, the contour C_k is an X-Y-separator; see Fig. 3. Let Z be a minimal X-Y-separator that only contains vertices from C_k. By the properties of the Schnyder realizer, each incoming n-edge of a vertex on C_k comes from a vertex below the contour. Hence, there is a path from each vertex on C_k to v_n in T_n that does not revisit C_k. Further, the paths (w_o, v_k), $(w_o, w_{o-1}, \ldots, w_p)$, and $(w_o, w_{o+1}, \ldots, w_q)$ connect w_o to each of v_k, w_p and w_q without traversing any other vertex from C_k. Hence, v_k, w_p, and w_q all have to lie inside Z. Otherwise a v_n–w_o-path would remain after deleting Z. Rose [16] has shown that every minimal X-Y-separator in a k-tree is a k-clique; thus, Z has to be the 3-cycle (w_p, w_q, v_k). The edge (w_p, w_q) cannot be an n-edge, since otherwise we would have a cycle in T_n. So either $(w_q, w_p) \in T_1$ or $(w_p, w_q) \in T_2$.

Proof for (b): Mondal et al. [15] have shown that this is true for every triangle in a planar 3-tree.

Proof for (c): The vertices have to lie inside the triangle (w_p, w_q, v_k) since otherwise their outgoing n-edge to v_k would cross the edge (w_p, w_q). Further, since the subgraph inside (w_p, w_q, v_k) is again a planar 3-tree, w_p, w_q, and v_k are the roots of the corresponding Schnyder realizer; $(w_{p+1}, w_p) \in T_1$ and $(w_{q-1}, w_q) \in T_2$ follows immediately. □

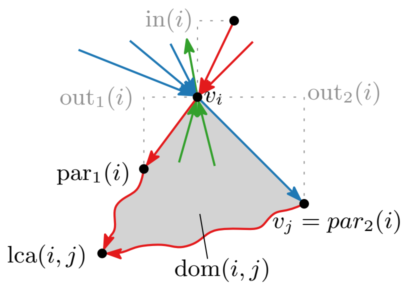


Figure 4: Definitions for the drawing algorithm for planar 3-trees.

The following lemma was proven by Durocher and Mondal [9].

Lemma 2 ([9]). *Let $a_1, \ldots, a_m$ be a strictly x-monotone polygonal chain C. Let p be a point above C such that the segments a_1p and a_mp do not intersect C except at a_1 and a_m. If the positive slopes of the edges of C are smaller than* slope(a_1, p), *and the negative slopes of the edges of C are greater than* slope(p, a_m), *then every a_i is visible from p.* □

Overview and Notation. The main idea of the algorithm is to draw the graph such that T_1 is drawn with one segment per leaf. We inductively place the vertices according to the canonical order $\sigma = (v_1, \ldots, v_n)$ and refer to the step in which vertex v_k is placed as *step k*. For the algorithm, we make use of the following additional notation; see Fig. 4. For each vertex v_i, the *1-out-slope* out$_1(i)$ is the slope of its outgoing 1-edge, the *2-out-slope* out$_2(i)$ is the slope of its outgoing 2-edge, and the *in-slope* in(i) is the highest slope of the incoming 1-edges in the current drawing. Further, we denote by par$_1(i)$ the 1-parent of v_i and by par$_2(i)$ the 2-parent of v_i. For two vertices v_i, v_j, we denote by lca(i, j) the lowest common ancestor of v_i and v_j in T_1. For an edge (v_i, v_j) we call the closed region bounded by (v_i, v_j), the path $(v_i, \ldots, \text{lca}(i, j))$, and the path $(v_j, \ldots, \text{lca}(i, j))$ the *domain* dom(i, j) of (v_i, v_j). For each step k, we denote by λ_k the number of leafs in the currently drawn subtree of T_1, by s_k the number of segments that are used to draw T_1, and by η_k the highest slope of the 1-edges in the current drawing. We denote by $C_k^{\rightarrow}$ the part of the contour C_k between v_k and v_2. Note that par$_2(k) \in C_{k-1}^{\rightarrow}$ because the canonical order is induced by T_2 in clockwise order; hence, either v_k and v_{k-1} are connected, or par$_2(k)$ is an ancestor of v_{k-1} on T_2.

Invariants. After each step k, $3 \leq k \leq n$, we maintain the following invariants:

(I1) The contour C_k is a strictly x-monotone polygonal chain; the x-coordinates along $C_k^{\rightarrow}$ increase by 1 per vertex.

(I2) The 1-edges are drawn with $s_k = \lambda_k$ segments in total with integer slopes between 1 and $\eta_k \leq \lambda_k$.

(I3) For each $(v_i, v_j) \in C_k^{\rightarrow}$, we have lca$(i, j) = \text{par}_1(j)$ and for each 1-edge $e \neq (\text{par}_1(j), v_j)$ in dom(i, j) it holds that slope$(e) > \text{out}_1(j)$.

(I4) For each $v_i \in C_k^{\rightarrow}$, out$_1(i) > \text{out}_2(i)$.

(I5) The current drawing is crossing-free and for each $(v_i, v_j) \in T_n$, slope$(v_i, v_j) > \text{out}_1(j)$.

(I6) Vertex v_1 is placed at coordinate $(0, 0)$, v_2 is placed at coordinate $(k - 1, 0)$, and every vertex lies inside the rectangle $(0, 0) \times (k - 1, (k - 1)\lambda_k)$.

The Algorithm. The algorithm starts with placing v_1 at $(0, 0)$, v_2 at $(2, 0)$, and v_3 at $(1, 1)$. Obviously, all invariants (I1)–(I6) hold. In step $k > 3$, the algorithm proceeds in two steps. Recall that v_k is a neighbor of all vertices on the contour between $v_l = \text{par}_1(k)$ and $v_r = \text{par}_2(k)$.

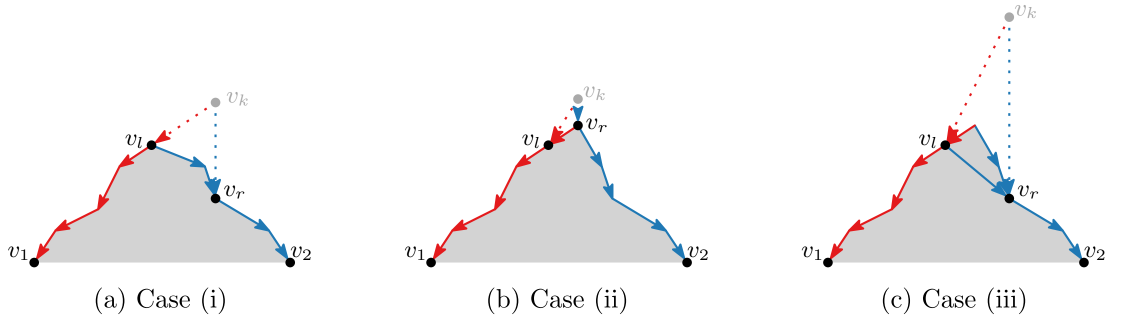


Figure 5: Inserting vertex v_k while maintaining invariant (I5).

In the first step, the *insertion step*, v_k is placed in the same column as v_r. We distinguish between three cases to obtain the y-coordinate of v_k; see Fig. 5. (i) If no incoming 1-edge of v_l has been drawn yet, then we draw the edge (v_l, v_k) with slope $\text{out}_1(l)$; (ii) If v_l already has an incoming 1-edge and v_l and v_r are the only neighbors of v_k in the current drawing, then we draw the edge (v_l, v_k) with slope $\text{in}(l) + 1$; (iii) Otherwise, we draw the edge (v_l, v_k) with slope $\eta_{k-1} + 1$. Note that this does not maintain invariant (I1).

In the second step of the algorithm, the *shifting step*, the vertices between v_r and v_2 on the contour C_k have to be shifted to the right without increasing the number of segments s_k used to draw T_1. To this end, we iteratively extend the outgoing 1-edge of these vertices, starting with v_r, to increase their x-coordinates all by 1; see Fig. 5b. This procedure places the vertices on the grid since the slopes of the extended edges are all integer by invariant (I2).

Theorem 3. *Every planar 3-tree admits a straight-line drawing that uses at most* $(8n - 17)/3$ *segments on an* $O(n) \times O(n^2)$ *grid.*

Proof. It remains to show that, after each step $k > 3$, all invariants (I1)–(I6) hold. We denote by $x(v)$ the x-coordinate of the vertex v, and similarly, by $y(v)$ the y-coordinate of v.

Invariant (I1). After inserting v_k, we have $C_k = (v_1, \ldots, v_l, v_k, v_r, \ldots, v_2)$. The part from v_1 to v_l is strictly x-monotone by induction. Since the vertices from v_r to v_2 are all moved to the right by one, this part is strictly x-monotone with the x-coordinates increasing by one per vertex by induction. Since $x(v_l) < x(v_k) = x(v_r) + 1$, the invariant holds for the whole contour.

Invariant (I2) In case (i) of the insertion step, v_l changes from a leaf to a non-leaf, while v_k is added as a leaf. Since (v_l, v_k) is drawn with the same slope as the outgoing 1-edge of v_l, we have $s_k = s_{k-1} = \lambda_{k-1} = \lambda_k \geq \eta_k$. In cases (ii) and (iii) of the insertion step, v_l was not a leaf before and a new integer slope is used to draw (v_l, v_k), so we have $s_k = s_{k-1} + 1 = \lambda_{k-1} + 1 = \lambda_k$; since the maximum slope increases by at most one, we have $\eta_k \leq \eta_{k-1} + 1 \leq \lambda_k$. For the shift step, as the vertices on $C_k^{\rightarrow}$ have no incoming

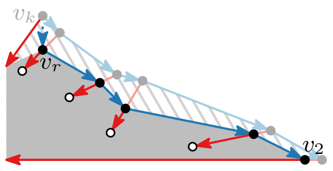
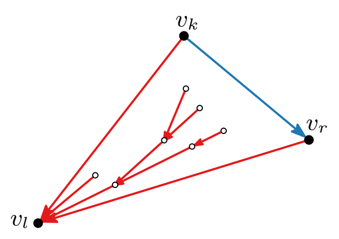
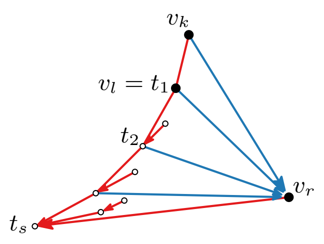


Figure 6: Shifting $v_r, \ldots, v_2$ along their outgoing 1-edge.



(a) Case $(v_r, v_l) \in T_1$



(b) Case $(v_l, v_r) \in T_2$

Figure 7: The domain $\mathrm{dom}(k, r)$ as in invariant (I3).

1-edges, and the slopes of their outgoing 1-edges do not change, the number of segments remains the same.

Invariant (I3). We only have to address the insertion step, since the shifting step does not affect the relevant slopes. Since the slopes of the 1-edges do not change, the invariant holds for edges on $(v_r, \dots, v_2)$ by induction; hence, it suffices to show the invariant for (v_k, v_r). By Lemma 1(a), the edge (v_l, v_r) exists in T_1 or T_2. We distinguish between two cases.

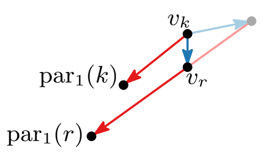
Case 1: $(v_r, v_l) \in T_1$; see Fig. 7a. It immediately follows that $\mathrm{lca}(k, r) = v_l = \mathrm{par}_1(k)$. Hence, the domain $\mathrm{dom}(k, r)$ is bounded by the triangle (v_l, v_r, v_k), which is a planar 3-tree by Lemma 1(b) with v_l as the root of T_1. By construction, the edge (v_r, v_l) has the smallest slope of all incoming 1-edges of v_l within the domain. Further, by construction, the first incoming 1-edge of each vertex is assigned the same slope as the outgoing 1-edge, while all other incoming 1-edges are assigned higher slopes. Thus, all 1-edges in the domain must have higher slopes than (v_r, v_l).

Case 2: $(v_l, v_r) \in T_2$; see Fig. 7b. Consider the path $(v_l = t_1, \dots, t_o = v_1)$ in T_1 from v_l to v_1. Let t_s be the first vertex on this path such that $(t_s, v_r) \notin T_2$; since $(v_1, v_r) \notin T_2$ and $(v_l, v_r) \in T_2$, this vertex exists with $s > 1$. By choice of t_s, we have that $(t_{s-1}, t_s) \in T_1$ and $(t_{s-1}, v_r) \in T_2$; hence, by Lemma 1(a), the edge (v_r, t_s) exists and since $(t_s, v_r) \notin T_2$ it has to be a 1-edge. Hence, we have $\mathrm{par}_1(r) = t_s = \mathrm{lca}(k, r)$. The domain $\mathrm{dom}(k, r)$ is bounded by the edges (v_k, v_r), (v_r, t_s), and the path $(t_0 := v_k, t_1, \dots, t_s)$. By Lemma 1(b), the domain can be divided into s planar 3-trees (t_i, v_r, t_{i-1}), $1 \le i \le s$. For the triangle (t_s, v_r, t_{s-1}), the argument from Case 1 can be directly applied; for $i < s$, we can use the same argument to show that no 1-edge has a higher slope than (t_i, t_{i+1}), which proves the invariant.

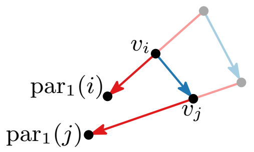
Invariant (I4). Let $v_i \in C_k^{\rightarrow}$. First, consider the case that $v_i = v_k$; see Fig. 8a. In the insertion step, v_k is placed at the same x-coordinate as v_r. In the shifting step, v_r is moved to the right by one column and upwards by $\mathrm{out}_1(r)$ rows; hence, we have $y(v_r) < y(v_k) + \mathrm{out}_1(r)$. Since $\mathrm{dom}(k, r)$ contains $v_l = \mathrm{par}_1(k)$, it follows from invariant (I3) that $\mathrm{out}_2(k) = y(v_r) - y(v_k) < \mathrm{out}_1(r) < \mathrm{out}_1(k)$.

Now, consider the case that $v_2 \ne v_i \ne v_k$; see Fig. 8b. Let $v_j = \mathrm{par}_2(i)$. We had $\mathrm{out}_1(i) > \mathrm{out}_2(i)$ before the shifting step of the algorithm. In the shifting step, $\mathrm{out}_1(i)$ does not change. However, v_i and v_j are shifted to the right by one column and upwards by $\mathrm{out}_1(i)$ rows and $\mathrm{out}_1(j)$ rows, respectively; hence, $\mathrm{out}_2(i)$ increases by $\mathrm{out}_1(j) - \mathrm{out}_1(i)$. Since $\mathrm{dom}(i, j)$ contains $v_l = \mathrm{par}_1(i)$, it follows from invariant (I3) that $\mathrm{out}_1(j) < \mathrm{out}_1(i)$. This implies that $\mathrm{out}_2(i)$ becomes smaller by the shifting step, so $\mathrm{out}_1(i) > \mathrm{out}_2(i)$ is maintained.

Invariant (I5). First, consider the drawing after the insertion step, but before the shifting step. By induction, the drawing was crossing-free before this step. Since no existing edge is modified, each crossing has to involve an edge of v_k. We will use Lemma 2 to show



(a) $i = k$ and $v_r = \mathrm{par}_2(k)$



(b) $i \neq k$ and $v_j = \mathrm{par}_2(i)$

Figure 8: Shifting $\mathrm{par}_2(i)$ while maintaining invariant (I4).

that v_k sees all its neighbors. By invariant (I1), the neighbors of v_k lie on a strictly x-monotone polygonal chain. Because of $\mathrm{slope}(v_k, v_r) = -\infty$, it suffices to show that the edges between v_l and v_r on the contour have smaller slope than $\mathrm{slope}(v_l, v_k)$.

Consider case (i) of the insertion step: v_l has no incoming 1-edge; see Fig. 5a. Then, $(v_l, v_r) \in C_{k-1}$; otherwise, we get a contradiction from Lemma 1(c). Hence, v_l and v_r are the only neighbors of v_k and we only have to show that $\mathrm{slope}(v_k, v_l) > \mathrm{slope}(v_l, v_r)$. By construction and invariant (I4), we have $\mathrm{slope}(v_k, v_l) = \mathrm{out}_1(l) > \mathrm{out}_2(l) = \mathrm{slope}(v_l, v_r)$.

Consider case (ii) of the insertion step: v_l already has an incoming 1-edge and v_l and v_r are the only neighbors of v_k; see Fig. 5b. In particular, that means that $(v_r, v_l) \in T_1$ and, by construction, $\mathrm{slope}(v_k, v_l) = \mathrm{in}(l) + 1 > \mathrm{in}(l) \geq \mathrm{slope}(v_r, v_l)$.

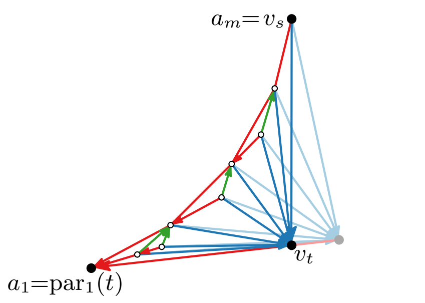
Consider case (iii) of the insertion step: v_l already has an incoming 1-edge and $(v_l, v_r) \notin C_{k-1}$; see Fig. 5c. In this case, (v_k, v_l) is drawn with $\mathrm{slope}(v_k, v_l) = \eta_{k-1} + 1$, so every 1-edge on the contour between v_l and v_r has lower slope than (v_k, v_l). By invariant (I2) and (I4), the 2-edges on the contour between v_l and v_r also cannot have higher slope than η_{k-1}.

Hence, we can use Lemma 2 in each case of the insertion step to show that the drawing remains planar. The second part of the invariant, for each $(v_i, v_j) \in T_n$, $\mathrm{slope}(v_i, v_j) > \mathrm{out}_1(j)$, holds for each vertex but v_k by induction. Since the edges of v_k are planar and v_k is placed vertically above $v_r = \mathrm{par}_2(k)$, all edges $(v_i, v_k) \in T_n$ have positive slope with $x(v_i) > x(v_l)$, so $\mathrm{slope}(v_i, v_k) > \mathrm{out}_1(k)$.

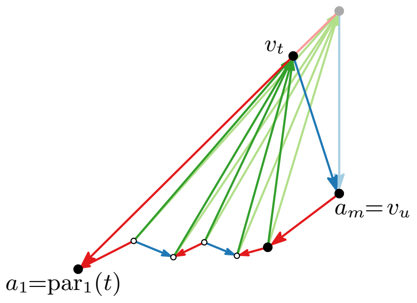
Now, consider the drawing after the shifting step. We shift the vertices between v_r and v_2 one by one on the contour $C_k^{\rightarrow}$ and show that, after each step, the invariant holds. Let v_t be the vertex next to shift, let v_s be its predecessor and let $v_u = \mathrm{par}_2(t)$ be its successor on the contour. Since either $v_s = v_k$ or v_s has been shifted in the previous step, the edge (v_s, v_t) is drawn vertically and the edge (v_t, v_u) has slope $\mathrm{out}_2(t)$. Since $(v_t, \mathrm{par}_1(t))$ lies on the boundary of both domains $\mathrm{dom}(s, t)$ and $\mathrm{dom}(t, u)$, the union of these domains contains all edges incident to v_t and all n-edges of v_t lie in $\mathrm{dom}(t, u)$. We will now show that both domains remain planar and for all $(v_i, v_t) \in T_n$, $\mathrm{slope}(v_i, v_t) > \mathrm{out}_1(t)$.

Consider the domain $\mathrm{dom}(s, t)$; see Fig. 9a. By the Schnyder realizer properties, all edges incident to v_t (except $(v_t, \mathrm{par}_1(t))$) are 2-edges. Let $\mathrm{par}_1(t) = a_1, \ldots, a_m = v_s$ be the path through the neighbors of v_t. All edges on this path are 1-edges or n-edges (otherwise T_2 would not be a tree). By invariant (I3), all 1-edges on this path have slope higher than $\mathrm{out}_1(t)$. By induction and by invariant (I3), for each n-edge (a_{i-1}, a_i) on this path, we have $\mathrm{slope}(a_{i-1}, a_i) > \mathrm{out}_1(i) > \mathrm{out}_1(t)$. As $\mathrm{out}_1(t) > 0$, the path $a_1, \ldots, a_m$ is an x-monotone chain without negative slopes. We can now apply Lemma 2; note that v_t lies *below* the x-monotone chain, so have the mirrored requirements that the positive slopes of the edges of the chain are *greater* than $\mathrm{slope}(a_1, v_t) = \mathrm{out}_1(t)$.

Now, consider the domain $\mathrm{dom}(t, u)$; see Fig. 9b. By the Schnyder realizer properties, all edges incident to v_t (except $(v_t, \mathrm{par}_2(t))$) are n-edges. Let $\mathrm{par}_1(t) = a_1, \ldots, a_m = v_u$ be the path through the neighbors of v_t. All edges on this path are 1-edges or 2-edges (otherwise T_n would not be a tree). If $m = 2$, then there is no n-edge in $\mathrm{dom}(t, u)$.



(a) Maintaining the invariant in $\mathrm{dom}(s,t)$



(b) Maintaining the invariant in $\mathrm{dom}(t,u)$

Figure 9: Shifting vertex v_t while maintaining invariant (I5).

By invariant (I3), we have $\mathrm{out}_1(t) > \mathrm{slope}(a_1, a_2) = \mathrm{out}_1(u)$ and we can move v_t in direction $\overrightarrow{\mathrm{par}_1(t)v_t}$ without changing the embedding. If $m > 2$, we look back on step t of the algorithm. Observe that $a_1, \ldots, a_m$ were the neighbors of v_t on the contour C_{t-1}. Since case (i) and (ii) imply $m = 2$, v_t had to be inserted with case (iii). By construction, the edge $(\mathrm{par}_1(t), v_t)$ was inserted with a higher slope than all 1-edges and, by invariant (I3), also all 2-edges on the contour. Further, as $a_1, \ldots, a_{m-1}$ were removed from the domain, the edges (a_i, a_{i+1}) for $1 \leq i \leq m-2$ were not changed afterward. Thus, we still have $\mathrm{out}_1(t) > \mathrm{slope}(a_i, a_{i+1}), 1 \leq i \leq m-2$. Additionally, by invariant (I3), also $\mathrm{out}_1(t) > \mathrm{slope}(a_{m-1}, a_m)$ holds, so by Lemma 2 we can place v_t above v_u with slope $\mathrm{out}_1(t)$ without crossings. The movement of v_t to the right and the planarity maintain that $\mathrm{slope}(a_i, v_t) > \mathrm{out}_1(t), 2 \leq i \leq m-1$. This establishes invariant (I5).
Invariant (I6). Since v_1 is never moved, it remains at $(0, 0)$. Vertex v_2 is moved $k-3$ times to the right by one column, so it is located at $((k-3)+2, 0) = (k-1, 0)$. As v_k is not placed to the right of v_2 and no vertex is moved to the right by more than one column, the maximum x-coordinate of all vertices is $k-1$. The contour only consists of 1-edges and 2-edges; by invariants (I4) and (I2), their maximum slope is λ_k. Hence, the maximum y-coordinate of all vertices is at most $(k-1)\lambda_k$. □

4 Maximal outerplanar graphs with segments on the grid

A graph is *outerplanar* if it can be embedded in the plane with all vertices on one face (called outerface), and it is *maximal outerplanar* if no edge can be added while preserving outerplanarity. This implies that all interior faces of a maximal outerplanar graph are triangles. Outerplanar graphs have degeneracy 2 [13], that is, every induced subgraph of an outerplanar graph has a vertex with degree at most two. Thus, we find in every maximal outerplanar graph a vertex of degree 2 whose removal (taking away one triangle) results in another maximal outerplanar graph. By this, we gain a deconstruction order that stops with a triangle. Let $G = (V, E)$ be a maximal outerplanar graph and let $\sigma = (v_1, \ldots, v_n)$ be the reversed deconstruction order.

Lemma 3. *The edges of G can be partitioned into two trees T_1 and T_2. Moreover, we can turn G into a planar 3-tree by adding a vertex and edges in the outerface. The additional edges form a tree T_n. The three trees T_1, T_2 and T_n induce a Schnyder realizer.*

Proof. We build the graph G according to the reversed deconstruction order. Let G_k denote the subgraph of G induced by the set $\{v_1, v_2, \ldots, v_k\}$. Further, let G'_k be the graph obtained by adding the vertex v_n and the edges (v_i, v_n) for all $1 \leq i \leq k$ to G. We prove

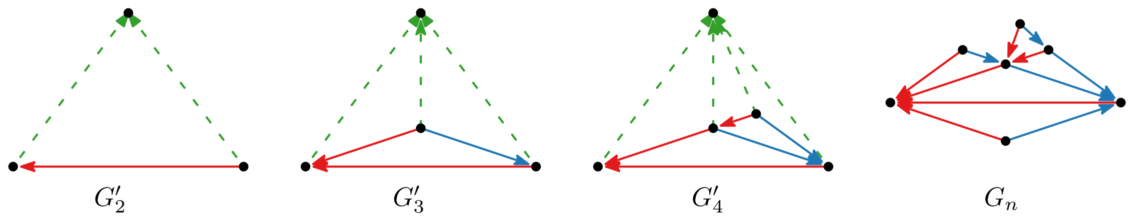


Figure 10: Construction of an maximal outerplanar graph embedded in a planar 3-tree as done in the proof of Lemma 3. The tree T_n is dashed.

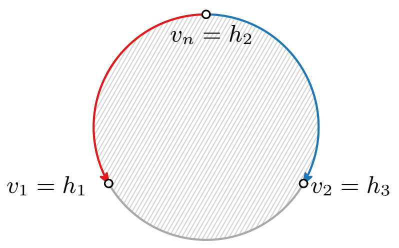
by induction over k that there exists a Schnyder realizer induced by T_1, T_2, T_n for G'_k such that the trees T_1 and T_2 form the graph G_k and G'_k is a planar 3-tree. Note that Felsner and Trotter [11] already proved this lemma without the statement that G'_k is a planar 3-tree.

For the base case $k = 2$, our hypothesis is certainly true; see G'_2 in Fig. 10. Assume our assumption holds for some k. In order to obtain G_{k+1} from G_k, we have to add the vertex v_{k+1} and two incident edges (v_i, v_{k+1}) and (v_j, v_{k+1}). Assume that v_i is left of v_j on C_k. We add (v_i, v_{k+1}) to T_1 and (v_j, v_{k+1}) to T_2. This is safe since we cannot create a cycle in any of the trees. There is another a new edge (v_{k+1}, v_n) in G'_{k+1} which we add to T_n. Again, no cycle can be created. The three outgoing tree edges at v_i form three wedges (unless v_i is the root of T_1). The new ingoing 1-edge (v_i, v_{k+1}) lies in the wedge bounded by the outgoing 2-edge and the outgoing n-edge. For the edge (v_j, v_{k+1}), we can argue analogously. Hence, the three trees induce a Schnyder realizer; see also Fig. 10. Moreover, the graphs G'_k and G'_{k+1} differ exactly by the vertex v_{k+1} that has been stacked into a triangular face of G'_k; thus, since G'_k is a planar 3-tree, so is G'_{k+1}. We now have proven the induction hypothesis for $k+1$. To obtain the statement of the lemma, we take the Schnyder realizer for the graph G'_{n-1} and move the edge (v_1, v_n) from T_n to T_1 and the edge (v_2, v_n) from T_n to T_2. Now T_1 and T_2 form G, and all three trees induce the Schnyder realizer of a planar 3-tree. $\square$

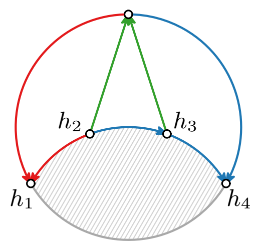
We can now rely on our methods developed for the planar 3-trees. The technique used in Theorem 3 produces a drawing of a planar 3-tree in which two of the trees of the Schnyder realizer are drawn with single-edge segments, whereas the third tree uses as many segments as it had leaves.

Consider a drawing according to Theorem 3 of the planar 3-tree introduced in Lemma 3 that contains the maximal outerplanar graph G as a spanning subgraph. By deleting T_n, we obtain a drawing of G. Note that in this drawing the outer face is realized as an interior face, which can be avoided (if this is undesired) by repositioning v_n accordingly. The Schnyder realizer has $2n - 5$ leaves in total, but $n - 3$ of them belong to T_n. We can assume that T_1 has the smallest number of leaves, which is at most $n/2 - 1$. We need $n - 2$ segments for drawing T_2 (one per edge), and three edges for the triangle v_1, v_2, v_n. In total, we have at most $n/2 - 1 + n - 2 + 3 = 3n/2$ segments. Since the drawing is a subdrawing from our drawing algorithm for planar 3-trees, we get the same area bound as in the planar 3-tree scenario. We summarize our results in the following theorem.

Theorem 4. *Every maximal outerplanar graph admits a straight-line drawing that uses at most $3n/2$ segments on an $O(n) \times O(n^2)$ grid.*



(a) Initial state of the algorithm.



(b) State of the algorithm after processing v_n.

Figure 11: Illustration of the algorithm to draw triangulations with few circular arcs. Hatching indicates the undrawn region.

5 Triangulations and planar graphs with circular arcs

In this section, we present an algorithm to draw a triangulation using few circular arcs. A graph is a *triangulation* (or maximal planar) if every face is a 3-cycle. Our algorithm draws upon ideas for drawing triangulations with line segments by Durocher and Mondal [9] as well as Schulz's algorithm for drawing 3-connected planar graphs with circular arcs [18].

Similar to Schulz [18], a canonical order $v_1, \ldots, v_n$ on the vertices of a triangulation is reversed and used to structure our drawing algorithm. We start by drawing v_1, v_2, and v_n on a circle; see Fig. 11a. We assume that they are placed as shown and hence refer to the arc connecting v_1 and v_2 as *the bottom arc*. The interior of the circle is the *undrawn* region $\mathcal{U}$ which we maintain as a strictly convex shape. The vertices incident to $\mathcal{U}$ are referred to as the *horizon* and denoted $h_1, h_2, \ldots, h_{k-1}, h_k$ in order; see Fig. 11b. We maintain that $h_1 = v_1$ and $h_k = v_2$. Initially, we have $k = 3$ and $h_2 = v_n$. We iteratively take a vertex h_i of the horizon (the latest in the canonical order) to process it, that is, we draw its undrawn neighbors and edges between these, thereby removing h_i from the horizon. Note that h_i will never be the first or last vertex on the horizon (v_1 or v_2).

Invariant. We maintain as invariant that each vertex v (except v_1, v_2, and v_n) has a segment ℓ_v incident from above such that its downward extension intersects the bottom arc strictly between v_1 and v_2. Observe that, since $\mathcal{U}$ is strictly convex, this and h are the only intersection points for ℓ_h with the undrawn region's boundary for a vertex h on the horizon.

Processing a vertex. To process a vertex h_i, we first consider the triangle $h_{i-1}h_ih_{i+1}$: this triangle (except for its corners) is strictly contained in $\mathcal{U}$. We draw a circular arc A from h_{i-1} to h_{i+1} with maximal curvature, but within this triangle; see Fig. 12a. This ensures a plane drawing, maintaining a strictly convex undrawn region. Moreover, it ensures that h_i can "see" the entire arc A.

Vertex h_i may have a number of neighbors that were not yet drawn. To place these neighbors, we dedicate a fraction of the arc A. In particular, this fraction is determined by the intersections of segments v_1h_i and v_2h_i with A; see Fig. 12b. By convexity of $\mathcal{U}$, these intersections exist. If h_{i-1} is equal to v_1, then the intersection for v_1h_i degenerates to v_1; similarly, the intersection of v_2h_i may degenerate to v_2. We place the neighbors in order along this designated part of A, drawing the relevant edges as line segments. This implies that all these neighbors obtain a line segment that extends to intersect the bottom arc, maintaining the invariant. We position one neighbor to be a continuation of segment ℓ_{h_i}, which by the invariant must extend to intersect the designated part of A as well. Two examples of fully drawn graphs are given in Fig. 13. Note that the angular resolution of these drawings decreases very rapidly as n increases. Resolving this issue, e.g. through

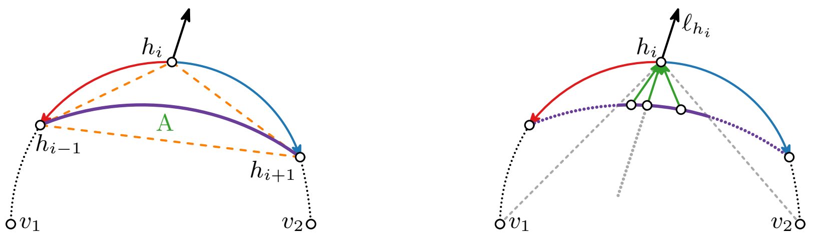


Figure 12: (a) Arc A lies inside the dashed triangle $h_{i-1}h_ih_{i+1}$. (b) Undrawn neighbors of h_i are placed on A, in the section determined by v_1 and v_2. One neighbor is placed to align with ℓ_{h_i} towards a predecessor of h_i.

ensuring a polynomial-size grid, remains an open problem. Another open problem is how many arcs we need if we restrict solutions to a polynomial-size grid.

Complexity. We perform our algorithm using the canonical order induced by the canonical ordering tree in the minimal Schnyder realizer having the fewest leaves; without loss of generality, let T_n be this tree. Recall that T_n has at most $(2n - 5 - \Delta_0)/3$ leaves; since $\Delta_0 \geq 0$, we simplify this to $(2n-5)/3$ for the remainder of the analysis. We start with one circle and subsequently process $v_n, \ldots, v_4$, adding one circular arc per vertex (representing edges in T_1 and T_2) and a number of line segments (representing edges in T_n). Note that processing v_3 has no effect since the edge v_1v_2 is the bottom arc. Counting the circle as one arc, we thus have $n - 2$ arcs in total. At every vertex in T_n, one incoming edge is collinear with the outgoing one towards the root. Hence, we charge each line segment uniquely to a leaf of T_n: there are at most $(2n - 5)/3$ segments.

Thus, the total visual complexity is at most $n - 2 + (2n - 5)/3 = (5n - 11)/3$. In particular, this shows that, with circular arcs, we obtain greater expressive power for a nontrivial class of graphs in comparison to the $2n$ lower bound that is known for drawing triangulations with line segments. Since a triangulation has $e = 3n-6$ edges, we conclude the following.

Theorem 5. *Every triangulation admits a circular arc drawing that uses at most* $5n/3 - 11/3 = 5e/9 - 1/3$ *arcs.*

This bound readily improves upon the result for line segments $(7e/9 - 10/3)$ by Durocher and Mondal [9]. Schulz [18] proved an upper bound of $2e/3$ arcs. The bound above is an improvement on this result, though only for triangulations.

Degrees of freedom. One circular arc has five degrees of freedom (DoF), which is one more than a line segment. Hence, our algorithm with circular arcs uses at most $5 \cdot (5n-11)/3 = (25n - 55)/3$ DoF, even if we disregard any DoF reduction arising from the need to have arcs coincide at vertices. This remains an improvement over the result of Durocher and Mondal [9], using at most $4 \cdot (7n - 10)/3 = (28n - 40)/3$ DoF. The lower bound for line segments $(4 \cdot 2n = 24n/3)$ is lower than what we seem to achieve with our algorithm. However, our algorithm uses line segments rather than arcs to draw the tree T_n. Thus, the actual DoF employed by the algorithm is $5(n - 2) + 4 \cdot (2n - 5)/3 = (23n - 50)/3$, which is in fact below the lower bound for line segments.

4-connected triangulations. We may further follow the rationale of Durocher and Mondal [9] by applying a result by Zhang and He [21]. Using regular edge labelings, they proved that a triangulation admits a canonical ordering tree with at most $\lceil (n + 1)/2 \rceil$

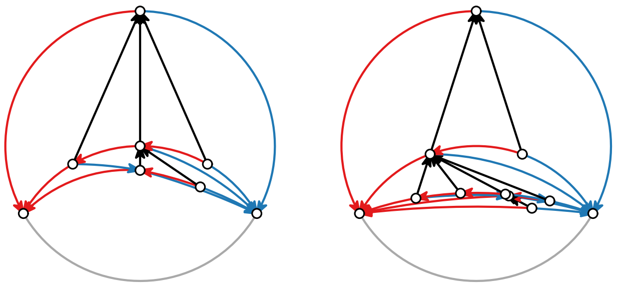


Figure 13: Drawings produced by the algorithm with $n = 7$ (left) and $n = 10$ (right).

leaves [21]. Applying this to our analysis, we find that our algorithm uses at most $n - 2 + \lceil (n+1)/2 \rceil = \lceil (3n-3)/2 \rceil \leq 3n/2 - 1$ arcs.

Theorem 6. *Every 4-connected triangulation admits a circular arc drawing that uses at most $3n/2 - 1 = e/2 + 2$ arcs.*

General planar graphs with circular arcs. The algorithm for triangulations easily adapts to draw a general planar graph G with $n \geq 3$ vertices and e edges. As connected components can be drawn independently, we assume G is connected. We need to only triangulate G, thereby adding $3n - e - 6$ chords. We then run the algorithm described in Theorem 5. Finally, we remove the chords from the drawing. Each chord may split an arc into two arcs, thereby increasing the total complexity by one. From Theorem 5, it follows that we obtain a drawing of G using $(5n/3 - 11/3) + (3n - e - 6) = 14n/3 - e - 29/3$ arcs. Since e is at least $n - 1$, a very rough upper bound is $11n/3 - 26/3$.

Theorem 7. *Every planar graph with $n \geq 3$ admits a circular arc drawing with at most $14n/3 - e - 29/3$ arcs.*

Again, this bound readily improves upon the upper bound for line segments ($16n/3 - e - 28/3$) by Durocher and Mondal [9]. Provided the graph is 3-connected, Schulz's [18] bound of $2e/3 - 1$ is lower than our bound, but only for sparse-enough graphs having $e < 14n/5 - 26/5$. However, there are planar graphs that are not 3-connected with as many as $3n - 7$ edges (one less than a triangulation): there is no sparsity for which planar graphs must be 3-connected and Schulz's bound is lower than our result. In case the original graph G is 4-connected, extending it to a triangulation by adding edges does not violate this property. Repeating the above analysis using the improved bound of Theorem 6 yields us the following result.

Theorem 8. *Every 4-connected planar graph admits a circular arc drawing with at most $9n/2 - e - 7$ arcs.*

Heuristic improvement for planar graphs. We investigate here a simple improvement upon the above by choosing a "good triangulation". However, we must finally conclude that for worst-case bounds, this improvement is only noticeable in small graphs.

For the improvement, we observe that at most two arcs at every vertex continue: one for the horizon and one for T_n. We thus reduce the necessary geometric primitives by picking a single vertex on every face and connecting all chords to that particular vertex. That is, we reduce the increase in complexity caused by removing the temporary chords in the final step of the algorithm. We may even further save on complexity by selecting the same vertex for two adjacent faces, but this is not analyzed here.

Since two chords in a face can lead to an increase in complexity, and since a face of size $|f|$ needs $|f|-3$ chords to be triangulated, we can readily conclude that this saves us $\max\{0,|f|-5\}$ for each face. In other words, only faces with size at least 6 help us reduce the number of circular arcs in the eventual drawing. We define the total reduction $R=\sum_{f\in G}\max\{0,|f|-5\}$; the complexity bound thus becomes $(14n/3-e-29/3)-\sum_{f\in G}\max\{0,|f|-5\}$. Below, we show that $R\geq\max\{0,5n-3e\}$.

For any n and e, we want to find the minimal value of R. Consider an arbitrary connected graph G on n vertices with e edges and consider the sizes of all faces. We turn this into an example with minimal value of R by readjusting the sizes of the faces, "reassigning" chords to different faces. We do so with the following two operations:

1. Let f' and f'' be two faces with $|f''|<5<|f'|$. Reassign a chord from f' to f''. Since $|f''|<5$ holds, $\max\{0,|f''|+1-5\}$ remains 0 and hence R is reduced by 1.
2. Let f' and f'' be two faces with $|f''|\geq|f'|>5$. Reassign a chord from f' to f''. Since both faces have size greater than 5, we know that $\max\{0,|f'|-1-5\}+\max\{0,|f''|+1-5\}=|f'|-5+|f''|-5=\max\{0,|f'|-5\}+\max\{0,|f''|-5\}$ and hence R is unchanged.

A first question would be whether the newly assigned face sizes can always be realized by a graph. We can answer this affirmatively by observing the following procedure to implement the reassignment. Let $f'=f_1,\ldots,f_k=f''$ denote the faces corresponding to a shortest path between f' and f'' in the dual of G. For $i=1$ up to $k-1$, pick an edge (u,v) that is incident to f_i and f_{i+1} and replace (u,v) by (u,w) where $w\neq u$ is a neighbor of v on f_i. This yields again a simple graph with n vertices and e edges, if $|f_i|>3$. As a result of this replacement, $|f_i|$ decreases by one, and $|f_{i+1}|$ increases by one. By assumption, $|f_1|=|f'|>3$ and thus this procedure is valid and has the eventual result that $|f'|$ decreases by one and $|f''|$ increases by one; all intermediate faces maintain their size.

Using the above two operations, we can turn any graph into one having minimal R, for given values of n and e. In particular, applying these operations until they can no longer be applied implies that either: (a) all faces have size at most 5; or (b) one face f has size $|f|>5$ and all other faces have size exactly 5. In case (a), no face contributes to R and thus $R=0$. In case (b), only face f contributes to R and thus $R=|f|-5$. This effectively reduces R to $\max\{0,|f|-5\}$.

Let us further investigate case (b). Double counting of edges along faces gives us $2e=\sum_{f'\in G}|f'|=|f|+5(F-1)$, where $F=2+e-n$ is the total number of faces in G. Hence, we find that $|f|=5n-3e+5$, implying $R=|f|-5=5n-3e$. The resulting bound is $2e-n/3-29/3$. As any planar graph can be drawn trivially with e arcs (or line segments), this new bound is only an improvement if $e>2e-n/3-29/3$, or equivalently, $e<n/3+29/3$. As the graph is connected, we know that $n-1\leq e$ must hold, and thus, we find that $n<16$ must hold for this improvement to be noticeable in the worst-case scenario.

We conclude that this approach does not readily yield an improvement to the complexity bound, but it does provide a heuristic to obtain a lower visual complexity, when a graph has multiple faces of size at least 6.

References

[1] N. Bonichon, B. L. Saëc, and M. Mosbah. Wagner's theorem on realizers. In P. Widmayer, F. T. Ruiz, R. M. Bueno, M. Hennessy, S. Eidenbenz, and R. Conejo, editors, *Proc. 29th Int. Coll. Automata, Languages and Programming (ICALP'02)*, vol-

ume 2380 of *Lecture Notes Comput. Sci.*, pages 1043–1053. Springer, 2002. doi:10.1007/3-540-45465-9_89.

[2] E. Brehm. 3-orientations and Schnyder 3-tree-decompositions. Master's thesis, Freie Universität Berlin, 2000. URL: http://page.math.tu-berlin.de/~felsner/Diplomarbeiten/brehm.ps.gz.

[3] S. Chaplick, K. Fleszar, F. Lipp, A. Ravsky, O. Verbitsky, and A. Wolff. Drawing graphs on few lines and few planes. In Y. Hu and M. Nöllenburg, editors, *Proc. 24th Int. Symp. Graph Drawing Netw. Vis. (GD'16)*, volume 9801 of *Lecture Notes Comput. Sci.*, pages 166–180. Springer, 2016. doi:10.1007/978-3-319-50106-2_14.

[4] S. Chaplick, K. Fleszar, F. Lipp, A. Ravsky, O. Verbitsky, and A. Wolff. The complexity of drawing graphs on few lines and few planes. In F. Ellen, A. Kolokolova, and J. Sack, editors, *Proc. 15th Int. Symp. Algorithms Data Struct. (WADS'17)*, volume 10389 of *Lecture Notes Comput. Sci.*, pages 265–276. Springer, 2017. doi:10.1007/978-3-319-62127-2_23.

[5] H. de Fraysseix and P. O. de Mendez. On topological aspects of orientations. *Discrete Math.*, 229(1-3):57–72, 2001. doi:10.1016/S0012-365X(00)00201-6.

[6] H. de Fraysseix, J. Pach, and R. Pollack. Small sets supporting fary embeddings of planar graphs. In J. Simon, editor, *Proc. 20th Ann. ACM Symp. Theory Comput. (STOC'88)*, pages 426–433. ACM, 1988. doi:10.1145/62212.62254.

[7] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990. doi:10.1007/BF02122694.

[8] V. Dujmović, D. Eppstein, M. Suderman, and D. R. Wood. Drawings of planar graphs with few slopes and segments. *Comput. Geom. Theory Appl.*, 38(3):194–212, 2007. doi:10.1016/j.comgeo.2006.09.002.

[9] S. Durocher and D. Mondal. Drawing plane triangulations with few segments. In M. He and N. Zeh, editors, *Proc. 26th Canad. Conf. Comput. Geom. (CCCG'14)*, pages 40–45. Carleton Univ., 2014. URL: http://www.cccg.ca/proceedings/2014/papers/paper06.pdf.

[10] S. Durocher, D. Mondal, R. I. Nishat, and S. Whitesides. A note on minimum-segment drawings of planar graphs. *J. Graph Algorithms Appl.*, 17(3):301–328, 2013. doi:10.7155/jgaa.00295.

[11] S. Felsner and W. T. Trotter. Posets and planar graphs. *J. Graph Theory*, 49(4):273–284, 2005. doi:10.1002/jgt.20081.

[12] A. Igamberdiev, W. Meulemans, and A. Schulz. Drawing planar cubic 3-connected graphs with few segments: Algorithms and experiments. In E. Di Giacomo and A. Lubiw, editors, *Proc. 23rd Int. Symp. Graph Drawing Netw. Vis. (GD'15)*, volume 9411 of *Lecture Notes Comput. Sci.*, pages 113–124. Springer, 2015. doi:10.1007/978-3-319-27261-0_10.

[13] D. R. Lick and A. T. White. k-degenerate graphs. *Canad. J. Math.*, 22:1082–1096, 1970. doi:10.4153/CJM-1970-125-1.

[14] D. Mondal. *Visualizing graphs: optimization and trade-offs*. phdthesis, University of Manitoba, 2016. URL: http://hdl.handle.net/1993/31673.

[15] D. Mondal, R. I. Nishat, S. Biswas, and M. S. Rahman. Minimum-segment convex drawings of 3-connected cubic plane graphs. *J. Comb. Optim.*, 25(3):460–480, 2013. doi:10.1007/s10878-011-9390-6.

[16] D. J. Rose. On simple characterizations of k-trees. *Discrete Math.*, 7(3):317–322, 1974. doi:10.1016/0012-365X(74)90042-9.

[17] W. Schnyder. Embedding planar graphs on the grid. In D. S. Johnson, editor, *Proc. 1st Ann. ACM-SIAM Symp. Discrete Algorithms (SODA'90)*, pages 138–148. SIAM, 1990. URL: http://dl.acm.org/citation.cfm?id=320191.

[18] A. Schulz. Drawing graphs with few arcs. *J. Graph Algorithms Appl.*, 19(1):393–412, 2015. doi:10.7155/jgaa.00366.

[19] R. E. Tarjan. *Data Structures and Network Algorithms*, chapter 5 – Linking and Cutting Trees, pages 59–70. SIAM, 1983. doi:10.1137/1.9781611970265.ch5.

[20] G. A. Wade and J. Chu. Drawability of complete graphs using a minimal slope set. *Comput. J.*, 37(2):139–142, 1994. doi:10.1093/comjnl/37.2.139.

[21] H. Zhang and X. He. Canonical ordering trees and their applications in graph drawing. *Discrete Comput. Geom.*, 33(2):321–344, 2005. doi:10.1007/s00454-004-1154-y.