Drawing trees and triangulations with few geometric primitives

Gregor Hültenschmidt* Philipp Kindermann* Wouter Meulemans† André Schulz*

Abstract

We define the *visual complexity* of a plane graph drawing to be the number of geometric objects needed to represent all its edges. In particular, one object may represent multiple edges (e.g. you need only one line segment to draw two collinear edges of the same vertex). 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. 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 (crossing number, bends, angular resolution, etc.). In this abstract, we consider the *visual complexity* of planar graphs, that is, the number of simple geometric objects necessary for any drawing. For a number of graph classes, upper and lower bounds are known for segment drawings (allowing only straight-line segments) and arc drawings (allowing circular arcs); the upper bounds are summarized in Table 1. However, these upper bounds do not require the drawings to be on the grid. A trivial lower bound is provided by $\vartheta/2$, where ϑ denotes the number of odd-degree vertices. For triangulations and general planar graphs, a lower bound of $2n + O(1)$ is known [1], where n is the number of vertices.

In this abstract, we look at segment drawings of trees on a grid and at arc drawings of triangulations. We give an algorithm that draws trees on an $O(n^2) \times O(n^{1.58})$ grid using $3n/4$ straight-line segments. This algorithm can be modified to generate drawings with an optimal $\vartheta/2$ segments on a quasi-polynomial grid; so far, no algorithms on the grid have been known. Furthermore, 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 $2n + O(1)$ *lower* bound known for segment drawings [2] and the so far best-known $2e/3 + O(1) = 2n + O(1)$ upper bound for circular arc drawings [4]. A straightforward extension shows that $(14n-3e-29)/3$ arcs are sufficient for general planar graphs with e edges.

Table 1: Upper bounds on the visual complexity for planar graphs. New results are marked with an asterisk. 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	Segment		Arc	
Trees	$\vartheta/2$	[1]	$\vartheta/2$	[1]
3-trees	$2n$	[1]	$11e/18$	[4]
3-connected	$5n/2$	[1]	$2e/3$	[4]
cubic 3-conn.	$n/2$	[3]	$n/2$	[3]
triangulation	$7n/3$	[2]	$5n/3$	*
planar	$16n/3 - e$	[2]	$14n/3 - e$	*

2 Trees with segments on the grid

Heavy paths. Let $T = (V, E)$ be an undirected tree. Our algorithm follows the basic idea of the circular arc drawing algorithm by Schulz [4]. We make use of the *heavy path decomposition* [5] of trees, which is defined as follows. First, root the tree in some vertex r. 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 closest to the root the *top node* of a heavy path and the subtree rooted in the top node the *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 is by 1 larger than the maximum depth of all heavy paths that are connected by an outgoing light edge. Heavy path subtrees of common depth are disjoint.

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

*FernUniversität in Hagen (Germany), `gregorhueltenschmidt@gmx.de [philipp.kindermann | andre.schulz]@fernuni-hagen.de`. Supported by grant SCHU 2458/4-1 (DFG).

†giCentre, City University London (United Kingdom), `wouter.meulemans@city.ac.uk`. Supported by Marie Skłodowska-Curie Action MSCA-H2020-IF-2014 656741.

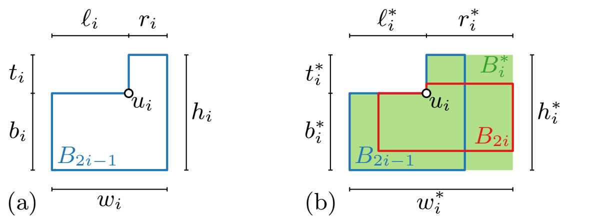

Figure 1: (a) The heavy path box B_i with top node u_i and its lengths; (b) the merged box B_i^* for B_{2i-1} and B_{2i} and its lengths.

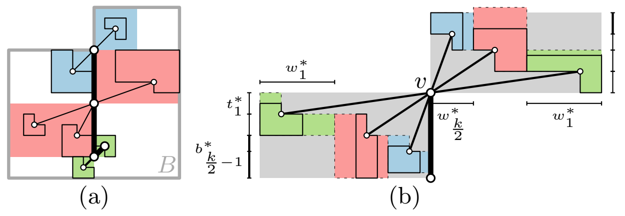

Figure 2: (a) Placement of a heavy path, its box B, and areas for the adjacent heavy path boxes. (b) Placement of the heavy path boxes adjacent to v.

inside an L-shaped box (*heavy path box*) with its top node placed at the reflex angle; see Fig. 1a 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 draw each heavy path subtree of depth 0 as 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 k. When drawing the subtree of a heavy path $\langle v_1, \ldots, v_m \rangle$ of depth $k+1$, we proceed as follows. The last vertex on a heavy path has to be a leaf, so v_m is a leaf. If 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 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. 1b 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. 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, we simply add a dummy box $B_{k+1} = B_k$ that we remove afterwards.

Analysis. We will now calculate the width w_v and the height h_v of this construction. For the width, we have

$$w_v = 2\sum_{i=1}^{k/2} w_i^* = 2\sum_{i=1}^{k/2} \max\{w_{2i-1}, w_{2i}\} \leq 2\sum_{i=1}^{k} w_i.$$

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 $\max_{1\leq j\leq k/2} b_{2j-1}$ in the lower rectangle and $\max_{1\leq h\leq k/2} b_{2h}$ in the upper rectangle. Since we require $b_i \geq t_i$ for each i, we have

$$\begin{aligned} h_v &\leq 2\sum_{i=1}^{k/2} t_i^* + \max_{1\leq j\leq k/2} b_{2j-1} + \max_{1\leq h\leq k/2} b_{2h} \\ &\leq 2\sum_{i=1}^{k} t_i + \sum_{j=1}^{k} b_i \leq \frac{3}{2}\sum_{i=1}^{k} h_i. \end{aligned}$$

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 $b_{2i} \leq b_{2i-1}$ for each i, for the constructed box B we have $b \geq t$.

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 [4], 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 $\lceil 3e/4 \rceil$ arcs on an $O(n^2) \times O(n^{1.58})$ grid.*

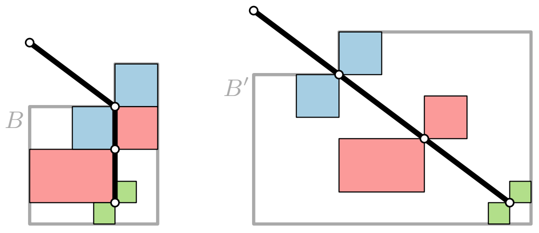

Figure 3: Further improvement on the visual complexity via increasing the size of heavy path boxes. The old box B and the modified box B'.

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. 3. 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.

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

3 Triangulations with circular arcs

As used in previous articles [2, 4], a canonical order $v_1, \dots, v_n$ on the vertices of a triangulation structures our drawing algorithm. However, we use the order in reverse. We start by drawing v_1, v_2, and v_n on a circle; see Fig. 4a. 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 in order, $h_1, h_2, \dots, h_{k-1}, h_k$; 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. Processing a vertex means that we draw its undrawn neighbors and edges between these, thereby removing h_i from the horizon.

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.

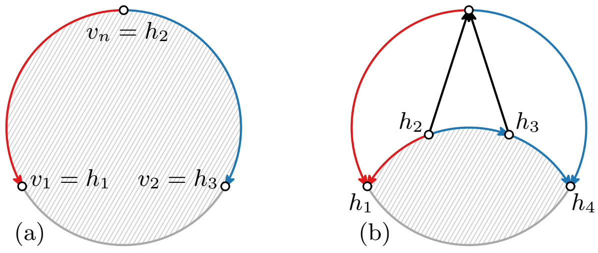

Figure 4: (a) Initial state of the algorithm. (b) State of the algorithm after processing v_n. Hatching indicates undrawn region.

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. 5a. 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. 5b. By convexity of $\mathcal{U}$, these intersections exist. If h_{i-1} is equal to v_1, 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.

Schnyder woods. Using a Schnyder realizer of the triangulation, we decompose the edges into three trees: T_1, T_2, and T_n rooted at v_1, v_2, and v_n, respectively. Following the rationale of Durocher and Mondal [2], we assume w.l.o.g. that T_n has the smallest number of leaves. In particular, the total number of leaves in a minimal realizer is upper bounded by $2n - 5$ [2]. Hence, T_n has at most $(2n - 5)/3$ leaves.

Complexity. We start with one circle and subsequently process $v_n, \dots, v_4$, 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. We charge each line segment 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

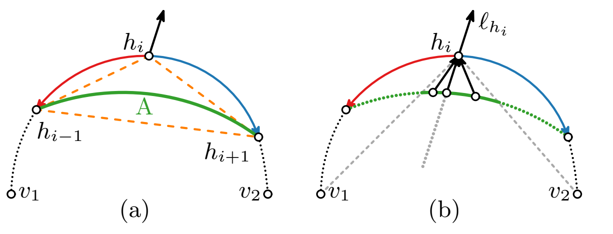

Figure 5: (a) Arc A lies inside the dashed triangle $h_{i-1}h_ih_{i+1}$. (b) Undrawn neighbors of h_i are placed on A, in the section determined by v_1 and v_2. One neighbor is placed to align with ℓ_{h_i} towards a predecessor of h_i.

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.

Degrees of freedom. One circular arc has five degrees of freedom (DoF) which is one more than a line segment. In this light, our algorithm with circular arcs uses roughly $5 \cdot 5n/3 = 25n/3$ DoF. 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 [2], using roughly $4 \cdot 7n/3 = 28n/3$ 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 roughly $5n + 4 \cdot 2n/3 = 23n/3$, which is in fact below the lower bound for line segments.

Theorem 3 *Every triangulation admits a circular arc drawing that uses at most* $(5n-11)/3$ *arcs.*

4 General planar graphs

Simple bound. The algorithm for triangulations easily adapts to draw a general planar graph G with n 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 the previous section, using $(5n-11)/3$ arcs. Finally, we remove the chords from the drawing. Each chord may split an arc into two arcs, thereby increasing the total complexity by one. We obtain a drawing of G using $(5n-11)/3 + 3n - e - 6 = (14n - 3e - 29)/3$ arcs.

Improved bound. We may do slightly better by finding a “good” triangulation, using the property that at most two arcs at every vertex continue: one for the horizon and one for T_n. We reduce the necessary geometric primitives by picking a single vertex on every face and connecting all chords to that particular vertex. (We may even further save on complexity by selecting the same vertex for two adjacent faces.) This saves us an additional $\max\{0, |f|-5\}$ on complexity for a face f of size $|f|$: it needs $|f|-3$ chords, but only two of these can increase the complexity on removal. We can thus obtain a complexity of $(14n-3e-29)/3 - \sum_{f\in G} \max\{0,|f|-5\}$. In other words, this approach reduces the complexity upper bound by $R = \sum_{f\in G} \max\{0,|f|-5\}$. Below, we show that $R \geq \max\{0, 5n-3e\}$, thus implying an overall upper bound of $(14n-3e-29)/3 - \max\{0,5n-3e\}$.

If $3e \geq 5n$, all faces may have size 5 and thus $R = 0$ is possible. We find that $(14n-3e-29)/3 \leq (14n - 5n - 29)/3 = 3n - 29/3$.

Since sparsity increases the upper bound, we construct a worst-case sparse graph to determine the lowest value of R. Consider an arbitrary connected graph G on n vertices with e edges and consider the sizes of all faces. If there there are two faces f' and f'' with $|f''| < 5 < |f'|$, we can reduce R by one by “reassigning” one chord of f' to f''. We ignore here whether the new graph with the given face sizes can actually be realized. We can also reassign a chord from f' to f'' if $|f''| \geq |f'| > 5$ without effecting R. Hence, a worst-case can be obtained if all faces but one have size at most 5; let f denote this one other face. This effectively reduces R to $\max\{0, |f|-5\}$.

Double counting of edges along faces gives us $2e = \sum_{f'\in G} |f'| \leq |f| + 5(F-1)$, where $F = 2 + e - n$ is the total number of faces in G. Hence, we find that $|f| \geq 5n - 3e + 5$. For $R = \max\{0, |f|-5\}$ to be equal to $|f|-5$, we need that $|f| \geq 5$, which is implied by $3e \leq 5n$. If this is indeed the case, then R is equal to $5n-3e$. The upper bound is then $(14n-3e-29)/3 - (5n-3e) = 2e - n/3 - 29/3$. Using $3e < 5n$ we find that this is at most $2(5n/3) - n/3 - 29/3 = 3n - 29/3$.

Theorem 4 *Every planar graph admits a circular arc drawing with at most* $\min\{3n, 14n/3 - e\} - 29/3$ *arcs.*

References

[1] V. Dujmovic, D. Eppstein, M. Suderman, and D. R. Wood. Drawings of planar graphs with few slopes and segments. *Comp. Geom. Theory & Appl.*, 38(3):194–212, 2007.

[2] S. Durocher and D. Mondal. Drawing plane triangulations with few segments. In *Proc. 26th Can. Conf. Comp. Geom. (CCCG'14)*, pages 40–45, 2014.

[3] A. Igamberdiev, W. Meulemans, and A. Schulz. Drawing planar cubic 3-connected graphs with few segments: Algorithms and experiments. In *Proc. 23rd Int. Symp. Graph Drawing & Netw. Vis. (GD'15)*, LNCS 9411, pages 113–124, 2015.

[4] A. Schulz. Drawing graphs with few arcs. *J. Graph Alg. & Appl.*, 19(1):393–412, 2015.

[5] R. E. Tarjan. Linking and cutting trees. In *Data Struct. & Netw. Alg.*, pages 59–70. SIAM, 1983.