Drawing Planar Graphs Using the *lmc*-ordering

Goos Kant

RUU-CS-92-33
October 1992

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

Drawing Planar Graphs Using the *lmc*-ordering

Goos Kant

Technical Report RUU-CS-92-33
October 1992

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN: 0924-3275

Drawing Planar Graphs Using the *lmc*-ordering*

Goos Kant†
Dept. of Computer Science, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, the Netherlands

Abstract

We introduce a new method to optimize the required area, minimum angle and number of bends of planar drawings of graphs on a grid. The main tool is a new type of ordering on the vertices and faces of triconnected planar graphs. With this method linear time and space algorithms can be designed for many graph drawing problems.

- We show that every triconnected planar graph G can be drawn planar on an $(2n-6) \times (3n-6)$ grid with minimum angle larger than $\frac{1}{d-2}$ radians and at most $5n-15$ bends, with d the maximum degree.
- If G has maximum degree four (three), then G can be drawn orthogonal with at most $\lfloor \frac{3n}{2} \rfloor + 3$ (at most $\lfloor \frac{n}{2} \rfloor + 1$) bends on an $n \times n$ grid ($\lceil \frac{n}{2} \rceil \times \lceil \frac{n}{2} \rceil$ grid, respectively).
- Every triconnected planar graph G can be drawn convexly with straight lines on an $(2n-4) \times (n-2)$ grid.

These results give in some cases considerable improvements over previous results, and give new bounds in other cases. Several other results, e.g. concerning visibility representations, are included.

1 Introduction

The problem of "nicely" drawing a graph in the plane has received increasing attention due to the large number of applications [9, 33]. Examples include VLSI layout, algorithm animation, visual languages and CASE tools. Several criteria to obtain

*This work was supported by the ESPRIT Basic Research Actions program of the EC under contract No. 4171 (project ALCOM II). An extended abstract of this paper was presented at the *33th Annual IEEE Symp. on Found. of Comp. Science*, Pittsburgh, 1992.

†Email: goos@cs.ruu.nl

a high aesthetic quality have been established. Typically, vertices are represented by distinct points in a line or plane, and are sometimes restricted to be grid points. (Alternatively, vertices are sometimes represented by line segments [19, 27, 29, 35].) Edges are often constrained to be drawn as straight lines [13, 11, 15, 29, 30] or as a contiguous set of line segments [31, 32, 34, 36] (e.g., when bends are allowed). The objective is to find a layout for a graph that optimizes some cost function, such as area, minimum angle, number of bends, or satisfies some other constraint.

It is well-known ([11, 40]) that every planar graph can be drawn planar with straight lines, and by recent algorithms ([6, 13, 15, 30]), this can be done in linear time and space on a grid of size $O(n) \times O(n)$ (m and n denote the number of edges and vertices of the graph, respectively.) Planar drawings require $\Omega(n^2)$ area in the worst-case [13]. However, a drawback of all these drawing algorithms is that the minimum angle between lines can be very small, which makes the drawing unattractive. In [26] it was shown that every d-planar graph G can be drawn planar such that the minimum angle is at least α^d radians, where $0 < \alpha < 1$ is a constant (a d-planar graph is a planar graph of degree at most d) [31]. However, the proof is non-constructive, and the minimum angle is still quite small. On the other hand, G can be drawn non-planar with straight lines such that the minimum angle is at least $\Omega(\frac{1}{d})$ [12].

A lot of work has been done for the case that all angles are a multiple of $\pi/2$ [12, 31, 32, 34, 36]. These so-called *orthogonal drawings* have numerous important applications in the field of VLSI-design and graphics [31]. Storer presented three heuristic algorithms for minimizing the bends of 4-planar graphs [31]. Tamassia & Tollis presented linear implementations of it [34]. The results are that every biconnected 4-planar (3-planar) graph can be drawn orthogonal and planar on a grid of size at most $n \times n$ such that there are at most $2n + 4$ (at most $n + 2$) bends [31, 34]. (Biconnectivity (triconnectivity) means that deleting any vertex (two vertices) preserves the connectivity.) This bound is tight for biconnected 4-planar graphs [36]. Tamassia also presented an $O(n^2 \log n)$ exact algorithm to draw every *embedded* 4-planar graph on an $n \times n$ grid with a minimum number of bends [32]. An embedding means that the edges around each vertex are given in clockwise order with respect to a planar drawing.

In [38], Tutte showed that every triconnected planar graph can be drawn with straight lines such that every interior face is convex (a so-called *convex drawing*). This criteria is quite essential for the readability of this class of planar graphs, and has therefore gained a lot of interest [4, 37, 38, 39]. Thomassen [37] characterized the class of planar graphs with admit a convex drawing, and Chiba et al. [4] presented a linear drawing algorithm for it. However, the coordinates of the vertices can be reals.

A lot of these drawing algorithms are based on the so-called *st-ordering* of biconnected planar graphs [29, 34]. The *st*-ordering, invented originally in [9] as a first step for testing planarity [24, 2], has also nice features for the dual graph and for visibility representations [29]. Very recent research established that using the

max. degree	tric.	grid	angle	bends	reference
d	no	$(2n-6) \times (3n-6)$	$1/(d-2)$	$5n-15$	this paper
4	yes	$n \times n$	$\pi/2$	$\lfloor \frac{3n}{2} \rfloor + 3$	this paper
	no	$n \times n$	$\pi/2$	$2n+4$	[31, 34]
3	no	$\lceil \frac{n}{2} \rceil \times \lceil \frac{n}{2} \rceil$	$\pi/2$	$\lfloor \frac{n}{2} \rfloor + 1$	this paper
	no	$n \times n$	$\pi/2$	$n+2$	[31, 34]

Figure 1: Summary of important results with bends.

st-ordering, a new and simple characterization of planar graphs can be given [28].

In this paper we introduce a new ordering on the vertices and faces of a triconnected planar graph. We call this ordering the *leftmost canonical ordering*, or the *lmc*-ordering. It generalizes the orderings in [13, 21, 23]. Every *lmc*-ordering is also an *st*-ordering and has direct consequences for the *lmc*-ordering of the dual graph. Using special techniques an *lmc*-ordering can be obtained in linear time and space. This leads to a general framework for drawing triconnected d-planar graphs G on a grid, and implies several further results.

We show that every triconnected planar graph G can be drawn planar with at most $5n-15$ bends on an $(2n-6) \times (3n-6)$ grid with minimum angle $> \frac{1}{d-2}$ radians, with vertices and bends placed on grid points. This seems to be the first practical drawing algorithm, having good bounds on the grid size, number of bends, and on the minimum angle for general planar graphs (see section 7 for the extension to general planar graphs). On the negative side, we prove the following theorem (proof can be found in the appendix):

Theorem 1.1 *Deciding whether a biconnected planar graph can be drawn planar with straight lines with minimum angle $\geq K$ is NP-hard.*

If G is triconnected and 4-planar, then G can be drawn with at most $\lfloor \frac{3}{2}n \rfloor + 3$ bends on an $n \times n$ grid. This improves the best known bound of $2n+4$ considerably in the triconnected case. For any 3-planar graph G we show that G can be drawn with at most $\lfloor \frac{n}{2} \rfloor + 1$ bends on an $\lceil \frac{n}{2} \rceil \times \lceil \frac{n}{2} \rceil$ grid. A nice characteristic is that G has a spanning tree using $n-1$ straight-line edges, and all non-tree edges have at most 1 bend. These bounds match the lower bounds and improve the best bound on the grid size by a factor 4 and the best bound on the number of bends by a factor 2 [31, 34]. For clarity, figure 1 summarizes the results.

We also show that the planar straight-line grid drawing algorithm of De Fraysseix, Pach & Pollack [13] can be implemented in linear time. (In [6] an alternative linear implementation is described, where the planar graph must be triangulated.) For our algorithm, it is sufficient that the input graph is triconnected instead of triangulated. In particular we show that, using the *lmc*-ordering, all interior faces can be made convex. This result outperforms the algorithms of [4, 39] and is not

only of theoretical interest, but also leads to more pleasing pictures (see figure 12). This also gives a new and rather simple proof that every triconnected planar graph admits a convex planar drawing.

We can also use the *lmc*-ordering to construct ***visibility representations*** of a d-planar graph G on a grid of size at most $(2n-5) \times (n-1)$, which drawings seems to be more compact than the drawings of the existing algorithms [19, 27, 29] (see figure 14). All algorithms mentioned above can be implemented to run in linear time and space.

The paper is organized as follows: in section 2 we introduce the *lmc*-ordering, the general drawing framework and some definitions. In section 3 we show how to use this algorithm for drawing d-planar graphs in the grid with minimum angle $> \frac{1}{d-2}$. In section 4 we inspect the orthogonal drawings of triconnected 4-planar graphs. In section 5 we present the results of orthogonal drawings of 3-planar graphs. In section 6 we present the algorithm for the convex drawings. In section 7 we present extensions, combinatorial aspects and optimizations. Section 8 contains some final remarks and open questions.

2 The general framework

In this section we introduce the *leftmost canonical* (*lmc*−) *ordering* for triconnected planar graphs, which will be used in various ways to get better planar drawing algorithms. Let an *embedding* of a triconnected planar graph G be given. (Such an embedding is unique for triconnected planar graphs, and can be constructed in linear time [3].) We first introduce the canonical ordering for G. This ordering generalizes the orderings described in [13, 21, 23]. G_k denotes the subgraph of G, induced on the vertices $v_1, \ldots, v_k$.

Theorem 2.1 *The vertices of a triconnected planar graph G can be ordered in a sequence $v_1, \ldots, v_n$ such that v_2 and v_n are neighbors of v_1 and share a common face, and for every $k, k > 3$:*

1. *either v_k is in the exterior face of G_k and has at least two neighbors in G_{k-1}, which are on the outerface of G_{k-1}. v_k has at least one neighbor in $G - G_k$. G_k is biconnected,*

2. *or there exists an $l \geq 1$ such that $v_k, \ldots, v_{k+l}$ is a chain in the exterior face of G_{k+l} and has exactly two neighbors in G_{k-1}, which are on the outerface of G_{k-1}. Every vertex $v_k, \ldots, v_{k+l}$ has at least one neighbor in $G - G_{k+l}$. G_{k+l} is biconnected.*

Proof: The vertices $v_n, v_{n-1}, \ldots, v_3$ will be defined by reverse induction. Let an embedding of G with an arbitrary outerface be given. Let v_1, v_2 and v_n be arbitrary vertices of the outerface, with v_2 and v_n neighbors of v_1. Let G_{n-1} denote the

subgraph of G after deleting v_n. By triconnectivity of G, the outerface C_{n-1} of G_{n-1} is a cycle.

Let $i < n$ be fixed, and assume that v_k has already been determined for every $k > i$ such that the subgraph G_k induced by $V(G)\backslash\{v_{k+1}, \ldots, v_n\}$ satisfies the conditions of theorem 2.1. Let C_i denote the boundary of the exterior face of G_i. Notice that by this construction, if there are vertices $v \in G_i$ of degree 2, then $v \in C_i$. Notice also that by triconnectivity of G, there are at least 3 vertices $c_\alpha, c_\beta, c_\gamma \in C_i$, having edges to vertices in $G - G_i$.

Assume first that C_i has no interior chords. If there is a maximal chain P of vertices $c_{\alpha_1}, \ldots, c_{\alpha_2}$ ($\alpha_2 \geq \alpha_1$) of degree 2 on C_i, then we take P as the next chain in our ordering, if $v_1, v_2 \notin P$. If this is not the case, then all vertices v of C_i have $deg(v) \geq 3$, except one possible chain of vertices $c_{\alpha_1}, \ldots, c_{\alpha_2}$ ($\alpha_2 \geq \alpha_1$) of degree 2, with $v_1, v_2 \in \{c_{\alpha_1-1}, \ldots, c_{\alpha_2+1}\}$. At least one vertex of $c_\alpha, c_\beta, c_\gamma$, say c_α, is no element of $\{c_{\alpha_1-1}, \ldots, c_{\alpha_2+1}\}$, because this would imply that $G - \{c_{\alpha_1-1}, c_{\alpha_2+1}\}$ is not connected. $c_\alpha \neq v_1, v_2$, $deg(c_\alpha) \geq 3$ and $c_\alpha \notin P$ (if there is a maximal chain P with $v_1, v_2 \in P$). Hence we take c_α as the next vertex in our ordering.

Assume finally that C_i has interior chords. Let $(c_a, c_b), b > a+1$ be a chord such that $b - a$ is minimal. Let also (c_d, c_e) be a chord with $e > d \geq b$ such that $e - d$ is minimal. (When there is no such chord then $(c_a, c_b) = (c_d, c_e)$ and we number the vertices in clockwise order around C_i such that $a = 1 < b = d < e = n + 1$.) By triconnectivity of G, there are vertices c_α, c_β, $a < \alpha < b, d < \beta < e$ on C_i, having edges to vertices, deleted in a step $j > i$. c_α and c_β are not adjacent and by planarity, have no internal chords. Assume w.l.o.g. that $v_1, v_2 \notin \{c_{a+1}, \ldots, c_{b-1}\}$. If $deg(c_\alpha) = 2$, then we can take a maximal chain P of vertices $c_{\alpha_1}, \ldots, c_{\alpha_2}$ of degree 2 (with $c_\alpha \in P$, and $a < \alpha_1 \leq \alpha_2 < b$) as the next chain P in our ordering, otherwise we take c_α as the next vertex in our ordering. It can easily be verified that $G_i - \{c_{\alpha_1}, \ldots, c_{\alpha_2}\}$, if we take chain P, or $G_i - \{c_\alpha\}$, if we take vertex c_α, satisfies the constraints of theorem 2.1. □

This means that starting with an edge (v_1, v_2) one can add in every step either a vertex v_k or a face (which is implied by the chain $v_k, \ldots, v_{k+l}$ and the involved vertices of G_{k-1}). We call this face F_k.

Theorem 2.2 *The canonical ordering can be computed in linear time and space.*

Proof: Let an embedding of the triconnected planar graph G be given. We store for each vertex v and face F an *edge-list*, which is a circular list of edges, in clockwise order of visiting them around each vertex or face. We store for each vertex v and face F a variable *interval*, which can have the value (a), which means: not yet visited, (b), which means: visited once, or (i), which means: visited twice or more and the visited edges form i intervals in the edge-list of this vertex or face (see also [13] for this idea). Every vertex v has a counter *chords*, denoting the number of incident chords of v. We take an arbitrary face as outerface. Let us

call this F_{out} during the algorithm. Assign v_1 with neighbors v_2 and v_n on F_{out}. For each edge e on F_{out}, let F' be the other face, where e belongs to. We mark e in *edge-list*(F') and update *interval*(F') (if $e \neq (v_1, v_2)$). Updating the variable *interval*(F'), when visiting $e = (v, v')$ is done as follows: if *interval*(F') = (a), then *interval*(F') becomes (b). If *interval*(F') = (b), and e is adjacent to the visited edge in *edge-list*(F'), then *interval*(F') becomes (1), else it becomes (2). Finally assume that *interval*(F') = $j, (j \geq 1)$. If the two adjacent edges of e are visited in *edge-list*(F'), then *interval*(F') becomes $(j-1)$. If none of these two edges is visited, then *interval*(F') becomes $(j+1)$, otherwise *interval*(F') remains unchanged. Analog follows for updating *interval*(v). It is clear that *interval*(F') means that the edges already visited and incident to F' are composed of j intervals in the circular order of edges at F'.

We now delete vertex v_n. For each neighbor v' of v_n, we mark edge (v_n, v') in *edge-list*(v') as being visited and update *interval*(v'). For each edge e', which becomes part of F_{out}, we mark e' in *edge-list*(F') as being visited and update *interval*(F'), with F' the other face, where e' belongs to. For each vertex v, which becomes part of F_{out} now, we count the neighbors of v, which are part of F_{out} as well. Except two edges, neighbors of v on F_{out}, these are chords, incident to v, and *chords*(v) is initialized accordingly. If v has a chord to v', then we also increase *chords*(v') by one.

In each step we delete either a vertex or a face. A face F with *interval*(F) = (1) can be the next face in the ordering, otherwise a vertex v (unequal to v_1 and v_2) with *interval*(v) = (b) or (1) and *chords*(v) = 0. (Such a vertex or face exists by theorem 2.1.) When we delete a vertex v or face F, then we update the variables in the same way as when deleting v_n: we update *interval*(v'), if v' is a neighbor of v, or if v' is one of the two neighbors of the new chain P, implied by F. We also visit all edges e' and vertices v', which become part of F_{out} in this step. We update *interval*(F') and *chords*(v') as described above. Notice that if the deleted vertex v had exactly two neighbors, v_a and v_b, and there was an edge (v_a, v_b), then (v_a, v_b) was a chord, which disappears now, i.e., *chords*(v_a) and *chords*(v_b) must be decreased by one. Also when we delete a face F, i.e., we delete a chain $w_1, \ldots, w_p$, and there was an edge (w'_1, w'_p) between the neighbors w'_1 of w_1 and w'_p of w_p, then (w'_1, w'_p) was a chord as well, and disappears now.

Computing *chords*(v) requires $O(deg(v))$ time, when v becomes part of F_{out}. When edge e becomes part of F_{out}, we only have to update *interval*(F'), which can be done in constant time. Deleting a vertex or face can be done constant in the number of deleted edges. Since $\sum deg(v) = 2m$, and $m = O(n)$, this yields a linear time and space algorithm. □

Let there be K steps total, in which we add either a vertex v_k or a face F_k, implied by the added vertices $w_1, \ldots, w_p$. In step 1 and 2 we add vertex v_1 and v_2 and in step K vertex v_n is added. We call the added vertices in step k *new*, the vertices $\in G_{k-1}$ are called *old*. Let $C_{k-1} : v_1 = c_1, c_2, c_3, \ldots, c_r = v_2$ be the outerface

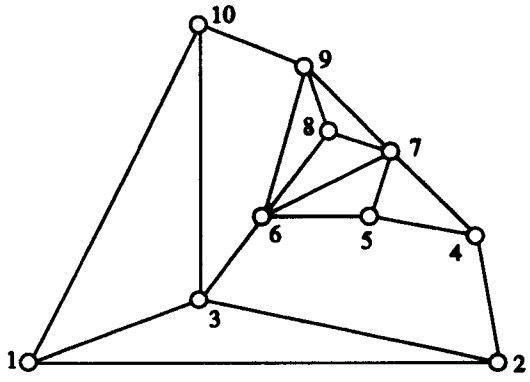


Figure 2: The *lmc*-ordering.

of G_{k-1}, and assume w.l.o.g. that v_1 is drawn most left and v_2 is drawn most right. When adding a face F_k or vertex v_k, let c_i and c_j be two adjacent vertices of v_k (of F_k) on G_{k-1}, with i and j as small and as big as possible, respectively. We call c_i the *leftpoint* and c_j the *rightpoint* of v_k or F_k. The other adjacent vertices c_l $(i < l < j)$ of v_k on C_{k-1} are called *internal* vertices. All these edges to v_k are called *incoming* edges of v_k, the other edges of v_k are called the *outgoing* edges of v_k. $inc(v_k)$ and $out(v_k)$ denote the number of incoming and outgoing edges of v_k, respectively. As a concequence, every vertex has at most one internal outgoing edge.

The general idea is to place the vertices in such a way on the grid that for a vertex v_k, the direction for the incoming edges of v_k is horizontal or downwards. Moreover, we want to maintain the invariant $x(c_1) \leq x(c_2) \leq \cdots \leq x(c_r)$ during each step of the drawing algorithm, hence when adding v_k or F_k, maybe $c_j, \ldots, c_r$ must be "shifted to the right". Of course, also several vertices, not on the outerface, must be shifted to the right as well. Updating all these distances in every step will take us too much time. Therefore, we use *lazy evaluation*: we compute the exact coordinates of a vertex only when we need them. This means that only the exact coordinates of the vertices on the outerface are essential during the insertions. As a first step, we refine the canonical ordering to the *leftmost* canonical ordering, which we will call the *lmc*-ordering from now on. (In section 7.3.1 it is described how to compute the coordinates of the drawing without refining the canonical ordering to the *lmc*-ordering, by applying Chrobak & Payne's drawing technique [6].)

Definition 2.1 *A canonical ordering is a leftmost canonical ordering if at any step k, if v_i (or face F_i) has rightpoint c_α and v_j (or face F_j) has leftpoint c_β (c_α and $c_\beta \in C_{k-1}$), with $\alpha \leq \beta$, then $k \leq i < j$ (or if $v_i, \ldots, v_{i+a}$ and $v_j, \ldots v_{j+b}$ are the new vertices of F_i and F_j, resp., then $k < i + a < j$).*

This means that at any step k during the canonical ordering, when we can add both v_i (or F_i) and v_j (or F_j), then we take this vertex or face, for which the

rightpoint c_l is minimal with respect to l. The algorithm for computing the *lmc*-ordering is as follows: assume that every vertex v_i and face F_i in the *lmc*-ordering has pointers to its rightpoint. We maintain a stack OS of vertices on the outerface from left to right, and we test whether vertex c_i is the rightpoint of a vertex or face, which can be inserted, with c_i the top element of OS. Every time we pop vertices c_i from OS, until we have a vertex c_j on top of OS, for which the consecutive edge of (c_{j-1}, c_j) in *edge-list*(c_j), say (c_j, v), has *rightpoint*$(v) = c_j$. Vertex v or face F (if v belongs to a chain of new vertices $w_1, \ldots, w_p$) is the next one in our ordering. We add v or $w_p, \ldots, w_1$ (in this order) to OS.

Theorem 2.3 *Given a canonical ordering, this algorithm computes in linear time an lmc-ordering.*

Proof: Suppose that this algorithm does not compute an *lmc*-ordering, i.e., at a step k, instead of a vertex v_i (or face F_i) with rightpoint c_α, a vertex v_j (or face F_j) with leftpoint $c_\beta, \beta \geq \alpha$, is inserted. Inspect the moment that c_α is deleted from OS. Then the consecutive edge of $(c_{\alpha-1}, c_\alpha)$ in *edge-list*(c_α) has endpoint v_i, with *rightpoint*$(v_i) = c_\alpha$. Hence v_i (or F_i) can be inserted, which is a contradiction.

Only new vertices will be added to OS, hence every vertex will exactly once be on OS. Since every test can be done in constant time, this yields a linear time algorithm. □

In figure 2 an example of the *lmc*-ordering is given, which will serve as an example for all drawing algorithms in this paper.

In the drawing algorithms, we distinguish the insertcoordinates of v_k (when we insert v_k by the *lmc*-ordering), and the endcoordinates of v_k (in the complete drawing). We introduce a boolean variable *correct*(v), denoting whether $x(v)$, with $v \in C_k$, is recalculated. *shift*(c_j) denotes the value, which must be added to all $x(c_k)$, $j \leq k \leq r$. When we insert v, we set *correct*(v) to *false* and *shift*(v) to zero. When adding a vertex v or face F, we walk on the outerface towards c_1 until we find the first *true* marked *correct*(c_α). Let c_j be the rightpoint of v or F. We walk from c_α to c_j. When visiting c_β $(\alpha < \beta < j)$, we add $\sum_{\alpha<k\leq\beta}$ *shift*(c_k) to $x(c_\beta)$ and set *correct*(c_β) to *true*, since $x(c_\beta)$ is recalculated. We add $\sum_{\alpha<k<j}$ *shift*(c_k) to *shift*(c_j). This approach is correct by the following three lemmas:

Lemma 2.4 *All vertices c_β, $\alpha < \beta \leq r$, have correct(c_β) false.*

Proof: Suppose not. Inspect the first time that a vertex c_γ on the outerface has *correct*(c_γ) = *false* and *correct*$(c_{\gamma+1})$ = *true*. *correct*$(c_{\gamma+1})$ is *true* means that in a step k', we updated $x(c_{\gamma+1})$ and *correct*$(c_{\gamma+1})$, due to the insertion of a vertex or face with leftpoint $c_i, i \geq \gamma + 1$. But *correct*(c_γ) is *false* means that in a step $k > k'$ we added a vertex or face with rightpoint c_γ. This contradicts with the definition of the *lmc*-ordering. □

Lemma 2.5 *$j > \alpha$ holds for rightpoint c_j.*

Proof: Suppose not. *correct*(c_α) = *true* means that in a step $k' < k$, we updated $x(c_\alpha)$, due to the insertion of a vertex or face with leftpoint $c_i, i \geq \alpha$. Adding a vertex v_k (or a face F_k) with rightpoint c_j in step k implies that $i > j$ holds. Since $k' < k$, this contradicts with the definition of the *lmc*-ordering. □

Lemma 2.6 *The total time for visiting the false marked vertices and updating shift$(v), x(v)$ and correct(v) for all vertices is $O(n)$.*

Proof: When we insert a vertex or face in step k, with leftpoint c_i and rightpoint c_j, extra time is required for walking towards c_1 to find the first true marked *correct*(c_α). All *correct*-values of the vertices $c_\alpha, \ldots, c_{j-1}$ are marked true after we visit them. If in a step $l, l > k$, *correct*$(c_\beta), \alpha \leq \beta \leq j-1$, becomes false again, then a vertex (or face) with rightpoint c_β is added, with $\beta \leq i$. Contradiction with the *lmc*-ordering, hence every *correct*(c_β) is at most once false and becomes true after visiting c_β again. Updating requires constant time, hence the total time for visiting the false marked vertices and updating *shift*$(v), x(v)$ and *correct*(v) for all vertices is $O(n)$. □

These three lemmas show that we can compute the *new* coordinates of the vertices on the outerface correctly, when we insert a vertex v_k or face F_k. Let $P(v) = (x_{\text{insert}}(v), y_{\text{insert}}(v))$ be the coordinates of v at the time of insertion of v. Computing the final coordinates is done as follows: we visit all vertices and faces in *reverse lmc*-order, and set initially *shift*$(v) = 0$ for every vertex v. When visiting vertex v then (depending on the algorithms in the next sections) we assign *shift*(v) to the shift of all its internal vertices and add at least *shift*(v) to its rightpoint. Then we add *shift*(v) to $x_{\text{insert}}(v)$, leading to the final coordinates of v. All this work can be done in linear time totally. We call this method the *shift-method.* This method will serve as a general framework for planar graph drawings on a grid. The idea of shifting vertices is widely used, e.g., in the grid drawing algorithm of Chrobak & Payne [6]. This technique will be explained in section 7.3.1.

In the next sections we will fill in the details for the different classes of planar graphs, which leads to several linear time algorithms and better (and sometimes new) bounds for the number of bends, minimum angle and grid size, and for convex drawings.

3 Drawing d-planar graphs

In this section we use the general framework, introduced in section 2, to draw any triconnected d-planar graph G on an $(2n-6) \times (3n-6)$ grid such that there are at most $5n - 15$ bends and minimum angle $> \frac{1}{d-2}$ radians. All vertices and

bendcoordinates will be placed on grid points only. Every edge will have at most 3 bends and length $O(n)$. Let the *lmc*-ordering be given. Let $out(v) = p$, then we place the p outgoing edges from v, say to $w_1, \ldots, w_p$ (in this order from left to right), such that they will go through the points $(x(v) - \lfloor \frac{p-1}{2} \rfloor + i - 1, y(v) + \lceil \frac{p-1}{2} \rceil), 1 \leq i \leq p$. We call these points the *out-points* of the corresponding edges. Let $inc(v) = p'$, with $p' \geq 3$, then we place the incoming edges to v, say from $w_1, \ldots, w_{p'}$ (in this order from left to right), such that they will go through the points $(x(v) - \lfloor \frac{p'-3}{2} \rfloor, y(v) - i + 1)$, if $1 \leq i < \lceil \frac{p'}{2} \rceil$, through $(x(v), y(v) - \lceil \frac{p'-3}{2} \rceil)$, if $i = \lceil \frac{p'}{2} \rceil$, and through $(x(v) + \lceil \frac{p'-3}{2} \rceil, y(v) - p' + i)$, otherwise. These points are called the *inc-points* of the corresponding edges. See figure 3(a) for the corresponding drawing of the inc- and outpoints. Every edge has one *out-point* and one *inc-point*, which can be connected by a vertical and horizontal line, which gives one extra bend, hence at most 3 bends in every edge. Every edge (u, v) has the form: from u to the next out-point of u, then ≥ 0 steps in vertical direction, then ≥ 0 steps in horizontal direction to the next inc-point of v, and then finally to v.

Vertex c_i has out-point $(x(c_i) + \lceil \frac{out(c_i)-1}{2} \rceil, y(c_i) + \lceil \frac{out(c_i)-1}{2} \rceil)$, and vertex c_{i+1} has out-point $(x(c_{i+1}) - \lfloor \frac{out(c_{i+1})-1}{2} \rfloor, y(c_{i+1}) + \lceil \frac{out(c_{i+1})-1}{2} \rceil)$. To preserve planarity, let $di_X(c_i, c_{i+1}) = \lceil \frac{out(c_i)-1}{2} \rceil + \lfloor \frac{out(c_{i+1})-1}{2} \rfloor$, then $x(c_{i+1}) \geq x(c_i) + di_X(c_i, c_{i+1}) + 1$ must hold, if $y(c_i) = y(c_{i+1})$. Otherwise assume $y(c_{i+1}) > y(c_i)$ and, hence, there is an outgoing edge from c_i to c_{i+1}, thus $x(c_{i+1}) \geq x(c_i) + di_X(c_i, c_{i+1})$ must hold.

In Y-direction, let v_k have incoming edges from $w_1, \ldots, w_p$. The out-points of w_i are on height $y(w_i) + \lceil \frac{out(w_i)-1}{2} \rceil$, and the inc-points of v_k are on height $y(v_k) - \lceil \frac{inc(v_k)-3}{2} \rceil$, thus $y(v_k) \geq \max_{1 \leq i \leq p} \{y(w_i) + \lceil \frac{out(w_i)-1}{2} \rceil\} + \lceil \frac{inc(v_k)-3}{2} \rceil$ must hold. Notice that if $y(w_i) > y(w_{i-1})$, then $y(w_i) \geq y(w_{i-1}) + \lceil \frac{out(w_{i-1})-1}{2} \rceil$ holds. Hence w_i $(1 < i < p)$ has one outgoing edge or $y(w_i) < y(w_{i+1})$ or $y(w_i) < y(w_{i-1})$ holds. Let $di_Y(v_k) = \max\{1 + \lceil \frac{inc(v_k)-3}{2} \rceil, \lceil \frac{out(w_1)-1}{2} \rceil, \lceil \frac{out(w_p)-1}{2} \rceil\}$, then $y(v_k) \geq \max_{1 \leq i \leq p} \{y(w_i)\} + di_Y(v_k)$ must hold to preserve planarity. The complete algorithm to compute the insertcoordinates is now as follows:

$P(v_1) := (0, 0); P(v_2) := (di_X(v_1, v_2), 0);$
for $k := 3$ **to** K **do**
  update $x(c_i)$ and $shift(c_j)$;
  assume we add $w_1, \ldots, w_p$ from c_i to c_j, with $p \geq 1$;
  if $p = 1$: let $u_1, \ldots, u_r$ be the incoming edges of w_1;
    $x(v_k) :=$ x-coordinate of last out-point of $u_{\lceil \frac{r}{2} \rceil}$;
    $y(v_k) := \max_{1 \leq i \leq r} \{y(u_i)\} + di_Y(w_1)$;
  if $p > 1$: $y(w_1) := \cdots := y(w_p) := \max\{y(c_i) + \lceil \frac{out(c_i)-1}{2} \rceil, y(c_j) + \lceil \frac{out(c_j)-1}{2} \rceil\}$;
    $x(w_1) := x(c_i) + di_X(w_1, c_i)$;
    $x(w_{i+1}) := x(w_i) + 1 + di_X(w_i, w_{i+1})$; $(1 \leq i < p)$
  $shift(c_j) := \max\{shift(c_j), x(c_i) + di_X(c_i, c_j) - 1 + \sum_{1 \leq i \leq p}(out(w_i) + 1) - x(c_j)\}$;
rof

In this algorithm only the coordinates of the vertices are given. The edges are placed on the grid via the inc- and outpoints. For the out-points of v_k we use two pointers, say l_k and r_k. l_k starts at $(x(v_k) - \lfloor \frac{out(v_k)-1}{2} \rfloor, y(v_k) + \lceil \frac{out(v_k)-1}{2} \rceil)$, and r_k starts at $(x(v_k) + \lceil \frac{out(v_k)-1}{2} \rceil, y(v_k) + \lceil \frac{out(v_k)-1}{2} \rceil)$. When a vertex v_j is added with leftpoint (rightpoint) v_k, then edge (v_j, v_k) goes via r_k (via l_k) and $x(r_k)$ is decreased by one ($x(l_k)$ is increased by one, respectively).

The final coordinates are computed by visiting the vertices and faces in reverse *lmc*-ordering, and computing and adding the corresponding shifts.

for all vertices v_i, $shift(v_i) := 0$;
for $k := K$ **downto** 2 **do**
  if $p = 1$: let $u_1, \ldots, u_r$ be the incoming edges of w_1;$(u_1 = c_i, u_r = c_j)$
    $x(w_1) := x_{\text{insert}}(w_1) + shift(w_1)$;
    $shift(u_{\lceil \frac{r}{2} \rceil}) := \ldots := shift(u_{r-1}) :=$
      $\max\{0, x(w_1) - x(c_i) - shift(c_i) - di_X(c_i, w_1)\}$;
  if $p > 1$: $x(w_i) := \ldots := x_{\text{insert}}(w_i) + \sum_{1 \le l \le i} shift(w_l)$; $(1 \le i \le p)$;
  $shift(c_j) := \max\{shift(c_j), x(c_j) - x(w_p) - di_X(w_p, c_j)\}$;
rof

Let d be the maximum degree of all vertices. The maximum number of incoming (outgoing) edges of all vertices is at most $d-1$ (at most $d-2$, resp.). Let $w_1, \ldots, w_{d-2}$ be the outgoing edges (in this order), then the smallest angle α is between the outgoing edges to w_{d-3} and w_{d-2} of v_k, visiting the out-points $(x(v_k) + \lceil \frac{d-3}{2} \rceil - 1, y(v_k) + \lceil \frac{d-3}{2} \rceil)$ and $(x(v_k) + \lceil \frac{d-3}{2} \rceil, y(v_k) + \lceil \frac{d-3}{2} \rceil)$. Let $k = \lceil \frac{d-3}{2} \rceil$, then $\alpha = \frac{\pi}{4} - \arctan(\frac{k-1}{k}) = \arctan(\frac{1}{2k-1}) = \frac{1}{2k-1} - \frac{1}{3}\frac{1}{(2k-1)^3} + \frac{1}{5}\frac{1}{(2k-1)^5} - \frac{1}{7}\frac{1}{(2k-1)^7} + \ldots > \frac{1}{d-2}$,[1] which proves the following lemma:

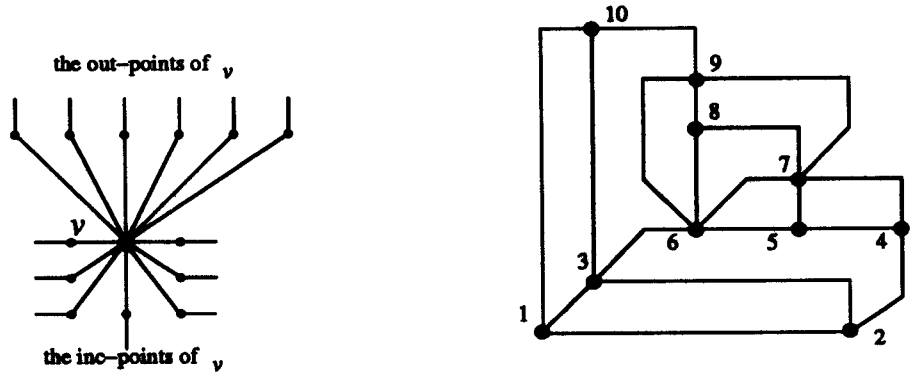
Lemma 3.1 *The smallest angle has size* $> \frac{1}{d-2}$.

Lemma 3.2 *The gridsize is at most* $(2n-6) \times (3n-6)$.

Proof: For the X-direction holds in every step that $x(c_{i+1}) = x(c_i) + \lceil \frac{out(c_i)-1}{2} \rceil + \lfloor \frac{out(c_{i+1})-1}{2} \rfloor$ on the outerface, if $y(c_{i+1}) \neq y(c_i)$, and otherwise $x(c_{i+1}) = x(c_i) + \lceil \frac{out(c_i)-1}{2} \rceil + \lfloor \frac{out(c_{i+1})-1}{2} \rfloor + 1$. But if $y(c_{i+1}) = y(c_i)$, then (c_i, c_{i+1}) is not an outgoing edge of any vertex. Let us call (c_i, c_{i+1}) *unmarked*. Counting leads to a horizontal distance of at most $\sum_{1 \le i < n} \lceil \frac{out(v_i)-1}{2} \rceil + \lfloor \frac{out(v_{i+1})-1}{2} \rfloor + |\textit{unmarked edges}| = \sum_{1 \le i < n} (out(v_i) - 1) = 2n - 6$.

Adding a vertex v_k requires more increase in Y-direction per vertex than adding a face, hence assume we add a vertex v_k in every step. Let the incoming edges of vertex v_k come from $u_1, \ldots, u_p$, then $y(v_k) \ge \max_{1 \le i \le p}\{y(u_i)\} + \max\{1 + \lceil \frac{inc(v_k)-3}{2} \rceil, \lceil \frac{out(u_1)-1}{2} \rceil, \lceil \frac{out(u_p)-1}{2} \rceil\}$ holds. The increase for every vertex v_k during the insertions

[1] With thanks to Maarten Pennings.



(a) Example of inc– and out–points.

(b) Drawing the graph of figure 2.

Figure 3: Drawing the graph of figure 2 with bends.

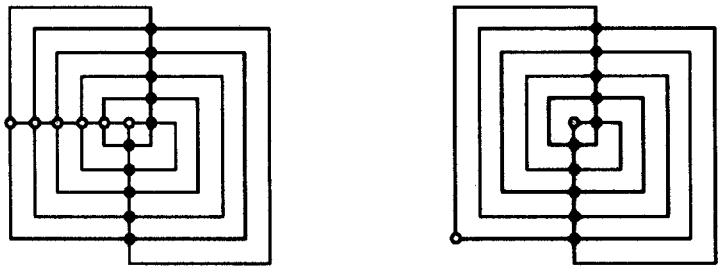
is at most $1 + \lceil \frac{inc(v_k)-3}{2} \rceil + \lceil \frac{out(v_k)-1}{2} \rceil \leq \lfloor \frac{deg(v_k)}{2} \rfloor$. Summarizing this for all vertices leads to a total distance in Y-direction of at most $3n - 6$ units. □

Lemma 3.3 *There are at most* $5n - 15$ *bends. Every edge has at most 3 bends and length* $O(n)$.

Proof: All outgoing edges of vertex v, except the one going straight upwards, requires one bend in worst-case to go in vertical direction. We assign these bends to the insertion step of v. Adding a face requires less bends per vertex than adding a vertex, so assume we only add vertices v_k. If $inc(v_k) = 2$, and assume $y(c_i) \geq y(c_j)$, then there will come at most 1 bend in (c_i, v_k) and 2 bends in (c_j, v_k). In each edge, one bend was already assigned to the insertion step of c_i and c_j, hence adding v_k with $inc(v_k) = 2$ requires at most one bend for the incoming edges. If $inc(v_k) \geq 3$, then at most $2 \cdot inc(v_k) - 4$ extra bends are required for the incoming edges. Edge (v_1, v_2) requires no bends. Counting this leads to totally at most $5n - 15$ bends. Every edge goes at most once vertical and once horizontal, hence requiring 3 bends in worst-case and by lemma 3.2, has length $O(n)$. □

Corollary 3.4 *There is a linear time and space algorithm to draw a triconnected d-planar graph planar on an* $(2n-6) \times (3n-6)$ *grid with at most* $5n - 15$ *bends and minimum angle* $> \frac{1}{d-2}$, *where every edge has at most 3 bends and length* $O(n)$.

In figure 3(b) the drawing of the graph of figure 2 is given.



(a) the graph G_5 (b) the graph G'_5

Figure 4: Lowerbound of $\frac{4}{3}(n-1)+3$ bends.

4 Orthogonal Drawings of 4-Planar Graphs

In this section we consider the problem of drawing a triconnected 4-planar graph G on a rectilinear grid with orthogonal edges, i.e., the edges are polygonal chains of horizontal and vertical segments. The vertices are represented by points. This problem has important applications in VLSI-design, and has received therefore a lot of attention during the last years [31, 32, 34, 36]. Using a variant of theorem 2 of [36], we can obtain the following lowerbound:

Theorem 4.1 *There are embedded triconnected 4-planar graphs G_n with $3n+1$ vertices and $6n+1$ edges, for which any layout requires at least $4n+2$ bends.*

Proof: Consider the triconnected planar graph G_n with $3n+1$ vertices, and its layout in figure 4(a), which has $4n+3$ bends. Notice that there are no bends in the edges between two white vertices. Deleting these edges and deleting the white vertices of degree 2 while connecting the incident edges leads to a biconnected planar graph G'_n with $2n+2$ vertices (see figure 4(b)). It is shown in corollary 4 in [36] that the shown layout in figure 4(b) is best possible with respect to the minimum number of bends, which is $4n+2$. If there was a layout for G_n with fewer than $4n+2$ bends, then there was a better layout of G'_n with fewer than $4n+2$ bends, which is a contradiction with corollary 4 of [36]. □

Let an *lmc*-ordering of G be given. $deg(v) \leq 4$, thus every vertex v has at most 2 outgoing edges. We introduce a variable $mark(v_k)$ for each vertex v_k, which is important when adding v_k to G_{k-1}. If v_k is the rightpoint of 2 vertices, or if v_k is the rightpoint of a vertex and has an internal outgoing edge, then we set $mark(v_k)$ = *left*, otherwise we set $mark(v_k)$ = *right*. There are four directions to connect an edge at v, viz., *left*, *right*, *up* and *down* of v, denoted by $l(v), r(v), u(v)$ and $d(v)$, respectively. A direction is called *free* if there is no edge connected in that direction

of v yet. We add v_k to G_{k-1} such that $d(v_k)$ is not free in G_k. Let c_i and c_j be the left- and rightpoint of v_k. We connect (c_i, v_k) at $r(c_i)$, if $r(c_i)$ is free, otherwise at $u(c_i)$, if $u(c_i)$ is free, otherwise at $l(c_i)$. The opposite direction is followed for c_j. We want to add v_k such that when $mark(v_k) = left$, $l(v_k)$ and $u(v_k)$ are both free after addition, since this implies only a few bends. Similar we want to have both $d(v_k)$ and $r(v_k)$ to be free, when $mark(v_k) = right$. The algorithm, which tries to achieve this as much as possible, can be described as follows:

edge (v_1, v_2) via $d(v_1)$ and $d(v_2)$;
for $k := 2$ **to** $K - 1$ **do**
- **if** we add v_k with $inc(v_k) = \{c_i, c_l, c_j\}$ **then**
  (c_i, v_k) via $l(v_k)$; (c_l, v_k) via $d(v_k)$; (c_j, v_k) via $r(v_k)$;
- **if** we add v_k with $inc(v_k) = \{c_i, c_j\}$ **then**
  if $mark(v_k) = left$ or ($l(c_j)$ free **and** $r(c_i)$ not free) **then**
  (v_k, c_i) via $d(v_k)$ and (v_k, c_j) via $r(v_k)$
  else
  (v_k, c_i) via $l(v_k)$ and (v_k, c_j) via $d(v_k)$;
- **if** we add $w_1, \ldots, w_p$ $(p > 1)$ from c_i to c_j **then**
  if $r(c_i)$ is free **then** (w_1, c_i) via $l(w_1)$ **else** via $d(w_1)$;
  for $l := 2$ **to** p **do**
  if $d(w_{l-1})$ is free **then** (w_{l-1}, w_l) via $d(w_{l-1})$ **else** via $r(w_{l-1})$;
  if $mark(w_l)$ is $left$ **then** (w_{l-1}, w_l) via $d(w_l)$ **else** via $l(w_l)$;
  if $l(c_j)$ is free **then**
  (w_{p-1}, w_p) via $d(w_p)$ and (w_p, c_j) via $r(w_p)$
  else
  (w_{p-1}, w_p) via $l(w_p)$ and (w_p, c_j) via $d(w_p)$

rof;
edges from $c_i, c_\alpha, c_\beta, c_j$ to v_n via $u(v_n), l(v_n), d(v_n)$ and $r(v_n)$, resp.;

From these connections we have to come to a correct planar drawing, such that if (v_i, v_j) goes via $r(v_i)$ and $l(v_j)$, then (v_i, v_j) is a straight line, i.e., $x(v_j) > x(v_i)$ and $y(v_j) = y(v_i)$. When (v_i, v_j) goes via $u(v_i)$ and $l(v_j)$, then one bend b must be introduced at place $P(b) = (x(v_i), y(v_j))$ and $x(b) < x(v_j)$ and $y(b) > y(v_i)$ must hold. (Similar when (v_i, v_j) goes via $u(v_i)$ and $r(v_j)$.) Finally, when (v_i, v_j) goes via $d(v_i)$ and $d(v_j)$, then 2 bends b_1 and b_2 are introduced. (Similar when (v_i, v_j) goes via $r(v_i)$ and $r(v_j)$ or via $l(v_i)$ and $l(v_j)$.)

To calculate the coordinates of the vertices and bends we use instead of the *shift*-method (which may also lead to a linear implementation) a new approach: We define 2 directed graphs X and Y. Every time when we add a vertex v_k or face F_k, we add all new vertices and bends as a node to both X and Y, and we add some directed edges. A directed edge (v_i, v_j) in X (in Y) means that $x(v_i) < x(v_j)$ ($y(v_i) < y(v_j)$, resp.) must hold. If (v_i, v_j) and (v_j, v_i) are both present in X (in Y), then $x(v_i) = x(v_j)$ ($y(v_i) = y(v_j)$, resp.) must hold. When we add a vertex v_k or face F_k, then we know the free directions of the vertices on the outerface, hence

we know precisely where bends are required, and the involved (in)equalities. Since G is planar, X and Y are planar as well and constructed during the *lmc*-ordering. Afterwards we construct from X and Y two new directed acyclic graphs X' and Y'. Every node V' in X' (in Y') corresponds with a set of nodes $\{v_a, \ldots, v_b\}$ of X (of Y), which must have the same x-coordinate (y-coordinate). A directed edge (V', V'') in X' (in Y') means that for every vertex $v' \in V'$ and $v'' \in V''$: $x(v') < x(v'')$ ($y(v') < y(v'')$) must hold. Since X and Y are planar, X' and Y' are planar as well, and thus have only $O(n)$ vertices and edges. We now want to compute a topological ordering for the vertices of both X' and Y', whose numbering corresponds with the coordinates in X- and Y-direction. X' and Y', however, need not to be a lattice, i.e., there can be ≥ 2 vertices initially in X' (in Y') without incoming or outgoing edges.

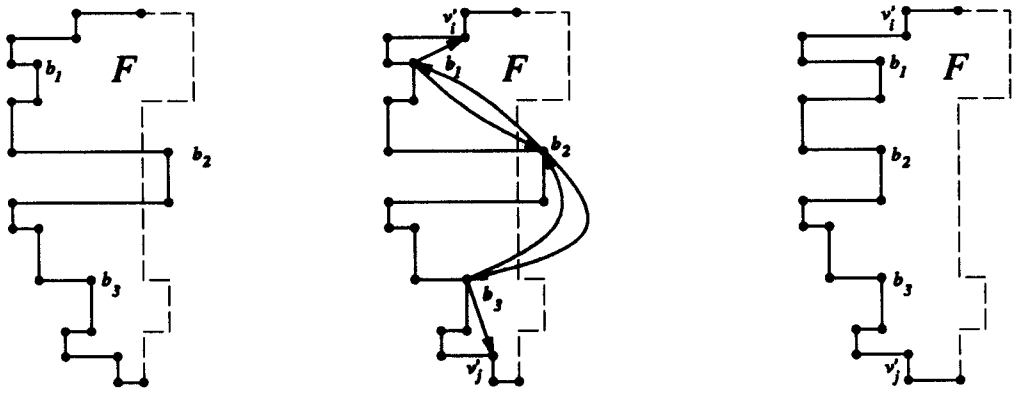
More precisely, this happens only when we have edges with 2 bends. E.g., assume edge (v_i, v_j) goes via $r(v_i)$ and $r(v_j)$, hence via 2 bends, b_1 and b_2, then the corresponding node V' in X', with $b_1, b_2 \in V'$, has no outgoing edges. Using an arbitrary topological ordering may lead to a numbering with V' numbered maximal, which give bends in (v_i, v_j) in the orthogonal drawing of G (see figure 5(a)). Similar when (v_i, v_j) goes via $l(v_i)$ and $l(v_j)$, then node V' in X', with $b_1, b_2 \in V'$ has no incoming edge. When (v_i, v_j) goes via $d(v_i)$ and $d(v_j)$, then node V' in Y', with $b_1, b_2 \in V'$, has no incoming edge.

Let us call the 2 bends in an edge (v_i, v_j), going from $r(v_i)$ to $r(v_j)$ *rightbends*. Correspondingly *leftbends* and *downbends* are defined. We now visit all faces F of G. Let $b_1, \ldots, b_r$ be the rightbends of F (in this order), belonging to different edges (see figure 5(a)). Let b_1 be adjacent to $v_i \in F$, and let there be a directed edge (V_i, V_i') in X', such that $v_i \in V_i$ and $v_i' \in V_i'$ with $v_i' \in F$. Similar let b_r be adjacent to $v_j \in F$, and let there be a directed edge (V_j, V_j') in X', such that $v_j \in V_j$ and $v_j' \in V_j'$ with $v_j' \in F$. We add edges $(b_\alpha, b_{\alpha+1})$ and $(b_{\alpha+1}, b_\alpha)$ to X $(1 \leq \alpha < r)$, and we add the edges (b_1, v_i') and (b_r, v_j') to X (see figure 5(b)). Adding these edges has the effect that all rightbends will have the same x-coordinate, which is smaller than all remaining x-coordinates of these vertices in F, having no path to or from $b_1, \ldots, b_{r-1}$ in X (see figure 5(c)).

We apply the same method to the leftbends and downbends in each face, and update X' and Y'. We compute a topological ordering for X' and Y'. We start with this node of X', containing a bend of (v_1, v_n), and this node of Y', containing a bend of (v_1, v_2). Let V_i' and V_j'' denote the topological ordering of the vertices of X' and Y', then vertex $v_a \in V_i'$ and $v_a \in V_j''$ has coordinates $P(v_a) = (i, j)$. One easily verifies the following result (see also figure 5).

Lemma 4.2 *The topological ordering of X' and Y' corresponds with a planar orthogonal drawing of G.*

Lemma 4.3 *The number of bends is at most $\lfloor \frac{3}{2}n \rfloor + 3$. Every edge has at most 3 bends.*



(a) Marking the rightbends.

(b) Adding the edges between the rightbends and to 2 other vertices in the face.

(c) The resulting orthogonal drawing is planar.

Figure 5: Adding extra edges to maintain planarity.

Proof: Assume first that we add p vertices $w_1, \ldots, w_p$ from c_i to c_j. For each vertex w_i we use its $d(w_i)$ connection and either $l(w_i)$ or $r(w_i)$. Using $d(w_i)$ always implies one extra bend in the corresponding edge ($1 < i < p$). Using $l(w_i)$ or $r(w_i)$ implies no extra bends. If the edge (c_i, w_1) goes via $u(c_i)$ and $d(w_1)$, then one extra bend is required for the outgoing edge via $l(c_i)$ when $mark(w_1) = right$. Similar when edge (c_j, w_p) goes via $u(c_j)$ and $d(w_p)$ and $mark(w_p) = left$. This gives at most p bends when adding a face F_k with p new vertices.

When we add one vertex, v_k, then it easily follows by case analysis that this implies at most 2 bends if $inc(v_k) = 3$. If $inc(v_k) = 2$, then in at most one incoming edge a bend is required. If $mark(v_k) = left$ and $r(c_i)$ and $u(c_j)$ are both free, then the connections $l(v_k)$ and $d(v_k)$ are used, hence this gives 1 extra bend for the outgoing edge of v_k via $r(v_k)$. Similar when $mark(v_k) = right$ and $u(c_i)$ and $l(c_j)$ are free. In both cases we assign the cost of this extra bend to the insertion step of v_k. Adding v_n implies at most 4 bends. $mark(v_1) = right$, but v_1 has 3 outgoing edges (except the edge to v_2 via $d(v_1)$), hence we use $l(v_1)$, which requires one extra bend, which we assign to the insertion step of v_1. Similar for using $r(v_2)$, hence starting with (v_1, v_2) implies 4 bends. Summarizing this leads to the following table:

step	# vertices	# edges	# bends
v_k, $inc(v_k) = 2$	1	2	1
v_k, $inc(v_k) = 3$	1	3	2
face F_k	p	$p+1$	p

Notice that every vertex v has $deg(v) \leq 4$, thus $m \leq 2n$. Hence adding a vertex v_k with $inc(v_k) = 3$ occurs at most $\lceil \frac{n}{2} \rceil - 2$ times. This leads to a total of $n + \lfloor \frac{m-n}{2} \rfloor + 3 \leq$

$\lfloor \frac{3}{2}n \rfloor + 3$ bends. (v_1, v_n) has at most 3 bends, all other edges have at most 2 bends. □

Lemma 4.4 *The gridsize is at most $n \times n$.*

Proof: Inspect again the increase in X- and Y-direction, when adding the vertices and faces. When we add p vertices, then it follows that the increase in Y-direction is at most p. In X-direction the distance between $x(w_{l-1})$ and $x(w_l)$ is 1 $(1 < l < p)$. The distance between $x(w_{p-1})$ and $x(w_p)$ can be > 1, but this implies $x(c_j) - x(c_i) > p-1$, hence this gives no increase in X-direction. Notice that (c_i, w_1) is either horizontal or vertical. If (c_i, w_1) is horizontal, then this means an increase in X-direction; if (c_i, w_1) is vertical and $mark(w_1) = right$, then one outgoing edge of w_1 will go via $l(w_1)$, hence this means also an increase in X-direction later. We assign this to the insertion step. Similar holds for (w_p, c_j). Since $x(c_j) \geq x(c_i) + 1$ holds initially, we have a total increase in X-direction of at most p units.

Consider the case of adding v_k with $inc(v_k) = 2$. In X-direction the increase is at most 1, due to the fact that one extra bend might be necessary for an outgoing edge of v_k. The increase in Y-direction is at most 1. If $inc(v_k) = 3$, then the increase in X-direction is 0 and the increase in Y-direction is 1.

For (v_1, v_2) we have outgoing edges via $l(v_1)$ and $r(v_2)$. We assign the corresponding increase in X-direction to the insertion step of (v_1, v_2). This means that starting with (v_1, v_2) implies 3 units in X-direction and 1 unit in Y-direction. Adding v_n only implies 2 units in Y-direction. The steps $3, 4, \ldots, K-1$ give the following increases in X- and Y-direction:

step	# vertices	# X-dir.	# Y-dir.
v_k, $inc(v_k) = 2$	1	1	1
v_k, $inc(v_k) = 3$	1	0	1
face F_k	p	p	p

This leads to a grid of size at most $n \times n$. □

Corollary 4.5 *There is a linear time and space algorithm to draw every triconnected 4-planar graph G orthogonal on an $n \times n$ grid with at most $n + \lfloor \frac{1}{2}(m-n) \rfloor + 3 \leq \lfloor \frac{3}{2}n \rfloor + 3$ bends, in which every edge has at most three bends and length $O(n)$.*

In figure 6 an example is given of the graph G of figure 2, minus the edge (6,7).

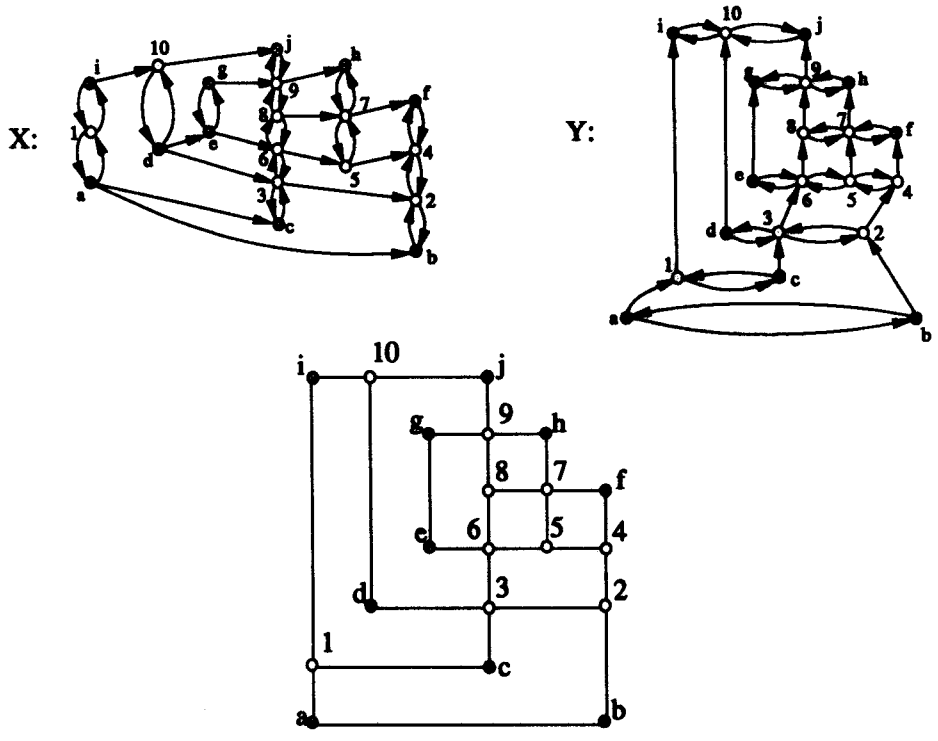


Figure 6: Orthogonal drawing of a 4-planar graph.

5 Orthogonal drawings of 3-planar graphs

5.1 Introduction

In this section we inspect the problem of drawing a 3-planar graph orthogonal on a rectilinear grid. In [31], Storer inspected the problem of embedding a 3-planar graph in the grid, and obtained the following negative result:

Theorem 5.1 *There are 3-planar graphs that require at least an $\frac{n}{2} \times \frac{n}{2}$ grid; there are 3-planar graphs that require at least $\frac{n}{2} + 1$ bends in any orthogonal grid drawing.*

On the positive side he presented several polynomial-time heuristics to draw a 3-planar graph with at most n bends. Tamassia [32] presented an $O(n^2 \log n)$ algorithm that, given an embedded 3-planar graph, computes a drawing with a minimum number of bends on an $n \times n$ grid. In [34], Tamassia & Tollis presented a linear algorithm to draw every 3-planar graph with at most $n+2$ bends on an $n \times n$ grid. In this section we present a linear time and space algorithm to draw every 3-planar graph with at most $\lfloor \frac{n}{2} \rfloor + 1$ bends on an $\lceil \frac{n}{2} \rceil \times \lceil \frac{n}{2} \rceil$ grid. This improves all previous bounds and matches the worst-case lower bounds and, hence, is best

possible. An interesting side-effect is that there is a spanning tree using $n-1$ straight-line edges. All $m-n+1$ non-tree edges have at most one bend.

5.2 Triconnected 3-planar graphs

Assume first that G is triconnected. By Euler's formulae, n is even, $m = \frac{3}{2}n$ and $f = \frac{n}{2}+2$. Let an *lmc*-ordering of G be given. Similar as in section 4 there are four directions to connect an edge at v, viz., left, right, up and down of v, denoted by $l(v), r(v), u(v)$ and $d(v)$, respectively. Every vertex v (except v_1, v_2 and v_n) has one outgoing edge, and we connect this edge via $u(v)$ at v. We start with placing v_1 and v_2 at $(0,1)$ and $(1,1)$. edge (v_1, v_2) goes via $d(v_1)$ and $d(v_2)$, hence via $(0,0)$ and $(1,0)$. We place the first face on the horizontal line between v_1 and v_2, i.e., via $r(v_1)$ and $l(v_2)$. When we add a face F_k with vertices $w_1, \ldots, w_p$ from leftpoint c_i to rightpoint c_j, then we place these vertices on the horizontal line on height $1 + \max\{y(c_i), y(c_j)\}$, with $x(w_1) = x(c_i)$, and if $p > 1$, we shift c_j such that $x(w_p) = x(c_j)$. The connections inbetween go via $d(w_1), r(w_1), l(w_2), r(w_2), \ldots, l(w_p), d(w_p)$. The complete (and simple) algorithm can now be described as follows:

$P(v_1) := (0,1); P(v_2) := (1,1);$
for $k := 3$ **to** $K-1$ **do**
  assume we add $w_1, \ldots, w_p$ $(p \geq 1)$, from c_i to c_j;
  $y(w_1) := \cdots := y(w_p) := 1 + \max\{y(c_i), y(c_j)\}$;
  update $x(c_i)$ and $shift(c_j)$; $x(w_1) := x(c_i)$;
  if $p > 1$ **then** $x(w_p) := \max\{x(w_1) + p - 1, x(c_j) + shift(c_j)\}$;
  for $l := 2$ **to** $p-1$ **do** $x(w_l) := x(w_1) + l - 1$;
  $shift(c_j) := \max\{shift(c_j), x(c_i) + p - 1 - x(c_j)\}$
rof;
$P(v_n) := (x(c_l), 1 + \max\{y(c_i), y(c_l), y(c_j)\})$, where $inc(v_n) = \{c_i, c_l, c_j\}$;
for $k := K$ **downto** 2 **do**
  assume we added $w_1, \ldots, w_p$ from c_i to c_j;
  $x(w_i) := x_{\text{insert}}(w_i) + \sum_{1 \leq j \leq i} shift(w_j)$; $(1 \leq i \leq p)$
  if $p = 1$ **then** $shift(c_j) := shift(w_1)$ **else** $shift(c_j) := x(w_p) - x_{\text{insert}}(c_j)$;
rof

Lemma 5.2 *The number of bends is at most* $\frac{n}{2}+2$.

Proof: Since $m = \frac{3}{2}n$, we add at most $\frac{n}{2}-2$ times a vertex v with $inc(v) = 2$, each one introduces 1 bend. The edge (v_1, v_2) introduces 2 bends, as well as adding v_n. □

Lemma 5.3 *The gridsize is at most* $\frac{n}{2} \times \frac{n}{2}$.

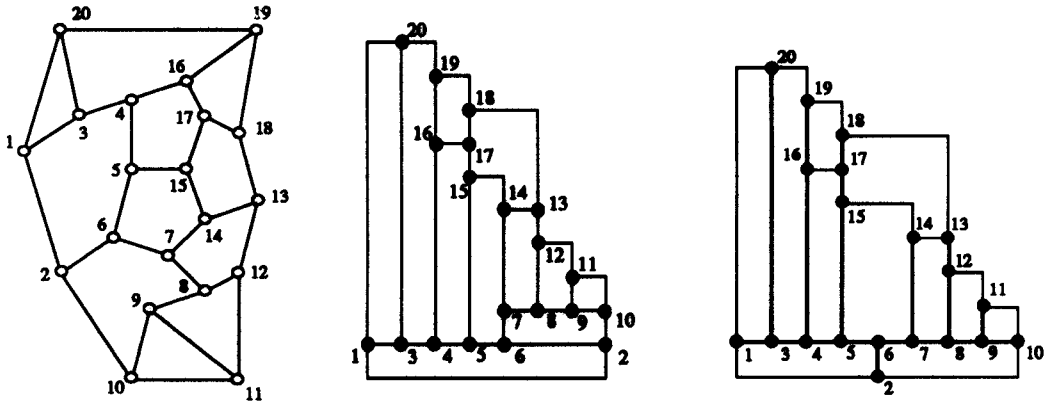


Figure 7: Orthogonal drawing of a 3-planar graph.

Proof: Edge (v_1, v_2) gives 1 unit in X- and Y-direction. Then we add $\frac{n}{2} - 1$ times a face with $p \geq 1$ vertices, increasing the X-direction with at most $p-1$ units and the Y-direction (except the first time) by 1 unit. Adding v_n increases the X-direction by 1 unit. Counting this together leads to at most $\frac{n}{2}$ units in X-direction and $\frac{n}{2}$ units in Y-direction. □

In figure 7(b), an example is given of a triconnected 3-planar graph from [21].

We can change the drawing as follows, such that there is one bend less, and there is a spanning tree, using only straight-line edges (if $n > 4$). Let the vertices of the first drawn face be numbered $v_1, v_i, v_{i-1}, \ldots, v_3, v_2$. We place $v_1, v_i, v_{i-1}, \ldots, v_3$ on a horizontal line, and place v_2 on $(x(v_3), y(v_3) - 1)$. Let F' be the other face, where (v_2, v_3) belong to. Let $v_j, \ldots, v_k$ be the other vertices of F'. We draw $v_j, \ldots, v_k$ on a horizontal line on height $y(v_3)$, as shown in figure 7(c). The remaining faces are drawn similar as before. Notice that using this strategy, every triconnected planar graph G with n vertices can be drawn orthogonal on a grid of size at most $\frac{n}{2} \times (\frac{n}{2} - 1)$, with at most $\frac{n}{2} + 1$ bends, in which there is a spanning tree, using only straight-line horizontal and vertical edges (if $n > 4$).

Finally we notice that better bounds can be obtained if the dual graph H of G is a 4-connected planar graph in which the outerface is a quadrangle, and every other face is a triangle. It has been shown by Bhasker & Sahni [1] that in this case G can be drawn in linear time orthogonal such that there are at most 4 bends. By a related canonical ordering for 4-connected planar graphs, and applying the placement method of He [16], we can achieve the same orthogonal drawing in a very simple way in linear time.

5.3 Arbitrary 3-planar graphs

5.3.1 Definitions

In this section we generalize the results of section 5.2 to arbitrary 3-planar graphs G in the same way as described in [21]. We assume first that G is biconnected. We start with some definitions with respect to triconnectivity, as described in [17]. If the pair of vertices $\{v_a, v_b\}$ disconnects G then $\{v_a, v_b\}$ is called a *separation pair*. Let the edges E of G be divided into equivalence classes $E_1, E_2, \ldots, E_p$, and let $E' = \bigcup_{i=1}^{k} E_i$ and $E'' = \bigcup_{i=k+1}^{p} E_i$. If $G_1 = (V(E'), E' \cup (a,b))$ and $G_2 = (V(E''), E'' \cup (a,b))$, then the graphs G_1 and G_2 are called the *split graphs* of G with respect to $\{v_a, v_b\}$. Replacing a graph G by two split graphs is called *splitting* G. The new edges (v_a, v_b), added to G_1 and G_2 are called *virtual edges*. Reassembling the two split graphs G_1 and G_2 into G is called *merging*. Merging is the inverse of splitting. Suppose a graph G is split, the split graphs are split, and so on, until no more splits are possible (each remaining graph is triconnected). The graphs constructed in this way are called *split components* of G. The split components of a graph G are of three types: *triple bonds* (i.e., a set of three multiple edges), *triangles* (i.e., a cycle consisting of three edges), and triconnected graphs. The *bonds* and *cycles* are obtained from triple bonds and triangles, respectively, by merging as far as possible. We call these bonds, cycles and triconnected graphs the *triconnected components* of G. The split components of a graph G are not necessarily unique, but the triconnected components of G are unique. Suppose finally that $\{v_a, v_b\}$ is a separation pair of a graph G and that G is split at $\{v_a, v_b\}$, the split graphs are split at $\{v_a, v_b\}$, and so on, until no more splits are possible at $\{v_a, v_b\}$. A graph constructed in this way is called an $\{v_a, v_b\}$-*split component* of G if it has at least one real (i.e., non-virtual) edge.

Let the triconnected components of G be given. From these components we construct a *2-subgraph tree* T (as introduced in [22][2]): for every triconnected component V_i we create a node v_i in T. Let one node v_r, not representing a bond or K_4, be the root of T. Let these nodes v_j be the children of v_r, for which V_j has at least one edge in common with V_r. Repeating this we make these nodes v'_j children of v_j, for which V'_j has at least one edge in common with V_j, etc. Notice that the triconnected components, which are leaves of T, have exactly one virtual edge, which is between its separation pair $\{v_a, v_b\}$. Every vertex has degree 2 or 3, hence the only bonds which can occur are triple bonds. Every separation pair has 2 or 3 split components. Every vertex v_a of a separation pair $\{v_a, v_b\}$ has degree 3 in G by definition. If $\{v_a, v_b\}$ has 3 split components, then the components have one incident edge with v_a and one edge with v_b. By definition of the triconnected components at least 2 of these 3 are cycles. The third component is a cycle or a bond. In the drawing algorithm, we merge the third component with one of the two other cycles. This merged graph is a cycle in which v_a and v_b are not necessarily neighbors. This eliminates all bonds from the 2-subgraph tree. More precisely, we can do it such that this merged

[2]Independently, a similar tree was also introduced as the SPQR-tree in [7].

graph is a child of the other $\{v_a, v_b\}$-split component in T. If $\{v_a, v_b\}$ has 2 split components, then one is a triconnected graph and the other one is a cycle.

5.3.2 Drawing Biconnected 3-planar Graphs

The drawing algorithm for biconnected 3-planar graphs follows the structure of the reduced 2-subgraph tree. Every triconnected component V_i has exactly the separation pair in common with the triconnected component V_j, if v_j is the parent of v_i in T. We call the common vertices of V_i and V_j the *parent vertices*. We draw the triconnected component with parent vertices v_a and v_b as follows:

- A triconnected graph with n' vertices is drawn using the algorithm, described in section 5.2 such that the parent vertices are v_1 and v_2. This means that after deleting virtual edge (v_1, v_2) the used area is at most $\frac{n'}{2} \times (\frac{n'}{2} - 1)$. $y(v_1) = y(v_2)$ and $x(v_2) - x(v_1) = \frac{n'}{2}$.

- A cycle C with n' vertices will be drawn on an area of size at most $\lceil \frac{n'}{2} \rceil \times (\lceil \frac{n'}{2} \rceil - 1)$. v_a is placed in the corner, and v_b is drawn such that $y(v_b) = y(v_a)$ (or $y(v_a) + 1$, if $(v_a, v_b) \notin C$), and $x(v_b) = x(v_a) + \lceil \frac{n'}{2} \rceil$.

We draw each triconnected component V_j after all triconnected components V_i, for which v_i is a child of v_j in T, are drawn. We want to replace the virtual edge (v_a, v_b) in V_j by V_i such that we again use an area of size at most $\lceil \frac{|V_i|+|V_j|}{2} \rceil \times (\lceil \frac{|V_i|+|V_j|}{2} \rceil - 1)$. If V_j is a cycle, then replacing the virtual edge by the triconnected graph or cycle V_i is straightforward as follows: assume (v_a, v_b) is a straight line. By the algorithm of section 5.2 $y(v_a) = y(v_b)$ and v_a and v_b are placed on cornerpoints in V_i. Thus $l(v_a), d(v_a)$ and $r(v_b), d(v_b)$ are free in V_i, hence we can stretch virtual edge (v_a, v_b) (which has length ≥ 1 already in V_j) to length $x(v_b) - x(v_a)$, and add V_i inside the drawing of V_j. This increases the X- and Y-direction both by at most $\frac{|V_i|}{2} - 1$. A similar follows if (v_a, v_b) has one bend.

Assume finally that V_j is a triconnected graph, thus V_i is a cycle. Assume w.l.o.g. that V_i also contains one other virtual edge (v_c, v_d) of a triconnected graph V_l, v_l child of v_i in T. We consider first the case that v_i consists of the cycle on the vertices v_a, v_b, v_c and v_d. After replacing (v_c, v_d) by V_l we assume that v_c and v_d are placed on the cornerpoints of the rectangle of size $\frac{|V_l|}{2} \times (\frac{|V_l|}{2} - 1)$ of the drawing of V_l, with $x(v_d) = x(v_c) + \frac{|V_l|}{2}$. There are several possibilities for the virtual edge (v_a, v_b) in the drawing of V_j, which has to be replaced by the edges $(v_a, v_c), (v_b, v_d)$ and the drawing V_l. The Y-direction of V_l has size $\frac{|V_l|}{2} - 1$, hence we can place edge (v_a, v_c) in Y-direction of V_l and (v_b, v_d) in direction of the virtual edge (v_a, v_b). Doing the replacements as shown in figure 8 for the different cases of (v_a, v_b), this leads to an increase in X- and Y-direction of at most $\frac{|V_l|}{2} = \frac{|V_l|}{2} - 1$.

Let us now inspect the different situations in figure 8 in more detail. Situation (a) occurs when we add one vertex. Situation (b) and (c) are the cases for adding

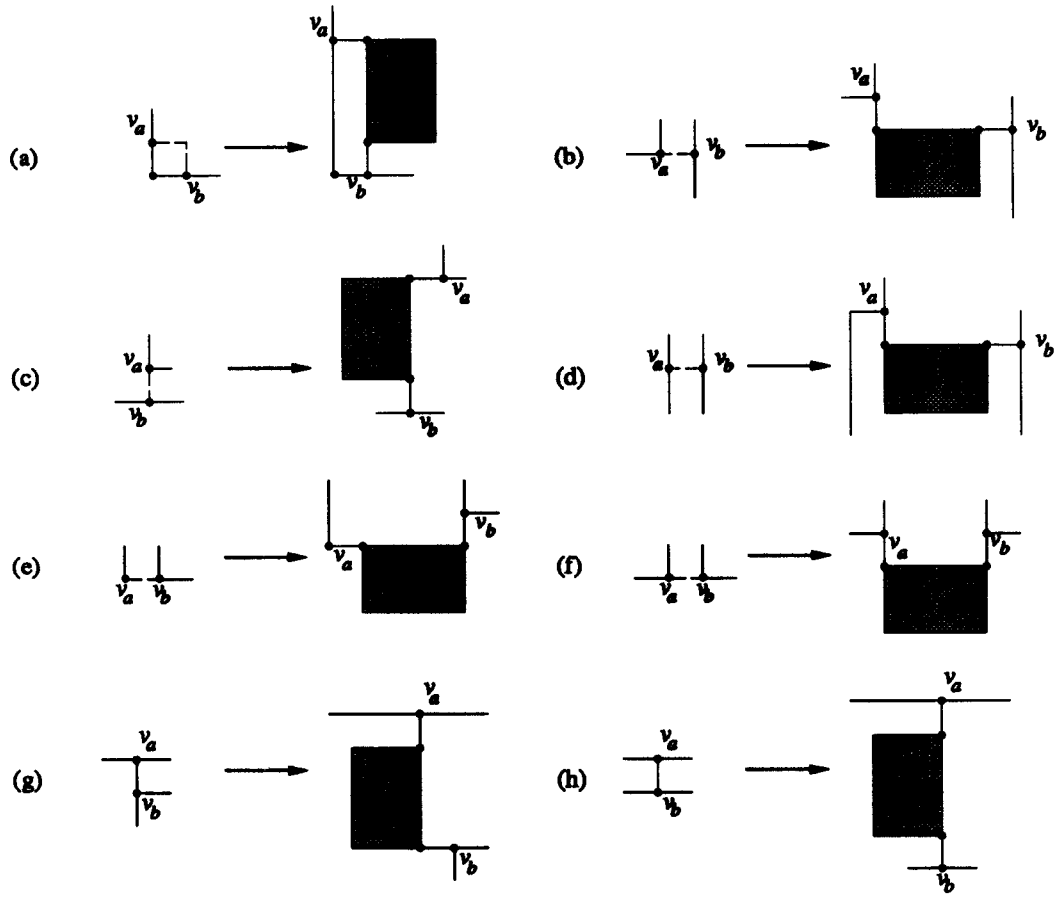


Figure 8: Replacing virtual edges by triconnected components.

a chain with at least 3 vertices. Situation (d) occurs when we add in V_j a chain of length 2. But here we may place a bend in the incoming edge of v_a, because we still have a path, using only straight-line edges from v_b to v_a. Also the increase in X-direction by adding a chain of length 2 is 1. Hence lemma 5.2 and lemma 5.3 still holds.

Situation (e) occurs when vertex v_a is also the parentvertex of V_j. A similar situation can occur for vertex v_b. From this replacement it follows that though the size of the total drawing is at most $\lceil \frac{|V_i|+|V_j|}{2} \rceil \times (\lceil \frac{|V_i|+|V_j|}{2} \rceil - 1)$, the parentvertices v'_a and v'_b of V_i are not necessarily the cornerpoints of this rectangle. But still $d(v'_a), l(v'_a)$ and $d(v'_b), r(v'_b)$ are free, and drawing straight lines from these connections does not cross any other vertex or edge of V_i. Hence we can still use the given replacements for drawing V_j inside another component. Situation (f) occurs when the virtual edge is a horizontal edge, belonging to the first added chain between the parentvertices of V_j.

When the virtual edge (v_a, v_b) is the internal incoming edge of v_n of V_j, then we have 2 situations. Situation (g) occurs when $|V_j| > 4$. Then either $l(v_a)$ or $r(v_b)$ is free, and we can use a similar replacement. If $|V_j| = 4$, then $l(v_a)$ and $r(v_b)$ are not free, hence the only possible replacement is as shown in situation (h). Though this leads to a drawing of size $\lceil\frac{|V_i|+|V_j|}{2}\rceil \times (\lceil\frac{|V_i|+|V_j|}{2}\rceil - 1)$, $x(v'_b) = x(v'_a) + \lceil\frac{|V_i|}{2}\rceil - 1$ holds. But adding V_i inside another triconnected graph by drawing the 2 connecting edges horizontal (as in situation (h)) already solves the problem.

We only considered the case yet that $|V_i| = 4$. If V_i contains more virtual edges, then we can replace these virtual edges by the triconnected graphs or cycles by the replacements described above. Also if V_i contains more real edges, then we can draw them alternatingly horizontal and vertical. This implies no bends and is correct with respect to the gridsize, thereby completing the following lemma:

Lemma 5.4 *After replacing a virtual edge (v_a, v_b) by the corresponding orthogonal drawing of V_i in an orthogonal drawing of V_j, the total required grid size is at most $\lceil\frac{|V_i|+|V_j|}{2}\rceil \times (\lceil\frac{|V_i|+|V_j|}{2}\rceil - 1)$.*

This means that after replacing all virtual edges of V_i by the triconnected components V_j, we obtain an orthogonal drawing of size $\lceil\frac{n'}{2}\rceil \times (\lceil\frac{n'}{2}\rceil - 1)$, with $n' = |V_i| + |\bigcup V_j, v_j$ a descendant of v_i in $T|$. We continue this approach until we are at root V_r. If V_r is a triconnected graph, then V_r is drawn as shown in figure 7(c).

One can now verify the following theorem:

Theorem 5.5 *There is a linear time and space algorithm to draw a biconnected 3-planar graph on a $\lceil\frac{n}{2}\rceil \times \lceil\frac{n}{2}\rceil$ grid with at most $\lfloor\frac{n}{2}\rfloor + 1$ bends, such that there is a spanning tree of $n-1$ straight-line edges, all non-tree edges have at most 1 bend (if $n > 4$).*

In figure 9 an example is given of a biconnected 3-planar graph G, the 2-subgraph tree of G, and the corresponding orthogonal drawing of G.

5.3.3 Drawing general 3-planar graphs orthogonal

Finally we extend the orthogonal drawing algorithm to arbitrary 3-planar graphs. Assume that all biconnected components V_i of G are drawn orthogonal on a grid of size at most $\lceil\frac{|V_i|}{2}\rceil \times (\lceil\frac{|V_i|}{2}\rceil - 1)$. We construct a *block tree* T: every biconnected component (block) and every cutvertex v_c is represented by a node in T. There is an edge between a blocknode b_i and a cutnode c_j, if $c_j \in V_i$, with V_i the corresponding block of b_i. Let one blocknode b_r be the root of T. For every block V_i, label this cutvertex c_i, which is the parent of b_i in T. We assume that the cutvertex c_i of block B_i is drawn in one corner of the orthogonal drawing of B_i (this can easily be obtained by letting v_1 and v_2 be the two neighbors of c_i in B_i). Let c_l be the other neighbor of c_i, then c_l is a cutvertex as well. c_l has one or two sons in T. We first

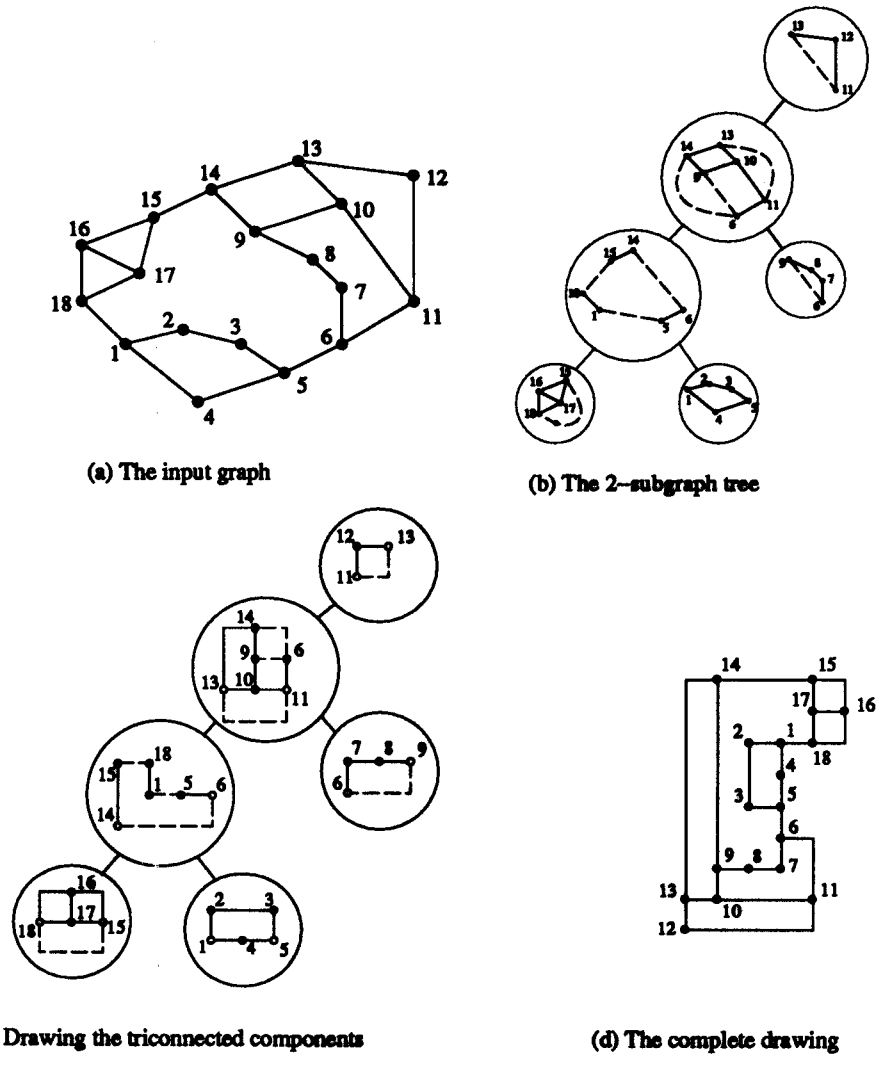


Figure 9: Orthogonal drawing of a biconnected 3-planar graph.

draw these 2 corresponding blocks, and then merge it into one drawing, as shown in figure 10.

It follows that there will be no extra bends included in the drawing. Also the required area for drawing blocks V_i and V_j is again at most $\lceil \frac{|V_i|+|V_j|}{2} \rceil \times \lceil \frac{|V_i|+|V_j|}{2} \rceil$, which completes the following theorem:

Theorem 5.6 *There is a linear time and space algorithm to draw a 3-planar graph on a* $\lceil \frac{n}{2} \rceil \times \lceil \frac{n}{2} \rceil$ *grid with at most* $\lfloor \frac{n}{2} \rfloor + 1$ *bends, such that there is a spanning tree of* $n-1$ *straight-line edges, all non-tree edges have at most 1 bend.*

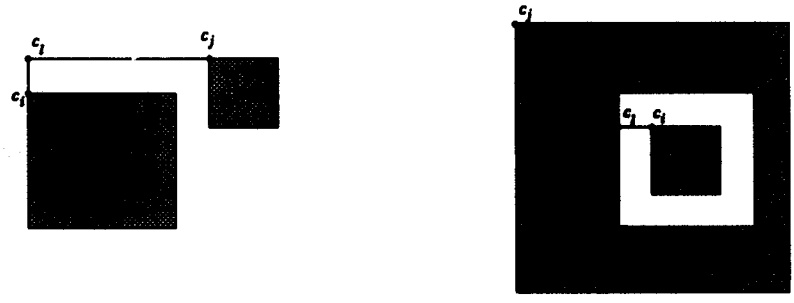


Figure 10: Drawing the blocks orthogonal.

6 Convex drawings

The *lmc*-ordering is a generalization of the canonical ordering of De Fraysseix et al. [13]. We can apply the *lmc*-ordering and the shift-method to get a linear implementation of the straight-line grid drawing algorithm of triangulated planar graphs [13]. Moreover, we will show that we can modify this algorithm such that we can draw every triconnected planar graph with convex faces on a grid. (In [6] another linear implementation of [13] is described, in which the input graph must be triangulated.)

The algorithm maintains a straight-line embedding during every step k of the *lmc*-ordering such that

1. v_1 is at $(0,0)$, v_2 is at $(2k-4,0)$.
2. If $v_1 = c_1, c_2, \ldots, c_r = v_2$ is the outerface of G_k in step k, then $x(c_1) < x(c_2) < \cdots < x(c_r)$.
3. The edges (c_i, c_{i+1}) have slopes $+1$ or -1.

Assume first that G is triangulated, in which case we can add a vertex v_k in every step k of the *lmc*-ordering [13]. Let $L(v)$ be a set of vertices. The idea of the algorithm is the following: when we add vertex v_k, then all vertices $c_{i+1}, \ldots, c_{j-1}$ are shifted 1 to the right, the vertices $c_j, \ldots, c_r$ are shifted 2 to the right (and of course, several internal vertices of G_{k-1} must be shifted 1 or 2 to the right as well). The crossing point of the line with slope $+1$ from c_i and the line with slope -1 from c_j denotes the place for vertex v_k. All vertices $c_i, \ldots, c_j$ are visible from this point, see figure 11 for the corresponding picture. The exact algorithm is as follows:

{ In each step k, let $c_1, \ldots, c_r$ be the outerface,
and c_i and c_j the left- and rightpoint of v_k, resp.}
$\mu((x_1,y_1),(x_2,y_2)) := \frac{1}{2}(x_1 - y_1 + x_2 + y_2, -x_1 + y_1 + x_2 + y_2)$;
$P(v_1) := (0,0)$; $L(v_1) := \{v_1\}$;
$P(v_2) := (2,0)$; $L(v_2) := \{v_2\}$;

$P(v_3) := (1,1); L(v_3) := \{v_3\};$
for $k := 4$ **to** n **do**
  for $v \in \bigcup_{l=j}^{r} L(c_l)$ **do** $x(v) := x(v) + 2;$
  for $v \in \bigcup_{l=i+1}^{j-1} L(c_l)$ **do** $x(v) := x(v) + 1;$
  $P(v_k) := \mu((x(c_i), y(c_i)), (x(c_j), y(c_j)));$
  $L(v_k) := \{v_k\} \cup \bigcup_{l=i+1}^{j-1} L(c_l);$
rof

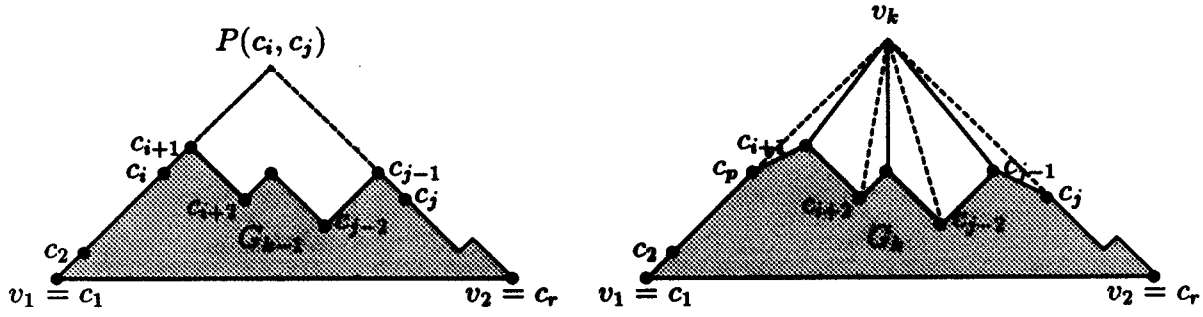


Figure 11: Idea of the straight-line drawing algorithm.

The correctness of this algorithm is described in [13]. There are shifts of 1 and 2 in the algorithm, and therefore we introduce 2 variables. The complete algorithm can now be implemented to run in linear time as follows:

$P(v_1) := P(v_2) := (0,0);$
for $k := 3$ **to** n **do**
  update $x(c_i)$ and $shift(c_j)$;
  $shift(c_j) := shift(c_j) + 2;$
  $P(v_k) := \mu((x(c_i), y(c_i)), (x(c_j) + shift(c_j), y(c_j)));$
rof;
$shift(v) := rshift(v) := 0$ for all $v \in V$;
for $k := n$ **downto** 2 **do**
  for all internal vertices v_i of v_k **do** $shift(v_i) := shift(v_k) + rshift(v_k) + 1;$
  $rshift(c_j) := rshift(c_j) + rshift(v_i) + 2;$
  $x(v_k) := x_{\text{insert}}(v_k) + shift(v_k) + rshift(v_k);$
rof

Moreover, using the *lmc*-ordering, it is already sufficient that the planar graph is triconnected, because when adding a face F_k with new vertices $w_1, \ldots, w_p$, we can draw $w_1, \ldots, w_p$ on a horizontal line with distance 2 inbetween. The edge (c_i, w_1) has slope +1, the edges $(w_1, w_2), \ldots, (w_{p-1}, w_p)$ have slope 0 and length 2, and the edge (w_p, c_j) has slope -1. It is easy to see that this still gives a correct straight-line drawing on an $(2n-4) \times (n-2)$ grid. We will now modify this algorithm a little such that all interior faces are convex.

Assume we add a vertex v_k, and let c_p and c_q two adjacent vertices on C_{k-1}, such that there is no neighbor c_l of v_k, with $p < l < q$. Let F' be the face, containing c_p, c_q and v_k.

Lemma 6.1 *All edges (c_a, c_{a+1}), $p \leq a < \alpha$, have slope -1; all edges (c_{b-1}, c_b), $\alpha \leq \beta < b \leq q$, have slope $+1$. $\beta = \alpha$ or $\alpha + 1$, and if $\beta = \alpha + 1$, then (c_α, c_β) has slope 0.*

Proof: Every vertex has a neighbor, added in a later step and, hence by the drawing algorithm, is placed higher. In step k the vertices $c_{p+1}, \ldots, c_{q-1}$ have already higher placed neighbors. Let c_β be the lowest placed vertex, with $p \leq \beta \leq q$. If $\beta > p$ and $(c_{\beta-1}, c_\beta)$ is horizontal, then from $c_{\beta-1}$ to c_p and from c_β to c_q the vertices are strictly increasing in Y-direction. If $\beta = p$ or $(c_{\beta-1}, c_\beta)$ is not horizontal, then from c_β to c_p and from c_β to c_q the vertices are strictly increasing in Y-direction. Since we have only slopes $+1$, 0 or -1 in every step on the outerface, this completes the proof. □

To achieve a convex face F', we add edges from c_q to $c_\beta, c_{\beta+1}, \ldots, c_{q-1}$ in step k. This does not destroy the planarity in the embedding, and it means in the algorithm of [13] that if we shift c_q, then we also shift $c_{\beta+1}, \ldots, c_{q-1}$ to the right, when adding vertex v_k by the algorithm of [13]. Notice that when c_q is the rightpoint or c_p is the leftpoint of v_k, then c_q is shifted 1 more to the right than c_p. But then $c_{\beta+1}, \ldots, c_q$ are shifted one more to the right than $c_p, \ldots, c_\beta$. (If $\beta = q$, then no internal vertex is shifted to the right.) We now have the following slopes after adding v_k, and applying the shifts.

- The slopes of (c_p, v_k) and (c_q, v_k) are in the range $(-\infty, -1] \cup [+1, \infty)$.
- The edges $(c_p, c_{p+1}), \ldots, (c_{\alpha-2}, c_{\alpha-1})$ have slope -1.
- The edge $(c_{\alpha-1}, c_\alpha)$ has slope in the range $[-1, 0)$.
- If $\beta > \alpha$, then the edge (c_α, c_β) has slope 0.
- The edges $(c_{\beta+1}, c_{\beta+2}), \ldots, (c_{q-1}, c_q)$ have slope $+1$.
- If $\beta < q$ then the slope of edge $(c_\beta, c_{\beta+1})$ is in the range $(0, +1]$, else the slope of the edge $(c_{\beta-1}, c_\beta)$ is in the range $[-1, 0]$.

This implies that the face F is convex when inserting v_k in step k. The same values follow when we add a complete chain $w_1, \ldots, w_p$ instead of one vertex v_k, because in this case $w_1, \ldots, w_p$ are placed on a horizontal line.

Theorem 6.2 *The faces remain convex during the algorithm.*

Proof: Inspect the face F', obtained by adding v_k in step k of the *lmc*-ordering, with neighbors c_p and c_q on C_{k-1}, also part of F'. (The proof is analog when we consider a face, obtained by adding a chain $w_1, \ldots, w_p$.) After adding v_k, only v_k and possibly c_p or c_q are part of C_k and, hence, they will obtain shifts of new added vertices. If $shift(c_q)$ is increased by adding a vertex $c_l, l > k$, then also $shift(c_{\beta+1}), \ldots, shift(c_{q-1})$ will be increased by the same value, since there are edges from $c_\beta, c_{\beta+1}, \ldots, c_{q-1}$ to c_q. If $shift(v_k)$ is increased by adding a vertex $v_l, l > k$, then also $shift(c_q)$ will be increased by the same value. If c_p is not the leftpoint of v_k, then also $shift(c_p)$ will be increased, in which case all vertices of F' will be shifted with the same value to the right.

Finally, when $shift(c_p)$ is increased by adding a vertex $v_l, l > k$, then by the *lmc*-ordering, c_p is not the rightpoint of v_l, hence c_p is an internal vertex of v_l. c_p is the leftpoint of v_k, and of some vertices $v_{\alpha_1}, \ldots, v_{\alpha_i}$. (Assume that if $i = 0$, then there are no such other vertices.) Notice that by the algorithm it follows that v_{α_1} is an internal vertex or rightpoint of v_l and that v_{α_j} is an internal vertex or rightpoint of $v_{\alpha_{j-1}}$ $(2 \leq j \leq i)$. (c_p, v_k) has slope $+1$ when inserting v_k, hence it follows by the algorithm that v_k is also the internal vertex of a vertex $v_{l'}, l' > k$. It can easily be verified that $l = l'$ if $i = 0$, or $v_{l'}$ is an internal vertex or rightpoint of v_{α_j}. But this yields that there is a path from v_l to v_k, following only internal vertices or rightpoints. This means that $shift(v_k)$ increases at least as much as $shift(c_p)$ by the addition of v_l.

This leads to the following slopes for all edges of face F', during any step of the algorithm:

- The edge (c_i, v_k) has slope in $(-\infty, -1] \cup (0, \infty)$.
- The edge (c_q, v_k) has slope in $(-\infty, 0) \cup [+1, \infty)$.
- The edges $(c_i, c_{i+1}), \ldots, (c_{\alpha-2}, c_{\alpha-1})$ have slopes -1.
- The edge $(c_{\alpha-1}, c_\alpha)$ has slope in $[-1, 0)$.
- If $\beta > \alpha$ then edge (c_α, c_β) has slope 0.
- The edge $(c_\beta, c_{\beta+1})$ has slope in $(0, +1]$
- The edges $(c_{\beta+1}, c_{\beta+2}), \ldots, (c_{q-1}, c_q)$ have slopes $+1$.

Since this implies convexity, this completes the proof. □

Finally we remove the added edges from c_q to $c_\beta, c_{\beta+1}, \ldots, c_{q-1}$.

Theorem 6.3 *There is a linear algorithm to draw a triconnected planar graph convex with straight lines on a grid of size at most* $(2n-4) \times (n-2)$.

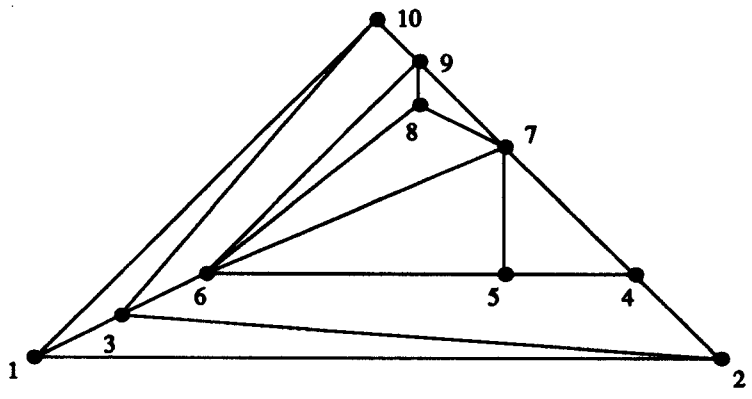


Figure 12: Convex drawing of the graph of figure 2.

In [38], Tutte showed that every triconnected planar graph admits a convex drawing. However, this approach seems not to be implementable to run in linear time [39]. Thomassen [37] characterized the class of planar graphs which admit a convex drawing. In [4], a linear algorithm for drawing this class of planar graphs convex is presented. However, here the vertices are not placed on grid coordinates and as a consequence, the edges can be arbitrary short, and the resulting drawings might be very hard to read.

Our algorithm not only outperforms the algorithms of [39, 4], it is also much easier to implement than the algorithm of [4] (see also figure 15 for this). Since the vertices are placed on grid points, the resulting drawings seem to be more pleasant than in [4]. Also, this algorithm gives a new proof that every triconnected planar graph admits a planar drawing, in which every interior face is convex. The outerface is a triangle. With respect to the tightness of the grid size we note that every strictly convex drawing of a cycle with n vertices requires an $\Theta(n^3)$ grid [25]. In figure 12, the straight-line convex drawing of the graph in figure 2 is given.

7 Extensions, duality aspects and optimizations

7.1 Duality aspects

In [21], a drawing algorithm to draw 3-planar graphs on a triangled grid is given, with in at most one edge bends. This is obtained by inspecting the dual graph H of the triconnected planar graph G: every vertex v_F in H corresponds with a face F in G. There is an edge $(v_{F'}, v_{F''})$ in H, if F' and F'' share a common edge in G. H is triconnected and planar as well. If G is d-planar, then every face in H has at most d edges. By Euler's formulae, $m - n - f + 2 = 0$, hence $n_H = m_G - n_G + 2$. Let an *lmc*-ordering of G be given. We construct a labeling of the faces of G as

follows: when k is the smallest integer such that all vertices of face F belong to G_k in the *lmc*-ordering, then we set $label(F) = K + 3 - k$. Let $(v_1, v_n) \in F', F''$, and assume that F' is the outerface. Then we set $label(F') = 1$ and $label(F'') = 2$. Now we obtain the following interesting result:

Theorem 7.1 *The labeling of faces of G corresponds with an lmc-ordering of the dual graph H.*

Proof: We first prove that the assigned labeling corresponds with the canonical ordering of the dual graph H, by reverse induction on the steps of the *lmc*-ordering of G. Let $\overline{F}$ denote the vertex in H, corresponding with face F in G. Let H_i denote the induced subgraph on the vertices $\overline{F}$ with $label(F) \leq i$ in G. We start with deleting v_n from G. Let (v_1, v_n) belong to F' and F'' (F' the outerface), then $label(F') = 1$ and $label(F'') = 2$. Since $deg(v_n) \geq 3$, there are $deg(v_n) - 2$ remaining faces F_i in G, with $label(F_i) = 3$. By duality aspects, the corresponding vertices $\overline{F_i}$ form a consecutive chain from $\overline{F'}$ to $\overline{F''}$. Hence H_3 satisfies the constraints of theorem 2.1.

Let k be fixed, $3 < k < K$, and assume that face F_i or vertex v_i has already been determined for every $i > k$ such that the subgraph H_{K+3-i} satisfies the constraints of theorem 2.1. Consider the 2 cases in step k: deleting a vertex v_k or a face F_k from G_k. Assume first that we delete v_k with p lower-numbered neighbors. Then $p - 1$ faces F_{α_i} are deleted from G_k, which all have $label(F_{\alpha_i}) = K + 3 - k$. By construction of H, $\overline{F_{\alpha_i}}$ all have degree 2 in H_{K+3-k}. $\overline{F_{\alpha_1}}$ and $\overline{F_{\alpha_{p-1}}}$ have neighbors with lower label and each $\overline{F_{\alpha_i}}$ has neighbors with higher label.

If we delete a face F_k in step k, then $\overline{F_k}$ is added to H_{K+3-k}. $\overline{F_k}$ has at least 2 neighbors with lower label, and at least one neighbor with higher label. In both cases one easily verifies that the added chain or vertex to H_{K+3-k} is on the outerface. H_{K+3-k} is biconnected, and by induction H_{K+3-k} satisfies the constraints of theorem 2.1. Finally we end with edge (v_1, v_2) in G. Assume (v_1, v_2) belongs to F' (the outerface) and F''', then $label(F''') = K$ by definition. But since v_2 and v_n are neighbors of v_1 and belonging to the outerface it follows that $\overline{F'''}$ and $\overline{F''}$ are both neighbors of $\overline{F'}$ in H, and belong to a common face, which completes the proof that the assigned labeling is a canonical ordering for H.

In the same way, by considering the dual graph H it easily follows that the canonical ordering is leftmost, because if we delete a vertex v_k or face F_k from G_k, then the corresponding face $\overline{F_i}$ or vertex $\overline{F_k}$ is also leftmost with respect to the vertices $\overline{F'}$ and $\overline{F''}$ in H. □

Notice that in [29] a similar result is obtained for the *st*-ordering. Hence it seems that the *lmc*-ordering is a powerful generalization of the *st*-ordering in the triconnected case. (See also the example in figure 13.)

If G is triconnected and 3-planar, then H is a triangulated planar graph, hence we could also use the canonical ordering of [13] to obtain an *lmc*-ordering in section 5 for the orthogonal drawings of 3-planar graphs.

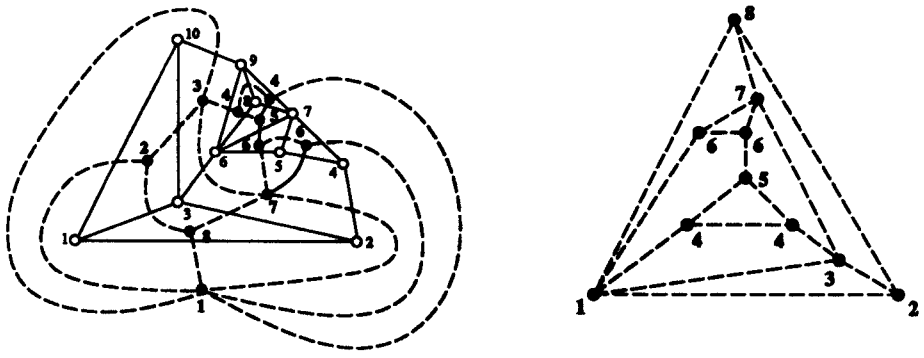
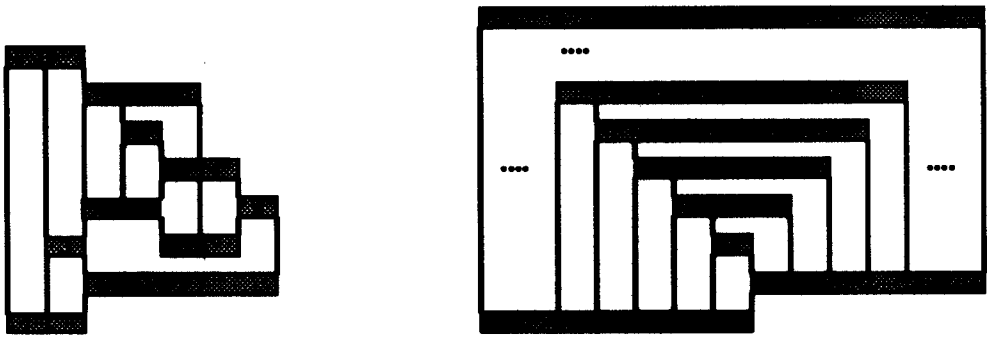


Figure 13: The dual of the *lmc*-ordering of the graph of figure 2 is also an *lmc*-ordering.

7.2 visibility representations

Since every *lmc*-ordering is also an *st*-ordering, we can use the *lmc*-ordering in various drawing applications, e.g., in constructing ***visibility representations*** [29]. A visibility representation maps vertices into horizontal segments and edges into vertical segments that intersects only the two corresponding vertex segments. All endpoints are on grid points. Assume we have an *lmc*-ordering in which we add a vertex in each step. This is possible by the ordering of [13] for triangulated planar graphs. (In [20] an alternative linear time algorithm is given for constructing visibility representations of triangulated planar graphs, using De Fraysseix' canonical ordering.) When we add a vertex v_k, then we have to know the y-coordinates of the segments of endpoints of incoming edges of v_k, and we have to know the x-coordinates of the edges (c_i, v_k) and (v_k, c_j). In this case we give every vertex only an y-coordinate, and every edge e receives an x-coordinate and a *shift*-variable. The horizontal segment, representing v, goes from $(x((v_k, c_i)), y(v_k))$ to $(x((v_k, c_j)), y(v_k))$. In every step we add a vertex, and we place all outgoing edges of v_k on distance 1 from each other on this segment. We update $shift((v_k, c_j))$ accordingly. The complete algorithm can now be described as follows:

$y(v_1) := 0$;
let v_1 have outgoing edges to $w_1, \ldots, w_p$; $x((v_1, w_i)) := i - 1; (1 \leq i \leq p)$
$y(v_2) := 1$;
let v_2 have outgoing edges to $w_1, \ldots, w_{p'}$; $x((v_2, w_i)) := p + i - 1; (1 \leq i \leq p')$
for $k := 3$ **to** K **do**
  update $x((c_i, v_k))$ and $shift((v_k, c_j))$;
  let v_k have outgoing edges to $w_1, \ldots, w_p$;
  $x((v_k, w_i)) := x((c_i, v_k)) + i - 1; (1 \leq i \leq p)$;
  $shift((v_k, c_j)) := \max\{shift((v_k, c_j)), x((v_k, c_i)) + p - 1 - x((v_k, c_j))\}$;



(a) Visibility representation of the graph of figure 2.

(b) graph with worst–case grid bounds for the visibility representation.

Figure 14: Visibility representations. The vertices have been given positive thickness for clarity.

rof;
for $k := K$ **downto** 3 **do**
$x((v_k, w_i)) := x_{\text{insert}}((v_k, w_i)) + \sum_{1 \leq j \leq i} shift((v_k, w_j)); (1 \leq i \leq p)$
$shift((v_k, c_j)) := \max\{0, x((v_k, w_p)) - x((v_k, c_j))\};$
rof

Since $inc(v) \geq 2$ for every vertex v, this means that adding v_k increases the Y-direction by 1 and the X-direction by $\max\{0, out(v_k) - 2\}$. If for every vertex v_i, $out(v_i) \geq 2$ holds, then this leads to a visibility representation on a grid of size at most $(n-2) \times (n-1)$. Indeed, if the planar graph is 4-connected, then a canonical ordering can be achieved in linear time, in which for every vertex v_k, $out(v_k) \geq 2$ holds, yielding a more compact layout.

Several compressing and optimization techniques are possible, leading in general to more compact layouts than the algorithms in [19, 27, 29]. Especially, when we add a face, then we can do it such that we increase the Y-direction by at most 2. Several other compressing techniques are possible. Moreover, compared with [29], we do not have to compute the dual graph.

Theorem 7.2 *There is an alternative linear time algorithm to construct a visibility representation of a planar graph on a grid of size at most* $(2n-5) \times (n-1)$.

In figure 14 a visibility representation of the graph in figure 2 is given, and a visibility representation of a graph is given, requiring an $(2n-5) \times (n-1)$ grid.

7.3 Optimizations

7.3.1 An alternative *shift*-method of Chrobak & Payne

In this section we explain the *shift*-method, as introduced by Chrobak & Payne [6]. Using this technique there is no need to verify the canonical ordering for triconnected planar graphs to the *lmc*-ordering. The crucial observation of [6] is that when we draw v_k, it is not really necessary to know the exact positions of c_i and c_j. If we only know their y-coordinates and their relative x-coordinates, i.e., if we know $x(c_j) - x(c_i)$, then we can compute $y(v_k)$ and the x-offset of v_k relative to c_i, that is $x(v_k) - x(c_i)$.

To obtain this, a tree T is constructed during the algorithm. In the first phase we add new vertices, compute their x-offsets and y-coordinates, and update the x-offsets of one or two vertices. In the second phase, we traverse the tree and compute the final x-coordinates by accumulating offsets. Suppose that vertex v is a T-ancestor of vertex w. By the *cumulative offset* from v to w, denoted by $c(v, w)$, we mean the sum of offsets along the branch from v to w including that of w but excluding that of v. Note that, if w is a T-ancestor of vertex x, then $c(v, x) = c(v, w) + c(w, x)$. By adding the x-coordinate of the root v_1 of the tree to the cumulative offset from v_1 to a node, one can determine the node's proper x-coordinate. We store for each vertex v the following information:

- *Left*(v) = the left T-son of v,
- *Right*(v) = the right T-son of v,
- $x(v)$ = the x-offset of v from its T-father,
- $y(v)$ = the y-coordinate of v.

Assume now that we add in step k the vertices $w_1, \ldots, w_p$ from c_i to c_j. If $p = 1$ then we add only one vertex. We apply the following adjustments of the tree T in each step:

1. *Right*$(c_i) := w_1$; *Right*$(w_p) := c_j$; *Left*$(w_p) := nil$;
2. *Right*$(w_i) := w_{i+1}$; *Left*$(w_i) := nil$; $(i \leq i < p)$
3. if $c_{i+1} \neq c_j$ then *Left*$(w_1) := c_{i+1}$ and *Right*$(c_{j-1}) := nil$;

Before adding this chain *Right*(c_i) was c_{i+1}, hence we have to update the x-offsets such that $x(c_{i+1}) + x(v_k) = x_{\text{old}}(c_{i+1})$. Similar must hold for $x(c_j)$. Updating these x-offsets is straightforward in the algorithms, described in this paper. As an example in figure 15 the complete convex drawing algorithm is described, using this technique of x-offsets of Chrobak & Payne [6].

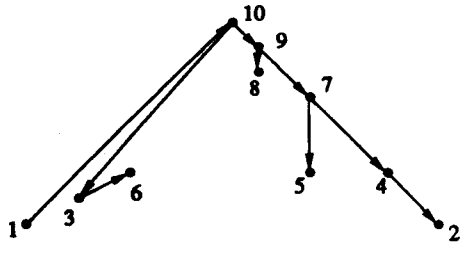
COMPLETE CONVEX DRAWING ALGORITHM

Let the canonical ordering be given for G;
$(x(v_1), y(v_1), \mathit{Left}(v_1), \mathit{Right}(v_1)) := (0, 0, \mathit{nil}, v_2)$;
$(x(v_2), y(v_2), \mathit{Left}(v_2), \mathit{Right}(v_2)) := (0, 0, \mathit{nil}, \mathit{nil})$;
for $k := 3$ **to** K **do**
  assume we add $w_1, \ldots, w_p$ $(p \geq 1)$ **from** c_i **to** c_j;
  for each neighbor c_α of w_p $(c_\alpha \neq c_i)$ **do**
    $\beta := \alpha - 1$;
    while $(c_\beta, w_p) \notin G_k$ and $y(c_{\beta-1}) < y(c_\beta)$ **do**
      $\beta := \beta - 1$;
    if $\beta < \alpha - 1$ **then**
      { change graph to maintain convexity }
      $\mathit{Right}(c_{\alpha-1}) := \mathit{Left}(c_\alpha)$;
      if $\mathit{Right}(c_{\alpha-1}) \neq \mathit{nil}$ **then**
        add $x(c_\alpha)$ to $x(\mathit{Right}(c_{\alpha-1}))$;
      $\mathit{Left}(c_\alpha) := \mathit{Right}(c_\beta)$; $\mathit{Right}(c_\beta) := c_\alpha$;
      $x(c_\alpha) := x(c_{\beta+1}) + \cdots + x(c_\alpha)$;
      $x(c_{\beta+1}) := x(c_{\beta+1}) - x(c_\alpha)$;
  rof;
  $\mathit{Right}(c_i) := w_1$; $\mathit{Right}(w_p) := c_j$; $\mathit{Left}(w_p) := \mathit{nil}$;
  for $l := 1$ **to** $p - 1$ **do**
    $\mathit{Right}(w_l) := w_{l+1}$, $\mathit{Left}(w_l) := \mathit{nil}$;
  if $c_{i+1} \neq c_j$ **then**
    $\mathit{Left}(w_1) := c_{i+1}$ and $\mathit{Right}(c_{j-1}) := \mathit{nil}$;
  $S := x(c_{i+1}) + \cdots + x(c_j)$;
  $(x(w_1), y(w_1)) := \mu((0, y(c_i)), (S + 2, y(c_j)))$;
  for $l := 2$ **to** p **do** $(x(w_l), y(w_l)) := (2, y(w_1))$;
  $x(c_j) := S + 2 - x(w_1)$;
  if $c_{i+1} \neq c_j$ **then** $x(c_{i+1}) := x(c_{i+1}) + 1 - x(w_1)$;
rof;
AccumulateOffset$(v_1, 0)$;

procedure AccumulateOffset(v : vertex, δ : integer);
begin
  $x(v) := x(v) + \delta$;
  if $\mathit{Left}(v) \neq \mathit{nil}$ **then** AccumulateOffset$(\mathit{Left}(v), x(v))$;
  if $\mathit{Right}(v) \neq \mathit{nil}$ **then** AccumulateOffset$(\mathit{Right}(v), x(v))$;
end;

The values of the x-offsets during the convex drawing algorithm, applied on the graph of figure 2.

vertex	steps k					
x-offset	3	4	5	6	7	8
1	0	0	0	0	0	0
2	1	2	2	2	2	2
3	1	1	1	1	1	-6
4	–	2	3	3	3	3
5	–	2	0	0	0	0
6	–	1	1	1	1	2
7	–	–	3	1	2	2
8	–	–	–	4	0	0
9	–	–	–	–	5	1
10	–	–	–	–	–	8



Endcoordinates:

vertex	$x(v)$	$y(v)$
1	0	0
2	16	0
3	2	0
4	14	2
5	11	2
6	4	2
7	11	5
8	9	6
9	9	7
10	8	8

Figure 15: The complete drawing algorithm with example.

7.3.2 Other Optimizations

Optimizations are possible in almost all drawing algorithms, presented in the previous sections. For example, in section 4 we can set $mark(v)$ left or right when $deg(v) = 3$, according to the smallest number of bends. In section 5 we originally always go upwards from c_i and c_j, even when there is no reason to. For example, if $l(c_j)$ is free and $y(c_j) > y(c_i)$, then we can place the new vertices on height $y(c_j)$. In the same way we can improve the drawings of section 3. If vertex v_k have $p \geq 7$ incoming edges, then we can use the points $(x(v) - \lceil \frac{p-4}{2} \rceil, y(v) + \lceil \frac{p-6}{2} \rceil), \ldots, (x(v) - \lceil \frac{p-4}{2} \rceil, y(v) - \lceil \frac{p-4}{2} \rceil)$ and $(x(v) + \lfloor \frac{p-4}{2} \rfloor, y(v) - \lfloor \frac{p-4}{2} \rfloor), \ldots, (x(v) + \lfloor \frac{p-4}{2} \rfloor, y(v) + \lfloor \frac{p-6}{2} \rfloor)$ as the inc-points for the incoming edges of v_k. If $inc(v_k) \geq 7$, then this means an improvement of a factor 2 in the height. We can apply the same technique for vertex v_n.

If the d-planar graph G is not triconnected, then we can use the algorithm of [22] to augment the graph by adding edges such that it is triconnected. Using a modification of [22], we can construct a linear algorithm which adds at most 2 times the optimal number of edges. If we want to minimize the maximum degree, then we can use the linear augmentation algorithm of [23], leading to a triangulated planar graph of G, for which the maximum degree is at most $\frac{3}{2}d + O(1)$. Indeed, there are planar graphs G for which any augmentation to a triconnected planar graph G' yields a maximum degree of at least $\frac{3}{2}d + O(1)$ (by modificating theorem 3.2 of [23]). It is well known that every triangulated planar graph is triconnected, hence using the algorithm of [23], we can draw all planar graphs on an $(2n-6) \times (3n-6)$ grid with at most $5n - 15$ bends and minimum angle $> \frac{2}{3d} + O(1)$.

In section 6 we can improve the grid size in several cases. When (c_i, c_{i+1}) has not slope $+1$ and (c_{j-1}, c_j) has not slope -1, then $c_{i+1}, \ldots, c_r$ need not to be shifted to the right. We can also draw the horizontal edges with length 1 initially. When adding a vertex v_k in a later step this length is only increased to 2 if $|x(c_j) - x(c_i)| + |y(c_j) - y(c_i)|$ becomes even, with c_i and c_j the left- and rightpoint of v_k, respectively.

The optimization techniques for using a smaller grid for the visibility representations are already described in section 7.2.

8 Final remarks and open problems

In this paper, a new ordering of the vertices and faces of a triconnected planar graph is introduced. This ordering leads to various algorithms to draw a planar graph on a grid. In some cases considerable improvements on existing results are obtained, in other cases new bounds are achieved. All algorithms can be implemented by straightforward techniques to run in linear time and space. Also the resulting pictures seems to be readable, if one compares them with the existing drawing algorithms (see the enclosed figures).

However, there are some open problems. We mention here:

- Decrease some of the bounds with respect to the grid size, number of bends

or minimum angle, given in this paper.

- Is it possible to use the *lmc*-ordering in other algorithms as well, to obtain better results in planar graph drawings on a grid?
- Inspect the more combinatorial aspects of the *lmc*-ordering. Very recently, the *st*-ordering of biconnected planar graphs lead to a new characterization of planar graphs [28]. We believe that also for the *lmc*-ordering, more combinatorial observations can be made than those, given in this paper.
- Can an arbitrary planar graph be drawn planar with straight lines such that the minimum angle is $> \frac{1}{d}$ (see also [26] for this question)?
- Devise a dynamic algorithms for the sequential algorithms of this paper. Very recently, in [5] a dynamic framework for graph drawing problems is described, but this approach seems not to work here. Also the parallel implementation is an interesting subject for further study.

Acknowledgements

The author wishes to thank Hans Bodlaender for the proof of theorem 1.1. He also thanks Hans Bodlaender and Jan van Leeuwen for reading earlier drafts.

References

[1] Bhasker, J., and S. Sahni, A linear algorithm to find a rectangular dual of a planar triangulated graph, *Algorithmica* 3 (1988), pp. 247–178.

[2] Booth, K.S., and G.S. Lueker, Testing for the consecutive ones property, interval graphs and graph planarity testing using PQ-tree algorithms, *J. of Computer and System Sciences* 13 (1976), pp. 335–379.

[3] Chiba, N., T. Nishizeki, S. Abe and T. Ozawa, A linear algorithm for embedding planar graphs using PQ-trees, *J. of Computer and System Sciences*, Vol. 30 (1985), pp. 54–76.

[4] Chiba, N., T. Yamanouchi, and T. Nishizeki, Linear algorithms for convex drawings of planar graphs, in: *Progress in Graph Theory*, J.A. Bondy and U.S.R. Murty (Eds.), Academic Press, 1984, pp. 153–173.

[5] Cohen, R.F., G. Di Battista, R. Tamassia, I.G. Tollis and P. Bertolazzi, A Framework for dynamic graph drawing, in: *Proc. 8th Annual ACM Symp. on Computational Geometry*, Berlin, 1992, pp. 261–270.

[6] Chrobak, M., and T.H. Payne, *A Linear Time Algorithm for Drawing Planar Graphs on the Grid*, Tech. Rep. UCR-CS-90-2, Dept. of Math. and Comp. Science, University of California at Riverside, 1990.

[7] Di Battista, G., and R. Tamassia, Incremental planarity testing, in: *Proc. 30th Annual IEEE Symp. on Found. of Comp. Science*, North Carolina, 1989, pp. 436–441.

[8] Di Battista, G., R. Tamassia and I.G. Tollis, Area requirement and symmetry display in drawing graphs, *Discr. and Comp. Geometry* 7 (1992), pp. 381–401.

[9] Eades, P., and R. Tamassia, *Algorithms for Automatic Graph Drawing: An Annotated Bibliography*, Dept. of Comp. Science, Brown Univ., Technical Report CS-89-09, 1989.

[10] Even, S., and R.E. Tarjan, Computing an *st*-numbering, *Theoret. Comp. Science* 2 (1976), pp. 436–441.

[11] Fáry, I., On straight lines representations of planar graphs, *Acta Sci. Math. Szeged*, 11 (1948), pp. 229–233.

[12] Formann, M., T. Hagerup, J. Haralambides, M. Kaufmann, F.T. Leighton, A. Simvonis, E. Welzl and G. Woeginger, Drawing graphs in the plane with high resolution, *Proc. 31th Ann. IEEE Symp. on Found. of Comp. Science*, St. Louis, 1990, pp. 86–95.

[13] Fraysseix, H. de, J. Pach and R. Pollack, How to draw a planar graph on a grid, *Combinatorica* 10 (1990), pp. 41–51.

[14] Fraysseix, H. de, and P. Rosenstiehl, A depth first characterization of planarity, *Annals of Discrete Math.* 13 (1982), pp. 75–80.

[15] Haandel, F. van, *Straight Line Embeddings on the Grid*, Dept. of Comp. Science, M.Sc. Thesis, no. INF/SCR-91-19, Utrecht University, 1991.

[16] He, X., On finding the rectangular duals of planar triangulated graphs, *SIAM J. Comput.*, to appear.

[17] Hopcroft, J., and R.E. Tarjan, Dividing a graph into triconnected components, *SIAM J. Comput.* 2 (1973), pp. 135–158.

[18] Hopcroft, J., and R.E. Tarjan, Efficient planarity testing, *J. ACM* 21 (1974), pp. 549–568.

[19] Jayakumar, R., K. Thulasiraman, and M.N.S. Swamy, Planar embedding: linear-time algorithms for vertex placement and edge ordering, *IEEE Trans. on Circuits and Systems* 35 (1988), pp. 334–344.

[20] Nummenmaa, J., Constructing compact rectilinear planar layouts using canonical representation of planar graphs, *Theoret. Comp. Science* 99 (1992), pp. 213–230.

[21] Kant, G., Hexagonal Grid Drawings, in: *Proc. 18th Intern. Workshop on Graph-Theoretic Concepts in Comp. Science WG'92*, Lecture Notes in Comp. Science, Springer-Verlag, 1992 (to appear).

[22] Kant, G., and H.L. Bodlaender, Planar graph augmentation problems, Extended Abstract in: F. Dehne, J.-R. Sack and N. Santoro (Eds.), *Proc. 2nd Workshop on Data Structures and Algorithms*, Lecture Notes in Comp. Science 519, Springer-Verlag, 1991, pp. 286–298.

[23] Kant, G., and H.L. Bodlaender, Triangulating planar graphs while minimizing the maximum degree, in: O. Nurmi and E. Ukkonen (Eds.), *Proc. 3rd Scand. Workshop on Algorithm Theory*, Lecture Notes in Comp. Science 621, Springer-Verlag, 1992, pp. 258–271.

[24] Lempel, A., S. Even and I. Cederbaum, An algorithm for planarity testing of graphs, *Theory of Graphs, Int. Symp. Rome* (1966), pp. 215–232.

[25] Lin, Y.-L., and S.S. Skiena, *Complexity Aspects of Visibility Graphs*, Manuscript, Dept. of Comp. Science, State Univ. of New York, Stony Brook, 1992.

[26] Malitz, S., and A. Papakostas, On the Angular Resolution of Planar Graphs, in: *Proc. 24th Annual ACM Symp. Theory of Computing*, Victoria, 1992.

[27] Otten, R.H.J.M., and J.G. van Wijk, Graph representation in interactive layout design, in *Proc. IEEE Int. Symp. on Circuits and Systems*, 1978, pp. 914–918.

[28] Rosenstiehl, P., H. de Fraysseix, and P. de Mendez, personal communication, 1992.

[29] Rosenstiehl, P., and R.E. Tarjan, Rectilinear planar layouts and bipolar orientations of planar graphs, *Discr. and Comp. Geometry* 1 (1986), pp. 343–353.

[30] Schnyder, W., Embedding planar graphs on the grid, in: *Proc. 1st Annual ACM-SIAM Symp. on Discr. Alg.*, San Francisco, 1990, pp. 138–147.

[31] Storer, J.A., On minimal node-cost planar embeddings, *Networks* 14 (1984), pp. 181–212.

[32] Tamassia, R., On embedding a graph in the grid with the minimum number of bends, *SIAM J. Comput.* 16 (1987), pp. 421-444.

[33] Tamassia, R., G. Di Battista and C. Batini, Automatic graph drawing and readability of diagrams, *IEEE Trans. on Systems, Man and Cybernetics* 18 (1988), pp. 61–79.

[34] Tamassia, R., and I.G. Tollis, Efficient embedding of planar graphs in linear time, *Proc. IEEE Int. Symp. on Circuits and Systems*, Philadelphia, pp. 495–498, 1987.

[35] Tamassia, R., and I.G. Tollis, A unified approach to visibility representations of planar graphs, *Discr. and Comp. Geometry* 1 (1986), pp. 321–341.

[36] Tamassia, R., I.G. Tollis and J.S. Vitter, Lower bounds for planar orthogonal drawings of graphs, *Inf. Proc. Letters* 39 (1991), pp. 35–40.

[37] Thomassen, C., Planarity and duality of finite and infinite planar graphs, *J. Comb. Theory Series B*, Vol. 29 (1980), pp. 244-271.

[38] Tutte, W.T., Convex representations of graphs, *Proc. London Math. Soc.* 10 (1960), pp. 302–320.

[39] Tutte, W.T., How to draw a graph, *Proc. London Math. Soc.* 13 (1963), pp. 743–768.

[40] Wagner, K., Bemerkungen zum vierfarbenproblem, *Jber. Deutsch. Math.-Verein* 46 (1936), pp. 26–32.

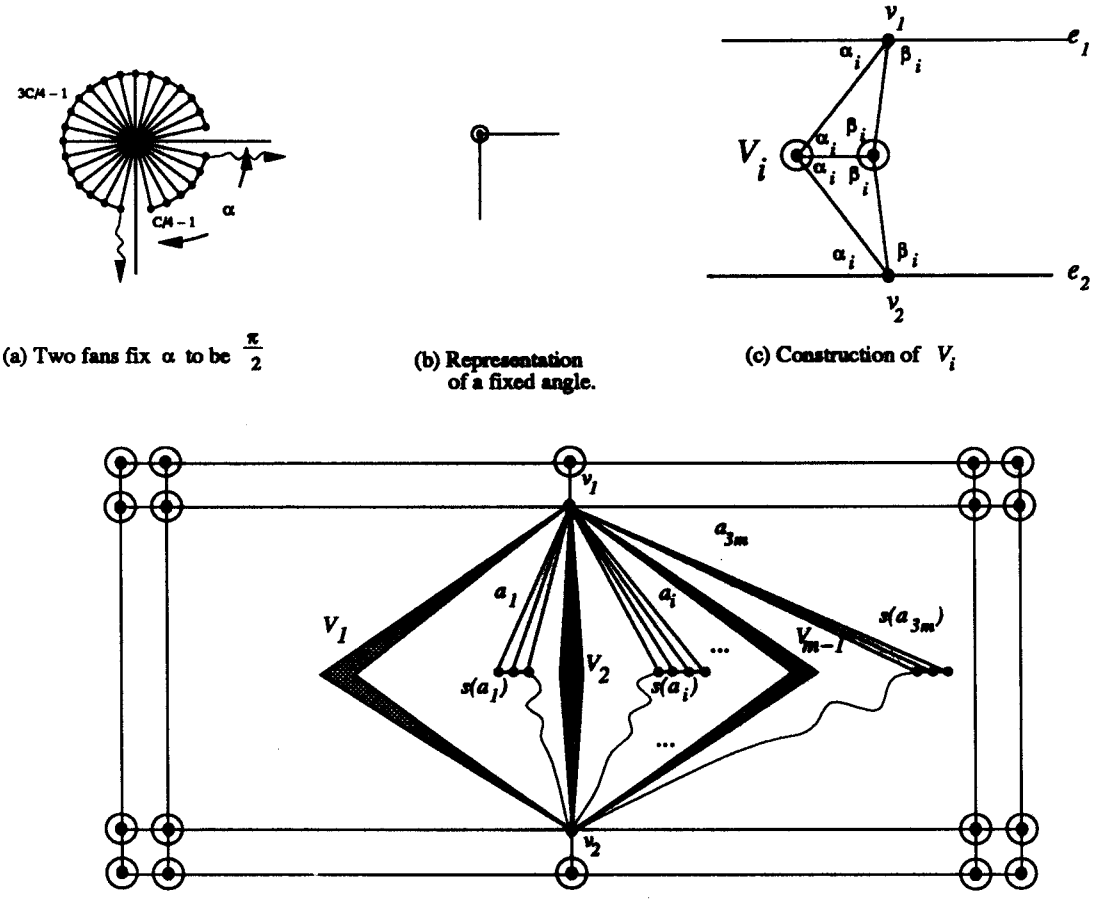
A Appendix

Theorem A.1 *Deciding whether a biconnected planar graph can be drawn planar with straight lines with minimum angle $\geq K$ is NP-hard.*

Proof: To prove NP-hardness, we show that 3-PARTITION (which is well known to be NP-complete in the strong sense) is reducible to the problem whether a biconnected planar graph can be drawn planar with straight lines such that the minimum angle is $\geq K$. Let an instance of 3-PARTITION be given, i.e., a set A of $3m$ elements $a_1, \ldots, a_{3m}$, a bound $B \in \mathbb{Z}^+$ and a size $s(a_i) \in \mathbb{Z}^+$ for each $a_i \in A$ such that $B/4 < s(a_i) < B/2$ and $\sum_{a_i \in A} s(a_i) = mB$. The question is whether A can be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$ such that, for $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B$ (note that each A_i must therefore contain exactly three elements from A). We construct a biconnected planar graph for which we will prove that there is a 3-PARTITION, if and only if there exists a planar drawing with minimum angle $\geq \frac{2\pi}{C}$, for some constant C to be filled in later. Given C, the placement of several edges of the graph can be fixed by introducing *fans*, as shown in figure 16(a). If we draw this graph with $\alpha < \frac{\pi}{2}$, then there must be an angle with size $\frac{4\alpha}{C} < \frac{2\pi}{C}$, since there are $\frac{C}{4}$ angles included in it. Similar if we draw the graph with $\alpha > \frac{\pi}{2}$, hence the angle α is fixed by adding the two fans. We denote this fixed angle as shown in figure 16(b). The size $\frac{2\pi}{C}$ is called a *unit*. From every fan one additional edge to an arbitrary vertex in that face is added to ensure the biconnectivity and to ensure that this fan belongs to the corresponding face.

Now we first construct two rectangles with fixed angles inside each other, as shown in figure 16(d). The idea is to split the remaining adjacent angle of v_1 of size π into gaps of $B+1$ units. Between each gap a subgraph V_i will be added, with an angle of 1 unit adjacent to v_1. So π will be divided into $m(B+1)+m-1 = 2m+mB-1$ units. Let $C = 2(2m + mB - 1)$ and m and B both odd, then C is divisible by 4. One unit has size $\frac{\pi}{2m+mB-1}$. Each subgraph V_i consists of an edge (a, b), with a and b both connected to v_1 and v_2. We fix $\alpha_i = i(B+1) + i$ units and $\beta_i = (m-i)(B+1) + (m-i) - 2$ units, by adding corresponding fans. It follows by geometric arguments (Z-angles) that all angles, denoted with an α_i in figure 16(c) are equal, as well as all angles, denoted with β_i. Furthermore it follows that if we want to draw this subgraph V_i with the minimum angle as wide as possible, then we should draw (a, b) parallel to e_1 and e_2 and on equal distance from v_1 and v_2 (see figure 16(c)).

Finally we add $3m$ fans $a_1, \ldots, a_{3m}$ to v_1 with size $s(a_i)$, $1 \leq i \leq 3m$, and we add an edge from each fan to v_2. See figure 16(d) for the complete graph. These $3m$ fans must be placed in the gaps of size $B+1$ between the subgraphs V_i. A drawing with minimum angle $\geq K = \frac{\pi}{2m+mB-1}$ is now possible if and only if every gap is split up into exact $B+1$ angles. Hence a planar drawing with minimum angle $\geq K$ is possible if and only if for each gap, exactly three fans $a_{i_1}, a_{i_2}, a_{i_3}$ are placed with total size $s(a_{i_1}) + s(a_{i_2}) + s(a_{i_3}) = B$, i.e., if and only if there exists a partition of A



(a) Two fans fix α to be $\frac{\pi}{2}$

(b) Representation of a fixed angle.

(c) Construction of V_i

(d) Construction of the graph for the NP–hardness proof.

Figure 16: Construction of the graph.

into m disjoint sets $A_1, \ldots, A_m$ such that for all i, $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B$.

As G can be constructed in time, polynomial in m and B, there is a polynomial time transformation form the the strongly NP-complete 3-PARTITION problem to the problem of drawing a biconnected planar graph planar in the plane with wide angles, hence the latter is NP-hard. □