International Journal of Computational Geometry & Applications

CONVEX GRID DRAWINGS OF 3-CONNECTED PLANAR GRAPHS

MAREK CHROBAK*
Department of Computer Science
University of California
Riverside, CA 92521
e-mail: marek@cs.UCR.edu

and

GOOS KANT†
Department of Computer Science
Utrecht University
Padualaan 14, 3584 CH Utrecht
the Netherlands

Received 3 February 1994
Revised 29 March 1996
Communicated by H. Imai

ABSTRACT

We consider the problem of embedding the vertices of a plane graph into a small (polynomial size) grid in the plane in such a way that the edges are straight, non-intersecting line segments and faces are convex polygons. We present a linear-time algorithm which, given an n-vertex 3-connected plane graph G (with $n \geq 3$), finds such a straight-line convex embedding of G into a $(n-2) \times (n-2)$ grid.

Keywords: Graph drawing, planar graphs, computational geometry, convexity

1. Introduction

In this paper we consider the problem of æsthetic drawing of plane graphs, that is, planar graphs that are already embedded in the plane. What is exactly an æsthetic drawing is not precisely defined and, depending on the application, different criteria have been used. In this paper we concentrate on the two following criteria: (a) edges should be represented by straight-line segments, and (b) faces should be drawn as convex polygons.

Fáry,[5] Stein[12] and Wagner[16] showed, independently, that each planar graph can be drawn in the plane in such a way that the edges are straight-line segments. Re-

*Research supported by the NSF grants CCR-9112067 and CCR-9503498.
†Research supported by ESPRIT Basic Research Actions of the EC under contract No. 7141 (project ALCOM II).

cently, there has been a lot of interest in *algorithms* that construct such embeddings, which are often referred to as simply *straight-line embeddings.* Straightforward algorithms that follow the proofs in Refs. [5,12,16] can be efficiently implemented, but they require floating-point arithmetic, which leads to a number of problems. First, small numerical errors can lead to an incorrect embedding, e.g., line intersections may not be detected. Second, when the embedding has to be drawn on a raster device, real vertex coordinates have to be mapped to integer grid points, and there is no guarantee that a correct embedding will be obtained after rounding.

It would be more convenient and practical to map the vertices into a small integer grid using only integer arithmetic, thereby avoiding roundoff errors and facilitating display on a screen. Also, this approach guarantees, automatically, that the resulting picture has fairly good proportions. We will refer to such embeddings as *grid embeddings* or *grid drawings.*

Rosentiehl and Tarjan[8] posed the question of whether it is always possible to find such an embedding into a polynomial-size grid. Later, de Fraysseix, Pach and Pollack[4] indeed gave a method that embeds an n-vertex planar graph into the $(2n-4)\times(n-2)$ grid in $O(n \log n)$ time. Chrobak and Payne[3] provided a linear-time implementation of their method. Schnyder[10] discovered a different method, based on so-called *barycentric* representations of graphs and some interesting combinatorial interpretation of vertex coordinates. His algorithm can be implemented in linear time and reduces the grid size to $(n-2)\times(n-2)$. Schnyder also pointed out[9] that the method from Ref. [3] can be modified to yield a smaller embedding into the $(n-2)\times(n-2)$ grid. (Throughout the paper we assume that $n \geq 3$.)

As for the lower bound, de Fraysseix, Pach and Pollack[4] present an example of a plane graph that requires a grid of size at least $2n/3 \times 2n/3$. A major open problem in this area is whether a $cn \times cn$ grid can be used for this purpose, for some $c < 1$.

The algorithms discussed above assume that the input graph is triangulated. If we want to use them to draw an arbitrary plane graph G, we have to extend it to a triangulated graph G', embed G', and then remove the added edges. The resulting faces can have very complex, irregular shapes. A more æsthetic embedding can be obtained by drawing faces as convex polygons. This can always be done if G is 3-connected, as proved by Tutte in Ref. [14]. In fact, it can be done even for some plane graphs which are not 3-connected, see Refs. [13,14,15]. Chiba *et al.* [2] developed a linear-time algorithm that draws convexly all planar graphs for which it is possible. For arbitrary 2-connected graphs, Chiba *et al.* ,[1] presented linear-time algorithms for producing æsthetic drawings that make the resulting picture "as convex as possible", in a sense that is precisely defined in Ref. [1]. On the other hand, it is NP-complete to decide whether a biconnected planar graph can be drawn with at least K convex faces.[7]

Recently, Kant[6] developed a method for constructing convex grid drawings of 3-connected plane graphs in linear time. His algorithm, related to those of Refs. [4] and [3], uses a $(2n-4)\times(n-2)$ grid.

In this paper we will show how to construct convex drawings of 3-connected plane graphs into a smaller, $(n-2)\times(n-2)$, grid in linear time. Our algorithm has been inspired by the ideas from Refs. [3,4,6,9]. It is very easy to implement and, in fact, in the paper we present a Pascal-like description of the algorithm.

A different convex embedding method for 3-connected planar graphs, using the $(n-1) \times (n-1)$ grid, was announced recently by Schnyder and Trotter.[11]

2. Algorithm for Convex Drawings

We introduce first the concept of a canonical decomposition, which generalizes canonical orderings defined in Ref. [4] for triangulated graphs.

Define a plane graph G to be *internally 3-connected* if (a) G is 2-connected, and (b) if removing two vertices u, v disconnects G then u, v belong to the outer face and each connected component of $G - \{u, v\}$ has a vertex on the outer face. In other words, G is internally 3-connected iff it can be extended to a 3-connected graph by adding a vertex and connecting it to all vertices on the outer face.

Canonical Decompositions. Let G be an n-vertex 3-connected plane graph with an edge (v_1, v_2) on the external face. Let $\pi = (V_1, \ldots, V_m)$ be an ordered partition of V, that is, $V_1 \cup \ldots V_m = V$ and $V_i \cap V_j = \emptyset$ for $i \neq j$. Define G_k to be the subgraph of G induced by $V_k \cup \ldots \cup V_m$, and denote by C_k the external face of G_k. We say that π is a *canonical decomposition of* G *with bottom edge* (v_1, v_2) if:

(CD1) V_1 is a singleton, $\{z_0\}$, where z_0 lies on the outer face and $z_0 \notin \{v_1, v_2\}$.

(CD2) C_m is a face of G, and each C_k is a cycle containing (v_1, v_2).

(CD3) Each G_k is 2-connected and internally 3-connected.

(CD4) For each k in $\{2, \ldots, m-1\}$, one of the two following conditions holds:

- (a) V_k is a singleton, $\{z\}$, where z belongs to C_k and has at least one neighbor in $G - G_k$.
- (b) V_k is a chain, $(z_1, \ldots, z_\ell)$, where each z_i has at least one neighbor in $G - G_k$, and where z_1 and z_ℓ each have one neighbor on C_{k+1}, and these are the only two neighbors of V_k in G_{k+1}.

Throughout the rest of the paper we will simply call π a *canonical decomposition*, since the bottom edge (v_1, v_2) will always be understood from context.

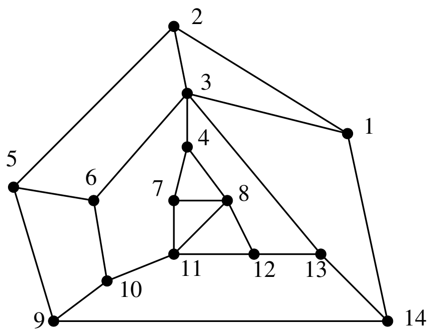
In Figure 1 an example of a canonical decomposition of a triconnected planar graph is given.

We will commonly view C_k as a path $(w_1, w_2, \ldots, w_j)$ (instead of a cycle) starting with $w_1 = v_1$ and ending with $w_j = v_2$, ignoring the edge (v_1, v_2). We will also view C_k as being ordered from "left" to "right", where w_1 is the leftmost and w_j is the rightmost vertex on C_k.

We will use the following lemma proved by Kant[6]:

Lemma 1 *Each 3-connected plane graph has a canonical decomposition.*

Proof. (Sketch) Pick an edge (v_1, v_2) and a vertex $z_0 \notin \{v_1, v_2\}$ on the outer face of G. Let $V_1 = \{z_0\}$.



m	V_m
1	1
2	2
3	3
4	4
5	5, 6
6	7
7	8
8	9 - 14

Figure 1: The canonical decomposition with bottom edge $(9, 14)$.

Suppose that we have already defined $V_1, \ldots, V_{k-1}$. If G_k is 3-connected, let V_k be $\{z\}$, where z is an arbitrary vertex from $C_k - \{v_1, v_2\}$ that has a neighbor in $G - G_k$.

Otherwise, if C_k contains a chain whose removal does not destroy 2-connectivity, let V_k be a maximal such chain — its members will have degree 2 in G_k (and will have a neighbor in $G - G_k$ by the 3-connectivity of G), and its two neighbors will have greater degree.

If, however, no such chain exists, pick two vertices in C_k whose removal disconnects G_k that are as close to each other as possible in the ordering of C_k. The triconnected component in between, by the 3-connectivity of G, contains a vertex z having a neighbor in $G - G_k$. Let V_k be $\{z\}$. $\square$

As it was shown by Kant[6] (Theorem 2.3) a canonical decomposition can be constructed in linear time.

Members of V_k are said to have *rank* k. A vertex in C_k is said to be *saturated in* G_k iff it has no neighbors outside G_k, i.e., no neighbors of smaller rank.

Given $C_k = (v_1 = w_1, \ldots, w_j = v_2)$, let $1 \leq a < b \leq j$ be such that w_a and w_b are not saturated in G_k but all vertices w_i, for $a < i < b$, are already saturated. Pick $a \leq c < b$ such that w_c has largest rank, and if there are two vertices with highest rank pick the left one. (From the definition of the canonical decomposition it can be shown that there are at most two such vertices.) Then we define $\mu_k^+(a) = \mu_k^-(b) = c$. We will often omit subscript k, for simplicity. Note that if $b = a + 1$ then $\mu_k^+(a) = \mu_k^-(b) = a$.

If a, b are as defined above, then the path $w_a, \ldots, w_b$ in G is a part of a facial cycle F, that also contains two edges that connect w_a and w_b with $G - G_k$, plus possibly some other edges in $G - G_k$. Intuitively, the algorithm will work in such a way that one of $w_\mu, w_{\mu+1}$, with $\mu = \mu^+(a)$, will be a lowest vertex on F (that is, it will have the smallest y-coordinate), and thus stretching edge $(w_\mu, w_{\mu+1})$, together with some other edge on the upper side of F, will not destroy convexity of F.

Our algorithm will be to add sets V_k, one by one, in reverse order, $V_m, \ldots, V_1$, adjusting the embedding at every step. By $\mathcal{E}(v)$ we will denote the current position of vertex v on the grid, i.e., $\mathcal{E}(v) = (x(v), y(v))$. By $\mathcal{E}(u, v)$ we denote the embedding of edge (u,v), that is, the line segment that connects $\mathcal{E}(u)$ with $\mathcal{E}(v)$. To each vertex w we assign a set of vertices, $\mathit{Under}(w)$, that will contain certain vertices that are located below w and have to be shifted right whenever w is shifted right. The precise definition of $\mathit{Under}(w)$ is part of the algorithm and is given below.

We will describe first an algorithm that uses the $(n-1)\times(n-1)$ grid, and then show how to improve it to $(n-2)\times(n-2)$.

Algorithm ConvexDraw. We initialize the embedding by drawing $C_m = (v_1 = z_1, z_2, \ldots, z_\ell = v_2)$ as follows:

$\mathcal{E}(z_1) := (0,0)$;
$\mathcal{E}(z_\ell) := (\ell-1,0)$;
$\mathcal{E}(z_i) := (i-1,1)$ for all $i = 2,\ldots,\ell-1$;
$\mathit{Under}(z_i) := \{z_i\}$ for all $i = 1,\ldots,\ell$.

Then, for each $k = m-1,\ldots,1$, we do the following. Let $C_{k+1} = (v_1 = w_1,\ldots,w_j = v_2)$ be the contour of G_{k+1}. Let $V_k = (z_1,\ldots,z_\ell)$. V_k may be a singleton or a chain, but in the algorithm we will treat both cases uniformly.

Let w_p and w_q be the leftmost and rightmost neighbors of V_k in G_{k+1}. Let $\alpha = \mu^+(p)$ and $\beta = \mu^-(q)$. Note that if V_k is a chain then all vertices $w_{p+1},\ldots,w_{q-1}$ that are being covered belong to one face and must have been saturated by now. Consequently, we will have $\alpha = \beta$. If V_k is a singleton (of degree at least 3), $V_k = \{z_1\}$, then all vertices among $w_{p+1},\ldots,w_{q-1}$ which are not neighbors of z_1 must have been saturated. In this case, we have $\alpha < \beta$. In fact, w_α and w_β will belong to different faces: to the first and last face that are created when adding z_1, respectively.

We now execute the following steps:

Update: We update $\mathit{Under}(w_p)$, $\mathit{Under}(w_q)$ and compute all $\mathit{Under}(z_i)$ as follows:

$$\begin{aligned}
\mathit{Under}(w_p) &:= \bigcup_{i=p}^{\alpha} \mathit{Under}(w_i);\\
\mathit{Under}(w_q) &:= \bigcup_{i=\beta+1}^{q} \mathit{Under}(w_i);\\
\mathit{Under}(z_1) &:= \{z_1\} \cup \bigcup_{i=\alpha+1}^{\beta} \mathit{Under}(w_i);\\
\mathit{Under}(z_i) &:= \{z_i\}\,;\quad i = 2,\ldots,\ell.
\end{aligned}$$

Shift: For each $u \in \bigcup_{i=q}^{j} \mathit{Under}(w_i)$ do

$$x(u) := x(u) + \ell$$

Install V_k: Let ϵ be 0 if w_p is saturated in G_k, and 1 otherwise. For each $i = 1,\ldots,\ell$, let $\mathcal{E}(z_i)$ be defined by

$$\begin{aligned}
x(z_i) &:= x(w_p) + i - 1 + \epsilon\\
y(z_i) &:= y(w_q) + x(w_q) - x(w_p) - \ell + 1 - \epsilon
\end{aligned}$$

In other words, we draw the V_k horizontally, in such a way that the slope of the segment $\mathcal{E}(z_\ell, w_q)$ is $-45°$. Vertex z_1 is placed above w_p if w_p is saturated, and at the next x-coordinate otherwise. Note that in the last formula we use the new updated value of $x(w_q)$.

This completes the description of the algorithm. Now we will prove its correctness (see Figure 2 for an illustration).

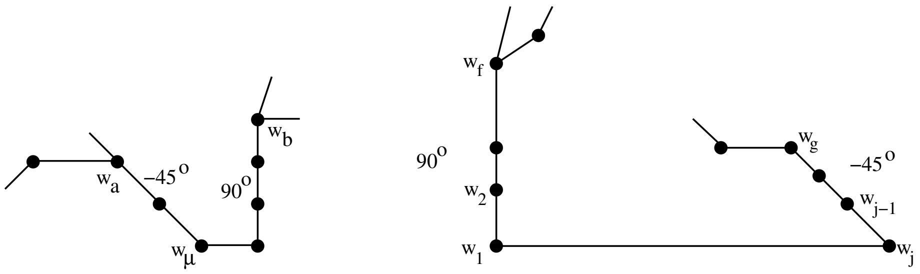


Figure 2: Illustration of Lemma 2.

Lemma 2 *Let $1 \le k \le m$, and $C_k = (v_1 = w_1, w_2, \ldots, w_j = v_2)$. Then $\mathcal{E}(v_1) = (0,0)$, $\mathcal{E}(v_2) = (k-1, 0)$, and all contour segments $\mathcal{E}(w_i, w_{i+1})$, $i = 1, \ldots, j-1$, have slopes in $\{-45°, 0°\} \cup [45°, 90°]$.*

More specifically the following conditions hold:

(a) if w_a, w_b are two non-saturated vertices such that all w_i, for $a < i < b$, are saturated, and if $\mu = \mu^+(a)$, then the line segments on C_k from w_a to w_b satisfy the following condition: The first $\mu - a$ segments have slope $-45°$ and the last $b - \mu - 1$ segments have slopes $90°$. The segment $\mathcal{E}(w_\mu, w_{\mu+1})$ has slope in $\{-45°, 0°\} \cup [45°, 90°]$, except that it cannot be $90°$ for $\mu = a$.

(b) If w_f is the first non-saturated vertex on C_k then all slopes on the path $w_1, \ldots, w_f$ are $90°$.

(c) If w_g is the last non-saturated vertex on C_k then all slopes on the path $w_g, \ldots, w_j$ are $-45°$.

Note that the lemma implies that, in particular, if $b = a + 1$ (and thus $\mu = a$), then the slope of $\mathcal{E}(w_a, w_b)$ belongs to $\{-45°, 0°\} \cup [45°, 90°)$.

Proof. (a) Backward induction on k. For $k = m$, the lemma is obvious. Fix $k \in \{1, \ldots, m-1\}$, and assume that the lemma holds for $k' = k + 1$. We will show that it also holds for k. Let w_p and w_q denote, as usual, the leftmost and rightmost neighbors of V_k in C_{k+1}. Clearly, both w_p and w_q are not saturated in G_{k+1}.

Let $V_k = (z_1, \ldots, z_\ell)$. Then the new contour is

$$C_{k+1} = (w_1, \ldots, w_p, z_1, \ldots, z_\ell, w_q, \ldots, w_j) = (w'_1, w'_2, \ldots, w'_{j+\xi}),$$

for $\xi = \ell - q + p + 1$. By the definition of the canonical decomposition, each z_i is not saturated in G_k, and therefore the lemma holds for all segments $\mathcal{E}(z_i, z_{i+1})$. Thus it is sufficient to prove that the lemma holds for vertices in the chains $w_1, \ldots, z_1$ and $z_\ell, \ldots, w_j$ of C_{k+1}.

We consider first the chain $w_1, \ldots, z_1 = w'_1, \ldots, w'_{p+1}$. If w_p is not saturated in G_k, then the lemma holds for the sub-chain $w_1, \ldots, w_p$ by induction and for w_p, z_1 by the algorithm.

Thus we can assume now that w_p becomes saturated in G_k. If all vertices $w_1, \ldots, w_p$ are saturated, then the chain $w_1, \ldots, w_p, z_1$ satisfies (b) with $f = p + 1$, by induction and by the fact that the slope of $\mathcal{E}(w_p, z_1)$ is $90°$. Otherwise, pick a non-saturated vertex w_a, $1 \le a < p$, that is closest to w_p. The lemma holds, by induction, for the chain $w_1, \ldots, w_a$. For $w_a, \ldots, w_p, z_1$, the lemma follows from the inductive assumption about $w_a, \ldots, w_p$, since the slope of $\mathcal{E}(w_p, z_1)$ is $90°$, and $\mu^+_k(a) = \mu^+_{k+1}(a)$.

The proof for the other chain, $z_\ell, w_q, \ldots, w_j = w'_{p+\ell} \cdots w'_{j+\xi}$, is similar. Let $r = p + \ell$. If $w'_{r+1} = w_q$ is not saturated in G_k, the lemma follows directly by induction.

Thus suppose that w_q becomes saturated in G_k. If all vertices $w_q, \ldots, w_j$ are saturated, then the chain $w'_r, \ldots, w'_{j+\xi} = z_\ell, w_q, \ldots, w_j$ satisfies (c) with $g = r$, by induction and by the fact that the slope of $\mathcal{E}(z_\ell, w_q)$ is $-45°$. Otherwise, pick a non-saturated vertex w_b, $q < b \le j$ that is closest to w_q. The lemma holds, by induction, for the chain $w_b, \ldots, w_j$. For $z_\ell, w_q, \ldots, w_b$, the lemma follows from the inductive assumption about $w_q, \ldots, w_b$, since the slope of $\mathcal{E}(z_\ell, w_q)$ is $-45°$, and $\mu^+_k(r) = \mu^+_{k+1}(q)$. □

The lemma above implies immediately that adding V_k does not destroy the embedding, as stated in the corollary below.

Corollary 1 *For each k, when we add V_k, then, after applying the shift operation, all neighbors of V_k are visible, that is the edges between V_k and C_{k+1} do not intersect themselves or edges in C_{k+1}.*

Whenever we add a vertex z (singleton or a member of a chain), we place it at the y-coordinate which is not smaller than the y-coordinate of its neighbors that had already been embedded. Also, y-coordinates never change. Thus the next lemma follows directly from the algorithm.

Lemma 3 *At each step of the algorithm, the y-coordinates are monotone with respect to ranks, in the following sense: if (u, v) is an edge and $rank(u) > rank(v)$ then $y(v) \ge y(u)$.*

What remains to show is that we do not destroy the planarity property and convexity when we apply the shift operation. This is proven in the next two lemmas.

Let us call a plane graph *internally convex* if all its internal faces are convex.

Lemma 4 *Each G_k is straight-line embedded and internally convex. Additionally, it has the following property: Suppose $C_k = (v_1 = w_1, w_2, \ldots, w_j = v_2)$, and pick any $1 \le s \le j$, and any integer δ. If we shift all nodes in $\bigcup_{i=s}^{j} Under(w_i)$ by δ to the right, then G_k remains straight-line embedded and internally convex.*

Proof. Backward induction on k. The lemma holds for $k = m$, by inspection. Assume the lemma holds for $k' = k + 1$; we will show that the above properties are preserved when we add V_k. We use the notation from the algorithm that the contour of G_{k+1} is $C_{k+1} = (w_1, \ldots, w_j)$, and now we are about to add V_k. Let w_p and w_q be the leftmost and rightmost neighbors of V_k in C_{k+1}.

Let $V_k = (z_1, \ldots, z_\ell)$. The contour of G_k is $C_k = (w'_1, w'_2, \ldots, w'_{j+\xi})$ for $\xi = \ell - q + p + 1$, where

$$w'_i = \begin{cases} w_i & i = 1, \ldots, p \\ z_{i-p} & i = p+1, \ldots, p+\ell \\ w_{i-\xi} & i = p+\ell+1, \ldots, j+\xi. \end{cases}$$

If $s > p+\ell$, V_k doesn't move, and the lemma follows directly by induction. If $s \le p$, the lemma also follows from the inductive assumption, since V_k shifts rigidly with the rest of the graph.

Let us assume now that V_k is a singleton, $V_k = \{z_1\}$, and consider the cases $s = p+1, p+2$. Let z_1 have t neighbors among $w_p, \ldots, w_q$, and let $F_1, \ldots, F_{t-1}$ be the faces created when adding z_1.

If $s = p+1$, then we apply the inductive assumption to G_{k+1}, with $s' = \mu^+_{k+1}(p) + 1$. The straight-line embedding and internal convexity are preserved in G_{k+1} by induction. All faces $F_2, \ldots, F_{t-1}$ are shifted rigidly with G_{k+1}, only F_1 will be deformed. But in F_1 we will only stretch the edge (w_p, z_1) and $(w_{s'}, w_{s'+1})$, and by the choice of s' this will not destroy the convexity of F_1.

If $s = p+2$, the proof is similar: we apply the inductive assumption to G_{k+1} with $s'' = \mu^-_{k+1}(q) + 1$. In this case only F_{t-1} can be deformed but, by the choice of s'', the convexity of F_{t-1} will be preserved.

The proof when V_k is a chain is very similar and is left to the reader. $\square$

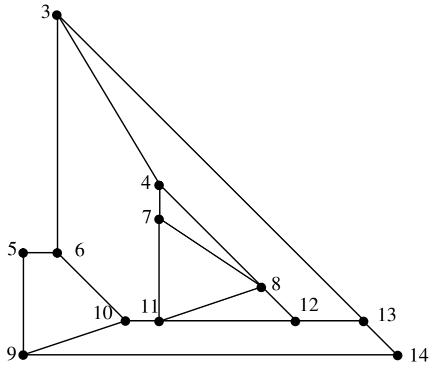
Improving the grid size. Now we sketch how to modify ConvexDraw in order to reduce the grid size to $(n-2) \times (n-2)$. First we pick z_0 to be the neighbor of v_2 different from v_1 on the outer face of G. We construct a canonical decomposition and run ConvexDraw for $m-1$ steps. In the last step, having already embedded G_2, we set $\mathcal{E}(z_0) := (1, n-2)$, and we *do not* shift any vertices to the right.

Let us call this modified algorithm *ConvexDraw2*. In order to show correctness, we only need to show that adding z_0 will result in a correct, convex embedding. By Lemma 2 and the algorithm, before adding z_0 we have $x(w_1) = x(w_2) = \ldots = x(w_p) = 0$ and $x(w_q) = n-2$, where $w_q = v_2$. The edge with slope $-45°$ from v_2 contains the point $(1, n-3)$. This implies that all vertices $w_p, \ldots, w_q$ are visible from $(1, n-2)$. The convexity of the outer face follows from the choice of z_0. Consequently, we obtain the following theorem:

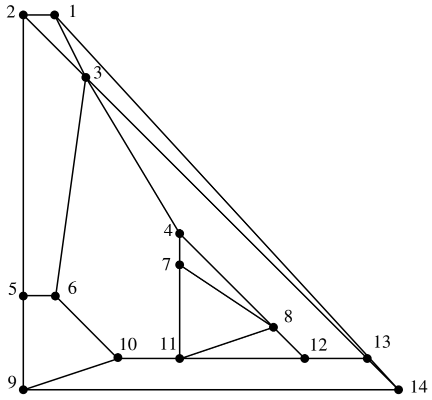
Theorem 1 *Given a 3-connected plane graph G, algorithm ConvexDraw2 constructs a straight-line convex embedding of G into the $(n-2) \times (n-2)$ grid.*

In Figure 3 an example of a drawing is given. After adding vertex 3, we have $\mathit{Under}(w) = \{w\}$ for $w \in \{5, 9, 13, 14\}$, $\mathit{Under}(6) = \{6, 10\}$ and $\mathit{Under}(3) = \{3, 4, 7, 8, 11, 12\}$. Therefore, when adding vertex 2, the vertices in $\mathit{Under}(3) \cup \mathit{Under}(13) \cup \mathit{Under}(14) = \{3, 4, 7, 8, 11, 12, 13, 14\}$ will be shifted right. After adding vertex 2, we have $\mathit{Under}(w) = \{w\}$ for $w \in \{2, 5, 9, 13, 14\}$, $\mathit{Under}(3) = \{3, 4, 6, 7, 8, 10, 11, 12\}$.

Notice that the drawing is not *strictly convex*, i.e., there are angles of size $180°$.



(a) The drawing of the graph G_3.



(b) The drawing of the graph G.

				x-coordinates of vertices													
k	V_k	w_p	w_q	1	2	3	4	5	6	7	8	9	10	11	12	13	14
8	9 - 14	–	–									0	1	2	3	4	5
7	8	11	12								3	0	1	2	4	5	6
6	7	11	8							2	4	0	1	2	5	6	7
5	5, 6	9	10					0	1	4	6	0	3	4	7	8	9
4	4	7	8				4	0	1	4	7	0	3	4	8	9	10
3	3	6	13			1	4	0	1	4	7	0	3	4	8	10	11
2	2	5	3		0	2	5	0	1	5	8	0	3	5	9	11	12
1	1	2	14	1	0	2	5	0	1	5	8	0	3	5	9	11	12
y-coordinates				12	12	10	5	3	3	4	2	0	1	1	1	1	0

Figure 3: The values of the different variables in ConvexDraw.

3. Linear-time Implementation

The linear-time implementation is achieved by representing the sets $Under(v)$ using a binary tree T. Furthermore, instead of computing absolute x-coordinates of vertices each time we add V_k, we will only maintain, for each v, its *relative* x-coordinate with respect to its father.

By $T(v)$ we denote the subtree of T rooted at v. Each node v is a record containing the following information:

$\mathit{left}(v)$	:	left child of v
$\mathit{right}(v)$	:	right child of v
$\Delta x(v)$	=	$x(v) - x(w)$, x-offset of v from its T-father w
$x(v)$	=	x-coordinate of v
$y(v)$	=	y-coordinate of v

The root of T is v_1. If v is in the contour then $\mathit{left}(v)$ is the node u such that $T(u) = Under(v) - \{v\}$, and $\mathit{right}(v)$ is the next node in the contour. Thus the path $C_k = (w_1, \ldots, w_j)$ consists of: v_1, $\mathit{right}(v_1)$, $\mathit{right}(\mathit{right}(v_1))$, etc. $Under(w_i)$ consists of w_i and its T-subtree rooted at $\mathit{left}(w_i)$. Thus we have the following relationship: $T(w_i) = \bigcup_{a=i}^{j} \mathit{Under}(w_a)$.

If v is not in the contour then *left*(v) and *right*(v) are used as work pointers to maintain the correct relationship between T and the sets $\mathit{Under}(u)$, and to minimize pointer manipulations.

If u, v are any two nodes, then let $\Delta x(u,v) = x(v) - x(u)$. In particular, $\Delta x(v) = \Delta x(u,v)$ where u is the father of v. We want to emphasize that the algorithm will store only $\Delta x(v)$ for each v; whenever the value of $\Delta x(u,v)$ is needed, where $v \neq \mathit{left}(u), \mathit{right}(u)$, it has to be computed by finding the lowest common ancestor w of u, v, adding all offsets on the path from w to v and subtracting all offsets on the path from w to u.

In terms of our tree T, when we add V_k, we need to shift $T(w_q)$ to the right. The crucial observation that leads to the linear-time algorithm is that it is not really necessary to know the exact positions of w_p and w_q at the time when we install $V_k = (z_1, \ldots, z_\ell)$. If we only know their y-coordinates and the offset $\Delta x(w_p, w_q)$ then for each $i > 1$ we can compute $y(z_i)$ and the x-offset of z_i relative to z_{i-1}, the x-offset of z_1 relative to w_p, and the x-offset of w_q relative to z_ℓ.

Algorithm FastConvexDraw. We will assume, for simplicity, that all links in T have been initialized to **nil**.

The algorithm consists of two phases. In the first phase we add new vertices, compute their x-offsets and y-coordinates. In the second phase, we traverse the tree and compute final x-coordinates by accumulating offsets.

We begin by embedding $V_m = (z_1, \ldots, z_\ell)$:

> **for** $i := 1$ **to** $\ell - 1$ **do** *right*$(z_i) := z_{i+1}$;
> $\mathcal{E}(z_1) := (0,0)$; $\mathcal{E}(z_\ell) := (\ell - 1, 0)$;
> **for** $i := 2$ **to** $\ell - 1$ **do** $\mathcal{E}(z_i) := (i-1, 1)$;

Now, for each $k = m-1, m-2, \ldots, 1$, we proceed as follows. Let $w_1, \ldots, w_j$ be the contour of G_{k+1}; and let w_p, w_q be the leftmost and rightmost neighbors of $V_k = (z_1, \ldots, z_\ell)$ in G_{k+1}. Then execute the following steps.

	$\alpha := \mu^+(p)$; $\beta := \mu^-(q)$;
Precompute offsets:	compute $\Delta_i = \Delta x(w_p, w_i)$, for $i = p+1, \ldots, q$;
Update node w_p:	**if** $\alpha > p$ (and thus $q > p+1$) **then begin**
	right$(w_\alpha) :=$ *left*(w_p);
	if *left*$(w_p) \neq$ **nil then** $\Delta x(\mathit{left}(w_p)) := \Delta x(\mathit{left}(w_p)) - \Delta_\alpha$;
	left$(w_p) :=$ *right*(w_p)
	end ;
	right$(w_p) := z_1$;
Install V_k:	**if** w_p is saturated **then** $\epsilon := 0$ **else** $\epsilon := 1$;
	$\Delta x(z_1) := \epsilon$;
	$y(z_1) := y(w_q) + \Delta_q - \ell + 1 - \epsilon$;
	for $i := 2$ **to** ℓ **do begin**
	right$(z_{i-1}) := z_i$;
	$\Delta x(z_i) := 1$;

```
                          y(z_i) := y(z_1)
                     end ;
                     right(z_ℓ) := w_q;
                     if α < β then begin
                          left(z_1) := w_{α+1};
                          Δx(w_{α+1}) := Δ_{α+1} − ε;
                          right(w_β) := nil;
                     end ;
Update node w_q:     if β + 1 < q then begin
                          right(w_{q−1}) := left(w_q);
                          Δx(left(w_q)) := Δx(left(w_q)) + Δx(w_q);
                          left(w_q) := w_{β+1};
                          Δx(w_{β+1}) := Δ_{β+1} − Δ_q;
                     end ;
                     Δx(w_q) := Δ_q − ℓ + 1 − ε;
```

At this point all y-coordinates and x-offsets have already been computed. All that remains to be done is to compute x-coordinates. In order to do so, we invoke AccumulateOffsets$(v_1, 0)$, where AccumulateOffsets is as follows:

```
procedure AccumulateOffsets(v: vertex, δ: integer);
begin
     if v ≠ nil then begin
          x(v) := δ + Δx(v);
          AccumulateOffsets(left(v), x(v));
          AccumulateOffsets(right(v), x(v))
     end
end
```

Correctness. In order to prove correctness, it is sufficient to show that FastConvexDraw is a correct implementation of ConvexDraw from the previous section.

That the sets *Under*(v) are represented correctly, as explained at the beginning of this section, follows by inspection of the pointer manipulations.

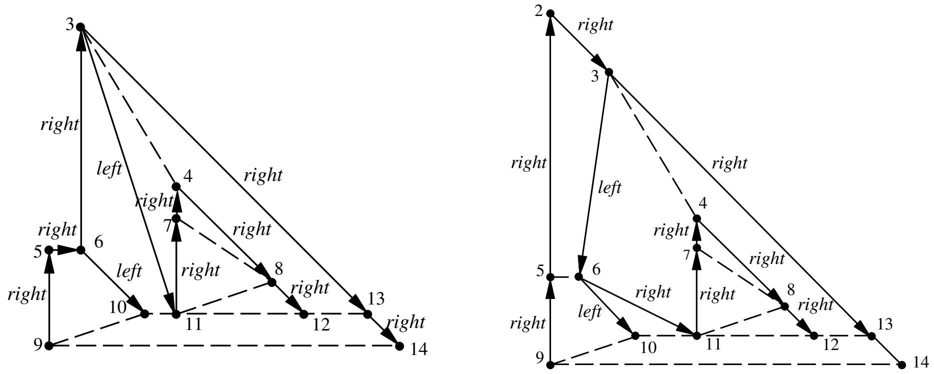
Since the x-coordinate of a vertex v equals to the sum of the offsets on the path from the root v_1 to v, it is sufficient to show that all offsets $\Delta x(v)$ are computed correctly. It is a matter of elementary algebra to verify that this is indeed true.

Complexity. As for its complexity, we have already mentioned that the canonical decomposition can be found in time $O(n)$. In the first phase, when we add $V_k = (z_1, \ldots, z_\ell)$, the cost is proportional to $\ell + q - p$, where w_p and w_q denote, as usual, the leftmost and rightmost neighbors of V_k in C_{k+1}. Thus the total cost of the first phase is proportional to the number of edges, that is, $O(n)$. The second phase can be trivially implemented to run in linear time.

Improving the grid size. In order to improve the grid size, we apply the modification outlined in the previous section. Let us call the resulting algorithm *FastConvexDraw2*. This change doesn't affect the time complexity, and thus we get the following theorem.

Theorem 2 *Given a plane graph G, algorithm FastConvexDraw2 computes a convex embedding of G into the $(n-2) \times (n-2)$ grid in $O(n)$ time.*

In Figure 4 the construction of the tree and the values of $\Delta x(v)$ are given for the example from Figure 3.



(a) The tree of the graph G_3.

(b) The tree of the graph G_2.

k	V_k	$\Delta x(v)$													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
8	9 -14									0	1	1	1	1	1
7	8								1	0	1	1	1	1	1
6	7							0	2	0	1	1	1	1	1
5	5, 6					0	1	0	2	0	2	1	1	1	1
4	4				0	0	1	0	3	0	2	1	1	1	1
3	3			0	0	0	1	0	3	0	2	3	1	2	1
2	2		0	2	0	0	-1	0	3	0	2	4	1	9	1

Figure 4: The tree T and $\Delta x(v)$.

Our algorithm can also be generalized to some graphs that are not necessarily 3-connected, using the following theorem of Thomassen:

Theorem 3 *Let G be a plane graph with outer face S such that all vertices not in S have degree ≥ 3. Then G has a convex representation with outer face S if and only if G is internally 3-connected.*

If G satisfies the assumptions in the above theorem and $S = (u_1, \ldots, u_j)$, then adding a vertex z_0 with edges to $u_1, \ldots, u_j$ gives a triconnected graph G^*. By applying the algorithm FastConvexDraw to G^*, and not adding z_0 in the last phase, we obtain a straight-line and internally convex drawing for G. This yields the following theorem:

Theorem 4 *If a plane graph G with degree ≥ 3 is convex drawable, then FastConvexDraw, modified as above, constructs in linear time an internally convex drawing of G into a $(n-1) \times (n-2)$ grid.*

Acknowledgements.

The authors wish to thank Tom Payne for useful comments.

References

1. N. Chiba, K. Onoguchi, and T. Nishizeki, "Drawing plane graphs nicely," *Acta Informatica* **22** (1985) 187–201.
2. N. Chiba, T. Yamanouchi, and T. Nishizeki, "Linear algorithms for convex drawings of planar graphs," in *Progress in Graph Theory*, eds. J.A. Bondy and U.S.R. Murty (Academic Press, 1984) pp. 153–173,
3. M. Chrobak and T. Payne, "A linear-time algorithm for drawing planar graphs," *Information Processing Letters* **54** (1995) 241–246.
4. H. de Fraysseix, J. Pach, and R. Pollack, "Small sets supporting Fáry embeddings of planar graphs," in *Proc. 20th Annual Symposium on Theory of Computing*, 1988, pp. 422–245.
5. I. Fáry, "On straight line representing of planar graphs," *Acta. Sci. Math. Szeged* **11** (1948) 229–233.
6. G. Kant, "Drawing planar graphs using the lmc-ordering," in *Proc. 33rd Symp. on Foundations of Computer Science*, 1992, pp. 101–110.
7. G. Kant, "Algorithms for Drawing Planar Graphs," PhD Thesis, Dept. of Computer Science, Utrecht University, 1993.
8. P. Rosentiehl and R.E. Tarjan, "Rectilinear planar layouts and bipolar orientations of planar graphs", *Discrete Computational Geometry* **1** (1986) 343–353.
9. W. Schnyder, personal communication, 1992.
10. W. Schnyder, "Embedding planar graphs in the grid," in *Proc. 1st Annual ACM-SIAM Symp. on Discrete Algorithms*, 1990, pp. 138–147.
11. W. Schnyder and W. Trotter, "Convex drawings of planar graphs," *Abstracts of the AMS* 92T-05-135 (1992).
12. S.K. Stein, "Convex maps," *Proc. Amer. Math. Soc.* **2** (1951) 446–466.
13. C. Thomassen, "Planarity and duality of finite and infinite planar graphs," *Journal of Combinatorial Theory B* **29** (1980) 244–271.
14. W.T. Tutte, "Convex representations of graphs," *Proc. London Math. Soc.* **10** (1960) 304–320.
15. W.T. Tutte, "How to draw a graph," *Proc. London Math. Soc.* **13** (1963) 743–768.
16. K. Wagner, "Bemerkungen zum vierfarbenproblem," *Jber. Deutsch. Math.-Verein* **15** (1936) 26–32.