A Linear-time Algorithm for Drawing a Planar Graph on a Grid

M. Chrobak   T.H. Payne
Department of Computer Science
University of California
Riverside, CA 92521

Abstract

We present a linear-time algorithm that, given an n-vertex planar graph G, finds an embedding of G into a $(2n-4) \times (n-2)$ grid such that the edges of G are straight-line segments.

1 Introduction

We consider the problem of embedding the vertices of a planar graph into a small grid in the plane in such a way that the edges are straight, non-intersecting line segments.

The existence of such *straight-line embeddings* for planar graphs was independently discovered by Fáry [Fa48], Stein [St51], and Wagner [Wa36]; this result also follows from Steinitz's theorem on convex polytopes in three dimensions [SR34]. The first algorithms for constructing straight-line embeddings [Tu63, CYN84, CON85] required high-precision arithmetic, and the resulting drawings were not very aesthetic, since they tend to produce uneven distributions of vertices over the drawing area.

Rosenstiehl and Tarjan [RT86] noticed that it would be convenient to be able to map veritices of a planar graph into a small (polynomial size) grid, because then high-precision operations would be unnecessary. The question whether such embeddings are possible or not was left open in [RT86].

This problem was solved by de Fraysseix, Pach and Pollack [FPP88, FPP90] who proved that, for $n \geq 3$, each n-vertex planar graph can be drawn on the $(2n-4) \times (n-2)$ grid (which has $(2n-3)(n-1)$ points). They also presented an $O(n \log n)$-time algorithm for constructing such embeddings, but left open the problem of whether it is possible to find such an embedding in linear time.

We answer this question in the affirmative by presenting a linear-time implementation of the technique from [FPP88, FPP90]. Although we base our algorithm on the general method from their proof, the algorithm itself differs significantly from theirs. Also, unlike their algorithm, our method does not require any sophisticated data structures. Instead, it distributes and carefully manages the information needed in the algorithm, so that we always have sufficient local information to find a tentative embedding of each new vertex — later it is moved to its correct final position using the information stored in a tree-like structure. The algorithm is very easy to implement; in fact, we give a short Pascal-like code segment for the main part of the algorithm.

The use of integer coordinates has another advantage (besides speed and accuracy): it guarantees, automatically, that the resulting picture has fairly good proportions. Jones *et al* [Jon93] show that, indeed, the algorithm presented in this paper compares favorably, in terms of aesthetics of drawings, with other graph drawing algorithms.

This algorithm was developed in 1989 [CP89], but not published. Progress on this problem has been made since that time. In particular, Schnyder [Sc90] showed a different technique that gives a linear-time embedding algorithm into a smaller $(n-2) \times (n-2)$ grid. It turned out, however, that his result did not eliminate the interest in our algorithm, possibly because our approach is easier to implement, and the resulting embeddings tend to be more aesthetic.

2 Outline of the Algorithm

In this section we sketch the proof of de Fraysseix-Pach-Pollack theorem, on which our algorithm is based. A linear-time implementation will be presented in the next section.

Since a planar graph can be triangulated in linear time (see, for example, [Ka93, Section 6.1]), we consider only maximal (triangulated) planar graphs. From now on, let G be a fixed, but arbitrary, n-vertex triangulated planar graph. We use the following lemma, proved by de Fraysseix, Pach and Pollack in [FPP88, FPP90].

Lemma 1 *It is possible to order the vertices of G in a sequence $v_1, \ldots, v_n$ such that for $k = 3, 4, \ldots, n-1$:*

(a) the subgraph G_k of G induced by $v_1, \ldots, v_k$ is 2-connected, internally triangulated, and the boundary of its exterior face is a cycle C_k containing the edge (v_1, v_2);

(b) v_{k+1} is in the exterior face of G_{k+1}, it has at least two neighbors in G_k, and these neighbors are consecutive on the path $C_k - (v_1, v_2)$.

Proof: We present only a sketch. Let $C_n = (v_1, v_2, v_n)$ be the outer face of G. Inductively, suppose that $v_n, v_{n-1}, \ldots, v_{k+1}$ have been defined, and that graphs $G_n, G_{n-1}, \ldots, G_{k+1}$ satisfy the lemma. Then there is a vertex v in G_{k+1}, $v \notin \{v_1, v_2\}$, which has exactly two neighbors in C_{k+1} (so it is not incident to any chords of C_{k+1}). We take $v_k = v$. $\square$

The ordering from Lemma 1 is called a *canonical ordering* of G. Our algorithm embeds G one vertex at a time in cannonical order, at each stage adjusting the current partial embedding. With each vertex v_k, we associate a set $L(v_k)$ of vertices that move with v_k whenever its position is adjusted.

By $P(v)$ we will denote the current position of v in the grid; for the sake of notation $P(v)$ is denoted $(x(v), y(v))$. For two grid points $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ whose Manhattan distance is even, we denote by $\mu(P_1, P_2)$ the grid point in the intersection of the line with slope $+1$ through P_1 and the line with slope -1 through P_2, that is

$$\mu(P_1, P_2) \;=\; \tfrac{1}{2}(x_1 - y_1 + x_2 + y_2, -x_1 + y_1 + x_2 + y_2).$$

Now we describe the embedding strategy. First, we set $L(v_i) := \{v_i\}$ for $i = 1, 2, 3$, and $P(v_1) := (0, 0)$; $P(v_2) := (2, 0)$; $P(v_3) := (1, 1)$.

Suppose that by stage k we have already embedded G_{k-1} (so the initial step is actually regarded as step 3) in such a way that the following conditions hold:

(e1) $P(v_1) = (0, 0)$ and $P(v_2) = (2k - 4, 0)$.

(e2) $x(w_1) < x(w_2) < \ldots < x(w_m)$, where $v_1 = w_1, w_2, \ldots, w_m = v_2$ is C_{k-1} in clockwise order.

(e3) All segments $(P(w_i), P(w_{i+1}))$, $i = 1, \ldots, m - 1$, have slopes either $+1$ or -1.

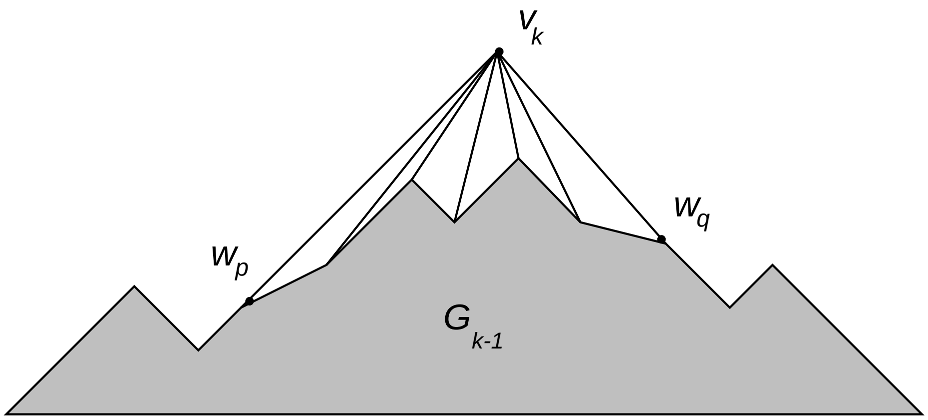

Figure 1: Embedding v_k.

From now on, we will refer to the sequence $w_1, \ldots, w_m$ (or, sometimes rather informally, to their embedding) as the *contour* of G_{k-1}. Let $w_p, \ldots, w_q$ be the neighbors of v_k in C_{k-1}. We will say that the vertex v_k *covers* the vertices $w_{p+1}, \ldots, w_{q-1}$ (See Figure 1). Note that by property (e3), the Manhattan distance between any two w_i's is even, so $\mu(w_p, w_q)$ is always a grid point. We now shift some portions of the embedding to the right, by one or two, and install v_k as follows:

Step 1: **for each** $v \in \bigcup_{i=q}^{m} L(w_i)$ **do** $x(v) := x(v) + 2$;

Step 2: **for each** $v \in \bigcup_{i=p+1}^{q-1} L(w_i)$ **do** $x(v) := x(v) + 1$;

Step 3: $P(v_k) := \mu(P(w_p), P(w_q))$;

Step 4: $L(v_k) := \{v_k\} \cup \bigcup_{i=p+1}^{q-1} L(w_i)$.

By moving some of the points $P(w_i)$ in Steps 1 and 2, we ensure that all neighbors of v_k will be visible from $P(v_k)$. Clearly, the conditions (e1), (e2), and (e3) are satisfied. We have to show that G_k is straight-line embedded. This follows from the following lemma, proved by de Fraysseix, Pach and Pollack in [FPP88, FPP90].

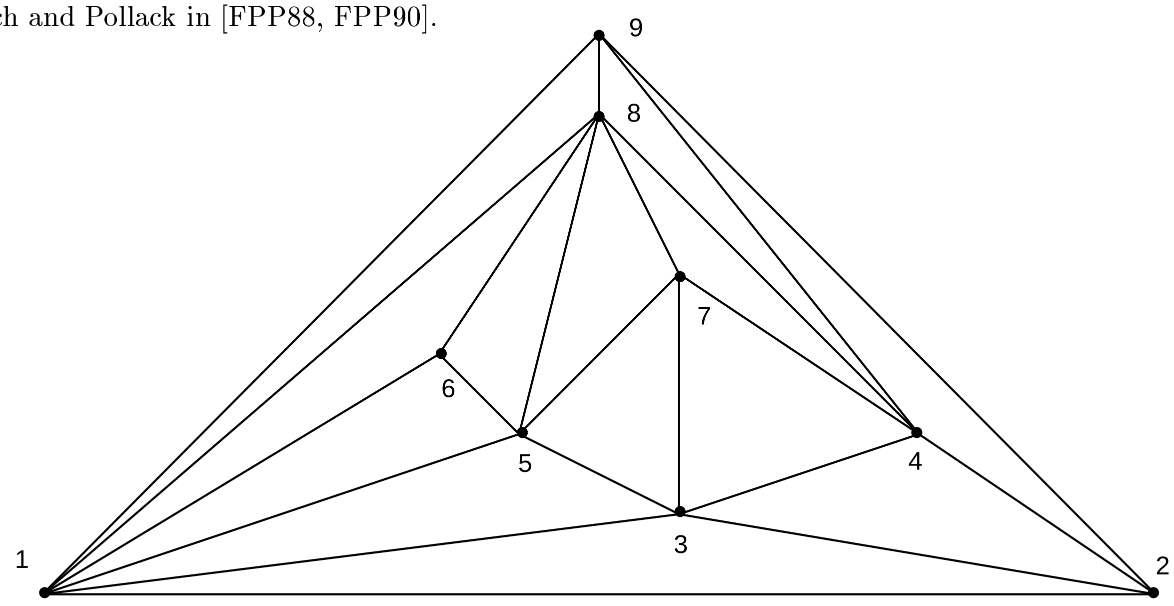

Figure 2: An example of a canonical ordering, sets $L(w_i)$, and a partial embedding.

Lemma 2 *Let G_k be straight-line embedded, as described above. Suppose we are given a non-decreasing sequence of non-negative integers $\rho_1 \leq \rho_2 \leq \ldots \leq \rho_m$. If, for each i, we translate the points in $L(w_i)$ by ρ_i to the right, we again obtain a straight-line embedding.*

Proof: The proof is by induction on k. For G_3 the lemma is obvious. So suppose that it holds for G_{k-1}. As in the algorithm, let the contour of G_{k-1} be $w_1, \ldots, w_m$. We are about to add v_k. Let

$$\rho_1 \leq \ldots \leq \rho_p \leq \rho \leq \rho_q \leq \ldots \leq \rho_m$$

be a fixed sequence of non-negative integers, and translate each $L(w_i)$ by ρ_i and $L(v_k)$ by ρ. We show that G_k remains straight-line embedded.

Take the sequence $\rho'_1 \le \rho'_2 \le \ldots \le \rho'_m$ defined by

$$\rho'_i = \begin{cases} \rho_i & \text{for } i = 1, \ldots, p \\ \rho + 1 & \text{for } i = p+1, \ldots, q-1 \\ \rho_i + 2 & \text{for } i = q, \ldots, m. \end{cases}$$

By induction, when we translate the sets $L(w_i)$ of G_{k-1} by the ρ'_i, G_{k-1} remains straight-line embedded. Thus G_k is straight-line embedded, because v_k moves rigidly with $w_p, \ldots, w_q$. $\square$

So, in the end we have a straight-line embedding of G such that $P(v_1) = (0,0)$ and $P(v_2) = (2n-4, 0)$. By (e3), $P(v_n) = (n-2, n-2)$. Therefore, the whole graph is embedded in the $(2n-4) \times (n-2)$ grid.

3 Linear-time Algorithm

It is easy to implement the method of the preceding section in time $O(n^2)$. In [FPP88, FPP90] this time bound was reduced to $O(n \log n)$ using data structures for rectangle-range queries. Unlike that algorithm, ours follows the method of the proof from Section 2, and the linear-time complexity is achieved by appropriately distributing information in the vertices of the graph.

We assume that G is already triangulated and embedded in the plane, and that a canonical ordering $v_1, \ldots, v_n$ of G is given. A planar embedding of G can be found in linear time using standard algorithms, for example [HT74] or [BL76]. It is quite easy to triangulate a given planar graph in linear-time, given its planar embedding (see, for example, [Ka93, Section 6.1]). Also, the existence proof for canonical orderings presented in the previous section can be implemented in linear time by maintaining a queue of vertices that have degree 2 in C_k (see also [FPP88, FPP90]).

We view G_k as a forest consisting of trees $L(w_1), \ldots, L(w_m)$ rooted at the members of $C_k = \{w_1, \ldots, w_m\}$. The children of a node are the vertices that it covers, its neighbors that leave the contour when it is installed. We follow the common practice of representing this forest as a binary tree T, where the left T-child of a node is its leftmost child (if any), and the right T-child of a node is its next sibling to the right (if any).

The root of T is v_1, and $C_k = \{w_1, \ldots, w_m\}$ consists of: v_1, its right T-child, its right T-child's right T-child, etc. $L(w_i)$ consists of w_i and its left T-subtree. Thus, the T-subtree rooted at w_i consists of $\bigcup_{j \ge i} L(w_j)$.

The crucial observation is that, when we embed v_k, it is not really necessary to know the exact positions of w_p and w_q. If we know only their y-coordinates and their relative x-cooridnates (that is, $x(w_q) - x(w_p)$), then we can compute $y(v_k)$, and the x-coordinate of v_k relative to w_p, that is $x(v_k) - x(w_p)$.

For each vertex $v \neq v_1$, the *x-offset* of v is defined as $\Delta x(v) = x(v) - x(w)$, where w is the T-parent of v. More generally, if w is an ancestor of v, the x-offset between w and v is $\Delta x(w,v) = x(v) - x(w)$.

With each vertex v we store the following information:

$$\begin{aligned} \mathrm{Left}(v) &= \text{ the left } T\text{-child of } v, \\ \mathrm{Right}(v) &= \text{ the right } T\text{-child of } v, \\ \Delta x(v) &= \text{ the } x\text{-offset of } v \text{ from its } T\text{-parent}, \\ y(v) &= \text{ the } y\text{-coordinate of } v. \end{aligned}$$

The algorithm consists of two phases. In the first phase, we add new vertices one by one, and each time we add a vertex we compute its x-offset and y-coordinate, and update the x-offsets of one or two other vertices. In the second phase, we traverse the tree and compute final x-coordinates by accumulating offsets.

The first phase is implemented as follows: First we initialize the values stored at v_1, v_2, v_3:

Initialize: $\Delta x(v_1), y(v_1), \mathrm{Right}(v_1), \mathrm{Left}(v_1) := 0, 0, v_3, \mathrm{nil};$
$\Delta x(v_2), y(v_2), \mathrm{Right}(v_2), \mathrm{Left}(v_2) := 1, 0, \mathrm{nil}, \mathrm{nil};$
$\Delta x(v_3), y(v_3), \mathrm{Right}(v_3), \mathrm{Left}(v_3) := 1, 1, v_2, \mathrm{nil};$

Then, we embed other vertices, one by one:

for $k := 4$ to n **do begin**
Notation: let $w_1, \ldots, w_m$ be the contour of G_{k-1};
let $w_p, \ldots, w_q$ be the neighbours of v_k in G_{k-1};
Stretch gaps: $\Delta x(w_{p+1}) := \Delta x(w_{p+1}) + 1;$
$\Delta x(w_q) := \Delta x(w_q) + 1;$
Adjust offsets: $\Delta x(w_p, w_q) := \Delta x(w_{p+1}) + \ldots + \Delta x(w_q);$
$\Delta x(v_k) := \frac{1}{2}\left[-y(w_p) + \Delta x(w_p, w_q) + y(w_q)\right];$
$y(v_k) := \frac{1}{2}\left[y(w_p) + \Delta x(w_p, w_q) + y(w_q)\right];$
$\Delta x(w_q) := \Delta x(w_p, w_q) - \Delta x(v_k);$
if $p + 1 \neq q$ **then** $\Delta x(w_{p+1}) := \Delta x(w_{p+1}) - \Delta x(v_k);$
Install v_k*:* $\mathrm{Right}(w_p) := v_k;$
$\mathrm{Right}(v_k) := w_q;$
if $p + 1 \neq q$ **then begin**
  $\mathrm{Left}(v_k) := w_{p+1};$
  $\mathrm{Right}(w_{q-1}) := \mathrm{nil}$
end
  else $\mathrm{Left}(v_k) := \mathrm{nil};$
end;

In the second phase, we invoke AccumulateOffsets($v_1, 0$); where the procedure Accumulate-Offsets is defined as follows:

```
procedure AccumulateOffsets(v: vertex, δ: integer);
begin
    if v ≠ nil then begin
        Δx(v) := Δx(v) + δ;
        AccumulateOffset(Left(v), Δx(v));
        AccumulateOffset(Right(v), Δx(v))
    end
end;
```

In order to prove correctness, it is sufficient to show that the offsets are computed correctly.

To see that the *stretch* step works correctly, recall that the T-subtree rooted at w_i consists of $\bigcup_{j \geq i} L(w_j)$. So, incrementing the offset of w_i increments the cumulative offset from v_1 of each member of that T-subtree, i.e., shifts them all to the right.

During the *adjustment* step of v_k, only the offsets of w_q and possibly w_{p+1} get changed. But w_p remains an ancestor of each, and their offsets from w_p remain unchanged, by simple algebra. It follows that the cumulative offsets of all vertices already in the graph remain unchanged by the adjustment step.

As for complexity, we have already mentioned that the triangulation and the cannonical ordering can be found in time $O(n)$. In first phase, when we add v_k, the cost is at most $O(\deg(v_k))$. So the time complexity of the first phase is $O(n)$. Obviously, the second phase runs in linear time.

4 Final Comments

As mentioned in the introduction, more progress has been made recently. Schnyder [Sc90] has developed another technique that yields a linear-time embedding algorithm into an $(n-2)\times(n-2)$ grid. He also pointed out (private communication) that our method can be improved to use a grid of this size: when we install v_k, we move all vertices $w_{p+1}, \ldots, w_m$ by one to the right, and then place v_k at $P(v_k) = (x, y)$, where $x = x(w_p) + 1$ and $y = y(w_q) + x(w_q) - x$, that is, the edge (v_k, w_q) will have slope -1. The invariant will be different: all slopes must belong to the set $[0, \infty) \cup \{-1\}$. The proof of correctness and the linear-time implementation are similar. The grid size is smaller, but the embedding occupies the triangle $(0,0) - (n-2, 0) - (1, n-2)$ (similarly to the one from [Sc90]), and the overall drawing is asymetric and may not be very aesthetic. Depending on the application, either of these two methods may be preferable.

The problem of convex drawing of planar graphs has recently attracted attention, see [CYN84, CON85, Ka92]. This is due to the fact that the method from [FPP88, FPP90], as well as Schnyder's algorithm [Sc90], triangulate the given graph before emedding it and remove the added edges afterwards — this may produce unaesthetic drawings when the initial graph is sparse. Goos Kant [Ka92] gave a linear algorithm, based on our technique, that produces convex drawings of planar graphs in a $(2n-4) \times (n-2)$ grid. Recently, Chrobak and Kant [CK93] observed that, using Schnyder's improvement of our algorithm, it can be modified to produce convex embeddings in a $(n-2) \times (n-2)$ grid. A similar result has been obtained independently by Schnyder and Trotter [ST92], using the technique from [Sc90].

References

[BL76] K.S. Booth, G.S. Lueker, *Testing for consecutive ones property, interval graphs and graph planarity testing using PQ-tree algorithms*, Journal of Computer and System Sciences 13 (1976) 335-379.

[CP89] M. Chrobak, T. Payne, *A linear-time algorithm for drawing planar graphs on a grid*, Technical Report UCR-CS-89-1, Department of Mathematics and Computer Science, University of California at Riverside, 1989.

[CK93] M. Chrobak, G. Kant, *Convex grid drawings of 3-connected planar graphs*, Technical Report RUU-CS-93-45, Department of Computer Science, Utrecht University, 1993.

[CYN84] N. Chiba, T. Yamanouchi, 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.

[CON85] N. Chiba, K. Onoguchi, and T. Nishizeki, *Drawing planar graphs nicely*, Acta Inform. 22 (1985) 187-201.

[Fa48] I. Fáry, *On straight lines representation of planar graphs*, Acta. Sci. Math. Szeged 11 (1948) 229-233.

[FPP88] H. de Fraysseix, J. Pach, R. Pollack, *Small sets supporting Straight-Line Embeddings of planar graphs*, Proc. 20th Annual Symposium on Theory of Computing, 1988, pp. 426-433.

[FPP90] H. de Fraysseix, J. Pach, R. Pollack, *How to draw a planar graph on a grid*, Combinatorica 10 (1990) 41-51.

[HT74] J. Hopcroft, R.E. Tarjan, *Efficient planarity testing*, Journal of ACM 21 (1974) 549-568.

[Jon93] S. Jones, P. Eades, A. Moran, N. Ward, G. Delott, R. Tamassia, *A note on planar graph drawing algorithms*, Technical Report 216, Dept. of Computer Science, Univ. of Queensland, 1991.

[Ka92] G. Kant, *Drawing planar graphs using the lmc-ordering*, Proc. 33rd Symp. on Foundations of Computer Science, Pittsburgh, 1992, pp. 101-110.

[Ka93] G. Kant, *Algorithms for Drawing Planar Graphs*, Ph.D. Dissertation, Department of Computer Science, University of Utrecht, 1993.

[RT86] P. Rosenstiehl, R.E. Tarjan, *Rectilinear planar layouts and bipolar orientations of planar graphs*, Discrete Computational Geometry 1 (1986) 343-353.

[Sc90] W. Schnyder, *Embedding planar graphs in the grid*, Proc. 1st Annual ACM-SIAM Symp. on Discrete Algorithms, San Francisco, 1990, pp. 138-147.

[ST92] W. Schnyder, W. Trotter, *Convex drawings of planar graphs*, Abstracts of the AMS, 13, 5 (1992), 92T-05-135.

[St51] S. K. Stein, *Convex maps*, Proc. Amer. Math. Soc., 2 (1951) 464-466.

[SR34] E. Steinitz and H. Rademacher, Vorlesungen über die Theorie Der Polyeder, Julius Springer, Berlin, Germany, 1934.

[TT86] R. Tamassia, I.G. Tollis, *A unified approach to visibility representations of planar graphs*, Discrete Computational Geometry 1 (1986) 321-341.

[Tu63] W.T. Tutte, *How to draw a graph*, Proc. London Math. Soc. 13 (1963) 743-768.

[Wa36] K. Wagner, *Bemerkungen zum vierfarbenproblem*, Jahresbericht der Deutschen Mathematiker-Vereinigung, 46 (1936) 26-32.