6

Planar Straight-Line Drawing Algorithms

Luca Vismara

6.1 Introduction

Planar straight-line drawings have been an early subject of investigation in combinatorial mathematics. A classic result states that every planar graph admits a planar straight-line drawing. Namely, if a graph can be drawn with no crossings using edges of arbitrary shape (e.g., polygonal lines or curves), then it can be drawn with no crossings using only straight-line edges (see Figure 6.1). The proof of this result was independently discovered by Steinitz and Rademacher [SR34], Wagner [Wag36], Fary [Fár48], and Stein [Ste51].

All the above classic constructions focus on establishing the existence of planar straight-line drawings but do not address the area of the drawing or the arithmetic precision required for representing the coordinates of the vertices. Indeed, following the constructions in these papers one obtains drawings of area exponential in the length of the shortest edge, which are unsuitable in practice.

Algorithms for constructing planar straight-line grid drawings, where the edges have integer coordinates, were developed by de Fraysseix, Pach, and Pollack [dFPP90] (shift method) and by Schnyder [Sch90] (realizer method). They independently showed that every n-vertex planar graph has a planar straight-line grid drawing with $O(n)$ height and $O(n)$ width, resulting in $O(n^2)$ area. These bounds are asymptotically tight in the worst case as can be shown with the example of Figure 6.2.

Convex drawings are planar straight-line drawings where all the faces are drawn as convex polygons (see Figure 6.1(c)). We say that a planar graph is convex planar if it admits a convex drawing. In another classic work, Tutte [Tut60, Tut63] showed how to construct a

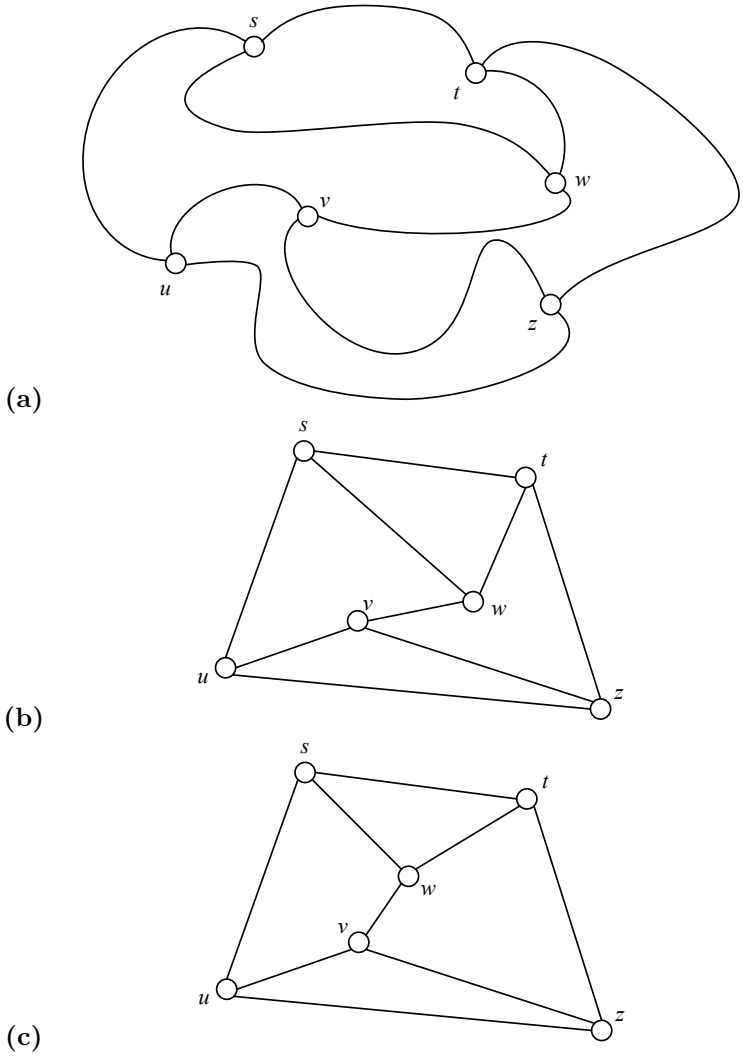


Figure 6.1 Examples of planar drawings of the same graph: (a) planar drawing with curved edges; (b) planar straight-line drawing; (c) planar convex drawing.

convex planar drawing of every triconnected planar graph. His method places the vertices of the external face on an arbitrary convex polygon and computes the coordinates of the remaining vertices by solving a system of linear equations.

The rest of this chapter is organized as follows. Basic definitions are introduced in Section 6.2. Tutte's classic algebraic method for convex drawings is presented in Section 6.3. Area bounds for planar straight-line grid drawings computed by the shift method and by the realizer method are summarized in Section 6.4. Canonical orderings of planar graphs are discussed in Section 6.5. Section 6.6 describes the shift method and Section 6.7 describes the realizer method.

For further details on the subject of planar drawings of graphs, we refer the reader to the book by Nishizeki and Rahman [NR04] and the survey by Di Battista and Frati [DF13]. See also the work by Cruz and Garg [CG95] for a declarative approach to the construction of planar drawings.

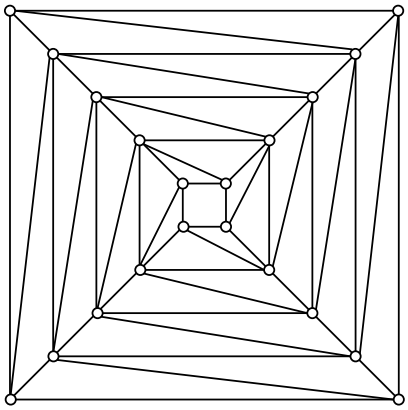


Figure 6.2 Planar straight-line grid drawing of graph S_5 consisting of five nested cycles of four vertices. This drawing has height 9 and width 9. In general, graph S_k has $4k$ vertices and requires height and width proportional to k in any planar-straight-line grid drawing.

6.2 Preliminaries

6.2.1 Planar Drawings

In the context of this chapter, a *drawing* of a graph G is a mapping of each vertex v of G to a distinct point $P(v) = (v_x, v_y)$ of the plane[1] and of each edge (u, v) of G to a simple Jordan curve with endpoints $P(u)$ and $P(v)$. A *straight-line drawing* is a drawing in which every edge is mapped to a straight-line segment; more formally, a straight-line drawing is an injective function $f : v \in V \to (v_x, v_y) \in \mathbb{R}^2$.

A drawing is *planar* if no two edges intersect, except, possibly, at common endpoints. A graph is planar if it has a planar drawing. Two planar drawings of a planar graph G are *equivalent* if, for each vertex v, they have the same circular clockwise sequence of edges incident with v. Hence, the planar drawings of G are partitioned into equivalence classes. Each of those classes is called an *embedding* of G. An *embedded* planar graph (also *plane* graph) is a planar graph with a prescribed embedding. A triconnected planar graph has a unique embedding, up to a reflection. A planar drawing divides the plane into topologically connected regions delimited by cycles; these cycles are called *faces*. The *external* face is the cycle delimiting the unbounded region; all the other faces are *internal*. Two equivalent planar drawings have the same faces. Hence, one can refer to the faces of an embedding. A vertex or edge of a plane graph is said to be *external* if it belongs to the external face, and *internal* otherwise.

A *maximal planar graph* is a planar graph with the maximal number of edges, i.e., adding an edge between any two vertices destroys its planarity. Note that in a maximal planar graph all faces consist of three edges. An *outerplanar graph* is a planar graph that admits a planar drawing with all its vertices on the same (say, the external) face; such a drawing is called an outerplanar drawing.

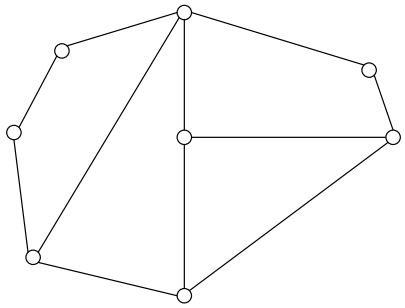
Let G be a plane graph; the *dual graph* G^* of G is defined as follows: (i) each face f of G has a *dual vertex* f^* in G^*; (ii) each vertex v of G has a *dual face* v^* in G^*; (iii) let e be

[1]We will use interchangeably (v_x, v_y) and $(x(v), y(v))$ to denote the coordinates of $P(v)$.

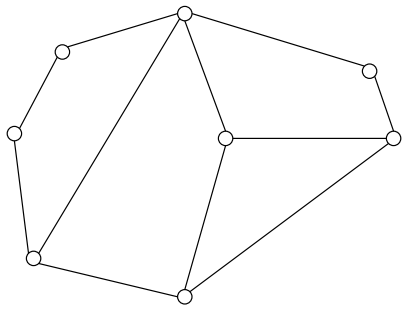
an edge of G and let f_1 and f_2 be the two faces of G incident with e (note that f_1 and f_2 may not be distinct); e has a *dual edge* $e^* = (f_1^*, f_2^*)$ in G^*.

6.2.2 Convex Drawings

A *polygon* is a finite set of segments such that every segment endpoint is shared by exactly two segments and no subset of segments has the same property. A polygon is *simple* if there is no pair of nonconsecutive segments sharing a point. A simple polygon is *convex* if its interior is a convex set. A simple polygon is *strictly convex* if its interior is a strictly convex set, i.e., no 180° angle is allowed. A *convex drawing* of a planar graph G is a planar straight-line drawing of G in which all faces are drawn as convex polygons (see Figure 6.3(a)). A *strictly convex drawing* of a planar graph G is a planar straight-line drawing of G in which all faces are drawn as strictly convex polygons (see Figure 6.3(b)). A planar graph is said to be *(strictly) convex planar* if it admits a (strictly) convex drawing.



(a)



(b)

Figure 6.3 (a) A convex drawing of a biconnected planar graph G. (b) A strictly convex drawing of a biconnected planar graph G.

6.2.3 Connectivity

We recall some basic definitions on connectivity. A *separating k-set* of a graph is a set of k vertices whose removal disconnects the graph; separating 1-sets and 2-sets are called *cutvertices* and *separation pairs*, respectively. A graph is *k-connected* if it contains more than k vertices and no separating $(k-1)$-set; 1-connected, 2-connected, and 3-connected

graphs are called *connected*, *biconnected*, and *triconnected*, respectively. A *separating edge* of a graph is an edge whose removal disconnects the graph.

The *biconnected components* of a connected graph (also called *blocks*) are its maximal biconnected subgraphs and its separating edges. The *triconnected components* of a biconnected graph G are defined as follows [HT73].

If G is triconnected, then G itself is the unique triconnected component of G. Otherwise, let $\{u, v\}$ be a separation pair of G. We partition the edges of G into two disjoint subsets E_1 and $E_2, |E_1| \geq 2, |E_2| \geq 2$, such that the subgraphs G_1 and G_2 induced by them have only vertices u and v in common. Graphs $G'_1 = G_1 + (u, v)$ and $G'_2 = G_2 + (u, v)$ are called the *split graphs* of G with respect to $\{u, v\}$ (multiple edges are allowed); edge (u, v) in G'_1 and G'_2 is called a *virtual edge*. Dividing G into split graphs G'_1 and G'_2 is called *splitting*. Reassembling split graphs G'_1 and G'_2 into G, is called *merging*. Note that only split graphs that resulted from the same splitting operation can be merged together. We continue the splitting process recursively on G'_1 and G'_2 until no further splitting is possible. Each resulting graph is either a triconnected simple graph, or a set of three multiple edges (called "*triple bond*" in [HT73]), or a cycle of length three (called "*triangle*" in [HT73]). The *triconnected components* of G are obtained from these graphs by merging the "triple bonds" into maximal sets of multiple edges (called "*bonds*" in [HT73]), and the "triangles" into maximal simple cycles (called "*polygons*" in [HT73]). When merging "triple bonds" into "bonds" and "triangles" into "polygons," virtual edges with both endvertices in common are removed; we refer to the remaining virtual edges at the end of the merging process as the *virtual edges of the triconnected components*. Note that, although the graphs obtained at the end of the splitting process depend on the order of the splittings, the triconnected components of G are unique. See [HT73] for further details.

In the rest of the chapter, we denote by n, m, and l the number of vertices, edges, and faces of a plane graph, respectively; we always assume $n \geq 3$. Unless otherwise specified, graphs are assumed to be simple, i.e., without self-loops and multiple edges. Often, we do not distinguish between a vertex (edge) of G and the point (segment) representing it.

We recall *Euler's formula*, which holds for every plane graph, and two bounds for the number of edges and faces of a plane graph (the equalities hold for maximal planar graphs), which easily follow from it:

$$n + l = m + 2 \tag{6.1}$$

$$m \leq 3n - 6 \tag{6.2}$$

$$l \leq 2n - 4 \tag{6.3}$$

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two points on the plane; the *Manhattan distance* between P_1 and P_2 is defined as $|x_1 - x_2| + |y_1 - y_2|$.

A $w \times h$ *integer grid* is a grid of integer points of width w and height h; note that a $w \times h$ integer grid contains $(w+1) \times (h+1)$ integer points. A *grid drawing* is an injective function $f : v \in V \rightarrow (v_x, v_y) \in \mathbb{Z}^2$. The *area* of a grid drawing is the number of integer points contained in the smallest integer grid containing the drawing. In the rest of the chapter, we will often omit "integer" before "grid" for brevity.

6.3 Real-Coordinate Drawings

In a classic paper, Tutte [Tut60, Tut63] presented a method for constructing strictly convex drawings of triconnected plane graphs by solving a system of linear equations that place each internal vertex at the barycenter of its neighbors. Hence, this method is referred to as the *barycenter method*.

Initially, the vertices of the external face are placed at the vertices of a strictly convex polygon, P. We refer to the vertices not on the external face as internal vertices.

For a vertex v, let $N(v)$ be the set of neighbors of v and $d(v)$ the degree of v, i.e., $d(v) = |N(v)|$. The position of an internal vertex v is determined by the following linear equations:

$$x(v) = \frac{1}{d(v)} \sum_{w \in N(v)} x(w) \tag{6.4}$$

$$y(v) = \frac{1}{d(v)} \sum_{w \in N(v)} y(w) \tag{6.5}$$

Tutte showed that the above system of linear equations admits a unique solution that corresponds to a strict convex drawing of the graph. An example of a drawing constructed with the barycenter method is shown in Figure 6.4.

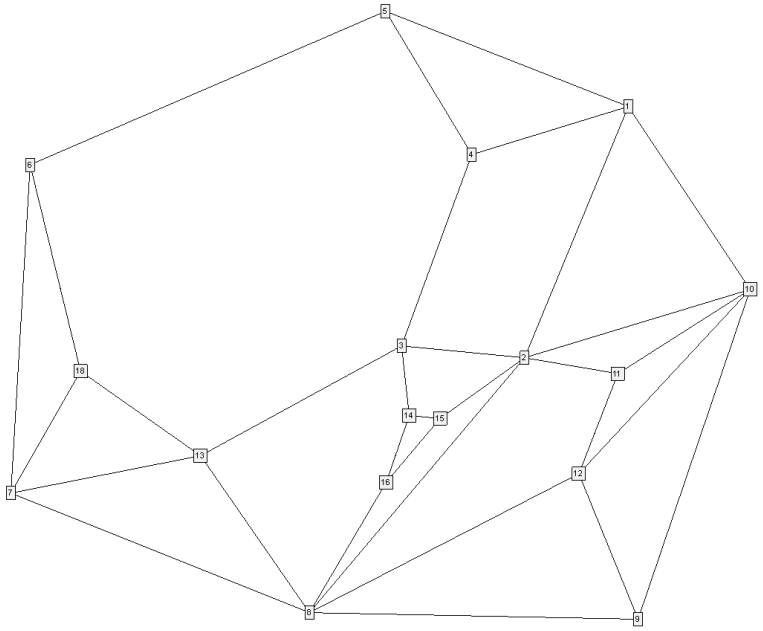


Figure 6.4 Planar convex drawing obtained with Tutte's barycenter method. Drawing created by the PIGALE tool (see Chapter 18).

Combinatorial characterizations of convex and strictly convex planar graphs and methods for constructing convex and strictly convex drawings appear in papers by Tutte [Tut60, Tut63], Thomassen [Tho80, Tho84], Chiba, Yamanouchi, and Nishizeki [CYN84], Chiba, Onoguchi, and Nishizeki [CON85], and Djidjev [Dji95]. Note that the above methods compute drawings with real coordinates for the vertices.

6.4 Grid Drawings

The drawings generated by Tutte's algorithm presented in Section 6.3 exhibit some drawbacks:

- they require high-precision real arithmetic relative to the size of the input graph, and therefore cannot be used even for graphs of moderate size; and
- in the produced drawings, the ratio of the largest distance to the smallest distance between vertices is very large (exponential in the size of the graph), i.e., vertices are represented by arbitrarily close points, or, equivalently, if the graph is drawn on an integer grid, then the grid has exponential size.

Motivated by these drawbacks, Rosenstiehl and Tarjan [RT86] posed the question whether every planar graph has a planar straight-line drawing on an $O(n^k) \times O(n^k)$ integer grid for some fixed constant k, where n is the number of vertices of the graph. As we will see, the question was answered in the positive and various algorithms were presented over the years. Selected algorithms are summarized in Table 6.1.

[CP95, dFPP90]	$(2n-4) \times (n-2)$	shift
[CN98]	$\lfloor \frac{2}{3}(n-1) \rfloor \times 4 \lfloor \frac{2}{3}(n-1) \rfloor - 1$	shift
[Bra08]	$\lceil \frac{4}{3}n \rceil \times \lceil \frac{2}{3}n \rceil$	shift
[Sch90]	$(2n-5) \times (2n-5)$	realizer
	$(n-2) \times (n-2)$	

Table 6.1 Width and height of the drawing achieved by selected planar straight-line grid drawing algorithms that use the shift method or the realizer method. We denote with n the number of vertices of the graph.

The algorithms listed in Table 6.1 are designed for drawing maximal plane graphs but can actually be used to draw general plane graphs: it is sufficient to transform the input plane graph into a maximal plane graph by adding a linear number of extra edges, draw the resulting graph, and then remove the segments corresponding to the extra edges from the obtained drawing. These algorithms are based on two different methods, called the *shift method* and the *realizer method*, and are described in Sections 6.6 and 6.7, respectively.

6.5 Canonical Orderings

In this section, we recall the definitions of canonical ordering of maximal plane graphs, as given by de Fraysseix, Pach, and Pollack [dFPP90], and of triconnected plane graphs, as given by Kant [Kan96].

DEFINITION 6.1 Let G be a maximal plane graph with n vertices, and let u_0, u_1, u_2 be the external vertices of G in counterclockwise order. A *canonical ordering* of G (see Figure 6.5) is an ordering $v_1, \ldots, v_n$ of the vertices of G such that the following conditions are verified:

1. $v_1 = u_1$, $v_2 = u_2$.
2. For $3 \leq k \leq n$, let G_k be the plane subgraph of G induced by vertices $v_1, \ldots, v_k$ and let C_k be the external face of G_k. Vertex v_k is on face C_k. Also, if $k < n$, vertex v_k has at least one neighbor in $G - G_k$.

3. For each $3 \leq k \leq n-1$, subgraph G_k is biconnected and internally maximal (i.e., all internal faces of G_k are triangles).
4. $v_n = u_0$.

LEMMA 6.1 **[dFPP90]** Each maximal plane graph has a canonical ordering, which can be computed in linear time and space.

A canonical ordering of G yields an incremental construction of graph G starting from edge (v_1, v_2). In step k ($3 \leq k \leq n$), vertex v_k and the edges between v_k and its neighbors in C_{k-1} are added to the current graph G_{k-1}. For each $3 \leq k \leq n$, we denote by $v_1 = w_1, w_2, \ldots, w_t = v_2$ the sequence of vertices of C_{k-1}, when traversed in clockwise order. For the sake of enhancing intuition, we visualize $w_2, \ldots, w_{t-1}$ as arranged from left to right above (v_1, v_2) in the plane. For each $3 \leq k \leq n$, let $w_p, \ldots, w_q$ be the subsequence of vertices of C_{k-1} that are adjacent to v_k (note that $p+1$ may be equal to q). After v_k has been added to G_{k-1}, vertices $w_{p+1}, \ldots, w_{q-1}$ (if any) are no longer external; we say that vertex v_k *covers* these vertices.

A canonical ordering $v_1, \ldots, v_n$ of graph G defines a spanning tree of graph $G - \{v_1, v_2\}$, called *cover tree*, which consists of all edges (u, v) such that u covers v. We set v_n as the root of the cover tree. Thus, the children of a vertex u in the cover tree are the vertices covered by u. (See Figure 6.6.) We define the *cover forest* associate with a canonical ordering as its cover tree together with the single-vertex trees v_1 and v_2.

The definition of canonical ordering can be generalized to triconnected plane graphs as follows. A biconnected plane graph G is said to be *internally triconnected* if for any separation pair $\{u, v\}$ of G, u and v are external vertices and each connected component of $G \setminus \{u, v\}$ contains an external vertex; in other words, G is internally triconnected if and only if the graph obtained from G by adding a new vertex and connecting it to all the external vertices of G is triconnected.

DEFINITION 6.2 Let G be a triconnected plane graph with n vertices, (u_1, u_2) be an external edge of G, and $u_0 \neq u_1, u_2$ be an external vertex of G. A *canonical ordering* of G is an ordering $v_1, \ldots, v_n$ of the vertices of G that can be partitioned into subsequences $V_1, \ldots, V_h$, where $V_k = \{v_{s_k}, \ldots, v_{s_k+d_k}\}$, $1 \leq k \leq h$, $1 = s_1 < s_2 < \cdots < s_h < s_{h+1} = n+1$, $d_k = s_{k+1} - s_k - 1$, such that the following conditions are verified:

1. $v_1 = u_1$, $v_2 = u_2$, and $V_1 = \{v_1, v_2\}$.
2. Let G_k be the plane subgraph of G induced by $V_1 \cup \cdots \cup V_k$, $1 \leq k \leq h$, and C_k be the external face of G_k. For each $2 \leq k \leq h-1$, one of the following cases occurs:
   (a) $V_k = \{v_{s_k}\}$ is a vertex of C_k (and has at least one neighbor in $G - G_k$);
   (b) $V_k = \{v_{s_k}, \ldots, v_{s_k+d_k}\}$ is a subpath of C_k, and each vertex v_i, $s_k \leq i \leq s_k + d_k$, has degree two in G_k (and has at least one neighbor in $G - G_k$).
3. Each subgraph G_k, $2 \leq k \leq h-1$, is biconnected and internally triconnected.
4. $v_n = u_0$ and $V_h = \{v_n\}$.

LEMMA 6.2 **[Kan96]** Each triconnected plane graph has a canonical ordering, which can be computed in linear time and space.

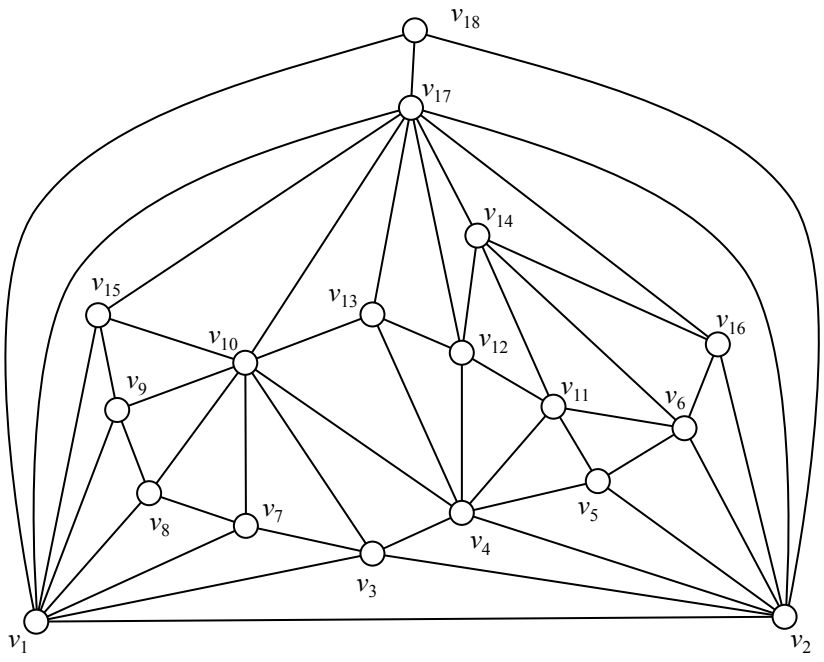


Figure 6.5 A canonical ordering of a maximal plane graph.

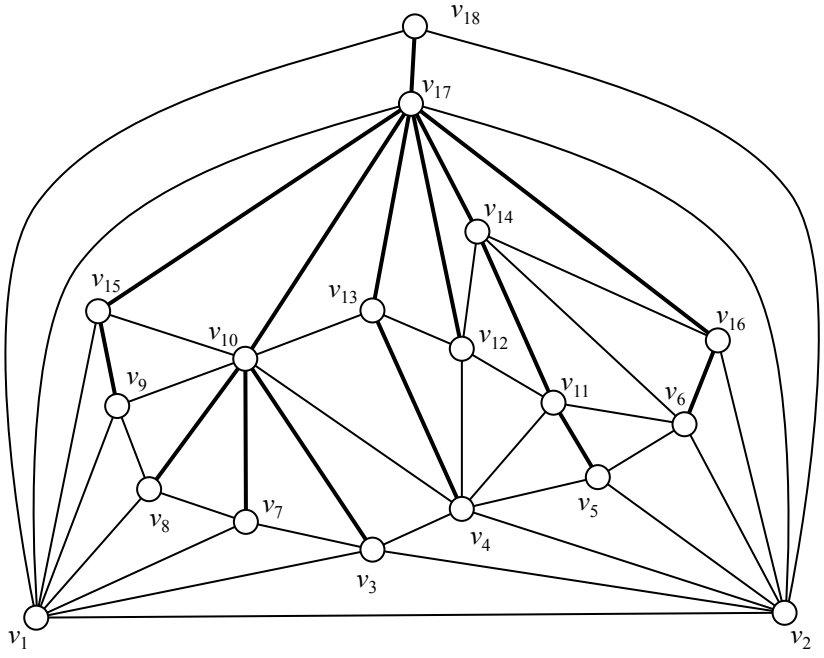


Figure 6.6 Cover tree induced by a canonical ordering of a maximal plane graph. The edges of the tree are drawn with thick lines.

6.6 Shift Method

de Fraysseix, Pach, and Pollack [dFPP90] presented an algorithm for constructing a planar straight-line drawing of an n-vertex maximal plane graph on the $(2n-4) \times (n-2)$ grid. The algorithm is summarized as follows:[2]

- the vertices are placed on the grid one at a time following a canonical ordering (see Definition 6.1) of the input graph;
- at each step, the contour of the drawing of the current graph satisfies certain invariants that involve restrictions on the slopes of the contour edges;
- when a vertex is placed on the grid, some of the previously placed vertices are shifted leftward and some others are shifted rightward to accommodate the new vertex while maintaining the contour invariants and the planarity of the current drawing.

6.6.1 Construction

We now give a detailed description of the algorithm. Let G be an n-vertex maximal plane graph, and let $v_1, \ldots, v_n$ be a canonical ordering of G. We denote by $P(v) = (x(v), y(v))$ the current position of vertex v on the grid. For each vertex v, we maintain the set of vertices that need to be shifted whenever v is shifted; we denote this set by $L(v)$.

As described in Section 6.5, for each $3 \leq k \leq n$, we denote by $v_1 = w_1, w_2, \ldots, w_t = v_2$ the sequence of vertices C_{k-1} (the external face of graph G_{k-1}) when traversed in clockwise order, and by $w_p, \ldots, w_q$ the subsequence of vertices of C_{k-1} that are adjacent to vertex v_k. We call w_p the *left attachment* of v_k and w_q the *right attachment* of v_k. Note that vertices $w_{p+1}, \ldots, w_{q-1}$ are covered by v_k.

For two grid points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, we denote by $\mu(P_1, P_2)$ the intersection of the line with slope $+1$ passing through P_1 and the line with slope -1 passing through P_2 (see Figure 6.7), i.e.,

$$\mu(P_1, P_2) = \left(\frac{x_2 + x_1 + y_2 - y_1}{2}, \frac{x_2 - x_1 + y_2 + y_1}{2} \right) \tag{6.6}$$

Note that if the Manhattan distance between P_1 and P_2 is even, then $\mu(P_1, P_2)$ is a grid point.

Initially, we set $P(v_1) = (-1, 0)$, $P(v_2) = (1, 0)$, and $P(v_3) = (0, 1)$, i.e., we draw G_3 as a triangle Γ_3; we also define *shift sets* $L(v_i) = \{v_i\}, 1 \leq i \leq 3$.

For each $4 \leq k \leq n$, we assume that a planar straight-line grid drawing Γ_{k-1} of G_{k-1} has been constructed in such a way that the following *contour conditions* hold (see Figure 6.8):

1. $P(v_1) = (-((k-1)-2), 0)$ and $P(v_2) = ((k-1)-2, 0)$;
2. $x(w_1) < x(w_2) < \cdots < x(w_{t-1}) < x(w_t)$;
3. each segment $P(w_i)P(w_{i+1}), 1 \leq i \leq t-1$, has slope either $+1$ or -1.

Note that, by Condition 3, the Manhattan distance between any two vertices of C_{k-1} is even; thus, $\mu(P(w_p), P(w_q))$ is a grid point.

[2] Our description of the algorithm, which uses left shifts and right shifts, is slightly different from the one given in [dFPP90], which uses only right shifts, but is conceptually equivalent.

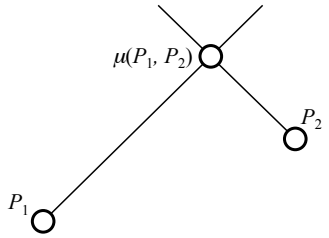


Figure 6.7 Definition of point $\mu(P_1, P_2)$ as the intersection of the line with slope $+1$ passing through P_1 and the line with slope -1 passing through P_2.

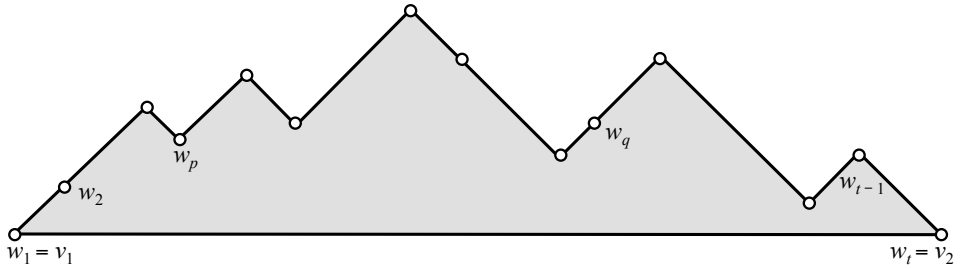


Figure 6.8 Schematic illustration of a drawing of Γ_{k-1} that satisfies the contour conditions, i.e., the external face is drawn as a polygon consisting of a horizontal edge and a chain of segments with slope $+1$ or -1 between endpoints $P(v_1) = (-((k-1)-2), 0)$ and $P(v_2) = ((k-1)-2, 0)$.

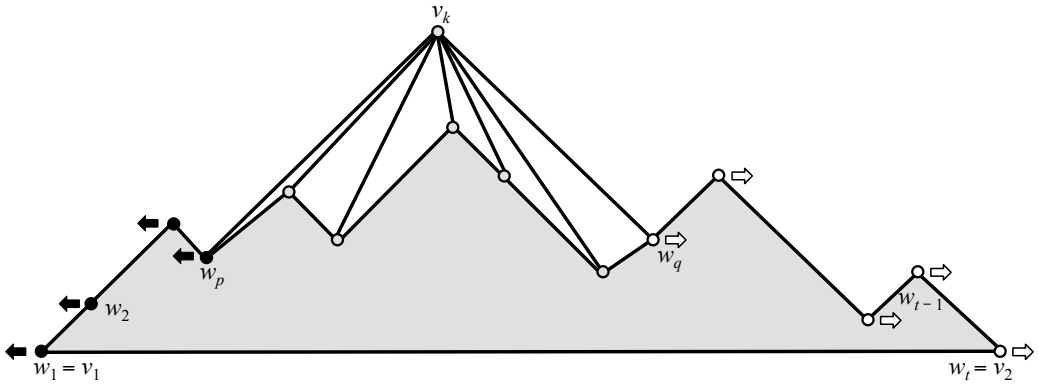


Figure 6.9 Schematic illustration of the addition of vertex v_k to drawing Γ_{k-1} to obtain drawing Γ_k. Contour vertices $w_1, \ldots, w+p$ (black-filled) are shifted by one unit to the left and contour vertices $w_q, \ldots, w_t$ (white-filled) are shifted by one unit to the right. When a contour vertex is shifted, we also shift all the vertices in its shift set (not shown). Finally, vertex v_k is placed at point $\mu(P(w_p), P(w_q))$. Drawing Γ_k satisfies the contour conditions, i.e., the external face is drawn as a polygon consisting of a horizontal edge and a chain of segments with slope $+1$ or -1 between endpoints $P(v_1) = (-k-2, 0)$ and $P(v_2) = (k-2, 0)$.

We now show how to add point $P(v_k)$ to Γ_{k-1} and obtain a planar straight-line drawing Γ_k of G_k (see Figure 6.9):

Step 1 For each $v \in \bigcup_{i=1}^{p} L(w_i)$, set $x(v) = x(v) - 1$. This step translates leftward by 1 the vertices of the external face from w_1 to the left attachment w_p of v_k plus all the other vertices in the shift sets of these vertices.

Step 2 For each $v \in \bigcup_{i=q}^{t} L(w_i)$, set $x(v) = x(v) + 1$. This step translates rightward by 1 the vertices of the external face from the right attachment v_q of v_k to w_t plus all the other vertices in the shift sets of these vertices.

Step 3 Set $P(v_k) = \mu(P(w_p), P(w_q))$. This step places vertex v_k so that it can be joined with straight-line edges to its neighbors.

Step 4 Set $L(v_k) = \{v_k\} \cup (\bigcup_{i=p+1}^{q-1} L(w_i))$. This step defines shift set $L(v_k)$ as the union of v_k and the shift sets of the vertices covered by v_k.

Steps 1, 2, and 3 ensure that points $P(w_p), \ldots, P(w_q)$ are all visible from $P(v_k)$, i.e., segments $P(v_k)P(w_i), p \leq i \leq q$, can be added to Γ_{k-1} without introducing crossings. Conditions 1–3 above are clearly satisfied in Γ_k. By Step 4, we obtain inductively that each set $L(u)$ is the subtree of the cover forest rooted at vertex u. Thus, sets $L(w_1), \ldots, L(w_t)$, form a partition of the vertices of G_{k-1}. It remains to prove that the shift operations in Steps 1 and 2 preserve the planarity of Γ_{k-1}, and this is done in the following lemma.

LEMMA 6.3 Let Γ_j be a planar straight-line drawing of G_j, as described above, and let $v_1 = w'_1, w'_2, \ldots, w'_{t'} = v_2$ be the sequence of vertices of C_j. Let s be an index such that $1 \leq s \leq t'$. If, for each $1 \leq i \leq s$ (resp., $s \leq i \leq t'$), we shift the vertices in $L(w'_i)$ leftward (resp., rightward) by a positive integer number ρ, then the resulting straight-line drawing is still planar.

Proof: By induction on j. For Γ_3 the lemma is trivially true. We now suppose that the lemma is true for $\Gamma_{j-1}, j \geq 4$, and prove that it is true for Γ_j. We use the notation from the algorithm description above; namely, $v_1 = w_1, w_2, \ldots, w_t = v_2$ is the sequence of vertices of C_{j-1}, and w_p and w_q are the leftmost and rightmost neighbors of v_j in C_{j-1}, respectively. We denote by ζ the difference between the number of vertices of C_{j-1} and the number of vertices of C_j, i.e., $\zeta = (q - p - 1) - 1 \geq -1$. Thus, we have:

$$t' = t - \zeta$$

$$w'_i = \begin{cases} w_i & \text{for } i = 1, \ldots, p \\ v_j & \text{for } i = p+1 \\ w_{i+\zeta} & \text{for } i = p+2, \ldots, t' \end{cases}$$

Note, in particular, that $w'_{p+2} = w_q$. We prove the claim for the rightward shift; the proof for the leftward shift is symmetric.

If $s > p + 2$, then v_j and its neighbors $w_p, \ldots, w_q$ in C_{j-1} do not move. Thus, by the induction hypothesis, Γ_j is planar.

If $s \leq p$, then v_j and its neighbors $w_p, \ldots, w_q$ in C_{j-1} shift rigidly rightward by ρ. Thus, by the induction hypothesis, Γ_j is planar.

If $s = p+1$, we apply the induction hypothesis to Γ_{j-1} with $s = p+1$; thus, the planarity of Γ_{j-1} is preserved. Vertex v_j and its neighbors $w_{p+1}, \ldots, w_q$ in C_{j-1} shift rigidly rightward by ρ, while w_p does not move. Point $P(w_p)$ is clearly still visible from points $P(v_j)$ and $P(w_{p+1})$, and thus, Γ_j is planar.

If $s = p+2$, we apply the induction hypothesis to Γ_{j-1} with $s = q$; thus, the planarity of Γ_{j-1} is preserved. Vertex v_j and its neighbors $w_p, \ldots, w_{q-1}$ in C_{j-1} do not move, while w_q

shifts rightward by ρ. Point $P(w_q)$ is clearly still visible from points $P(v_j)$ and $P(w_{q-1})$, and thus Γ_j is planar. □

In the end, we obtain a planar straight-line drawing of G in which $P(v_1) = (-(n-2), 0)$ and $P(v_2) = (n-2, 0)$. By Condition 3 above, $P(v_n) = (0, n-2)$. Therefore, G is drawn on the $(2n-4) \times (n-2)$ grid.

Figures 6.10 through 6.19 show several steps of the execution of the algorithm on the graph and canonical ordering of Figure 6.5. The final drawing is shown in Figure 6.20.

6.6.2 Implementation

A straightforward implementation of the shift method results in an $O(n^2)$-time algorithm. In their paper, de Fraysseix, Pach, and Pollack [dFPP90] were able to reduce this time bound to $O(n \log n)$. An optimal $O(n)$-time implementation of the shift method was presented by Chrobak and Payne [CP95], and this is the implementation we describe below.

The crucial observation is that, when vertex v_k is placed on the grid, it is not necessary to know the exact positions of w_p and w_q. If their y-coordinates and their x-offset, i.e., $x(w_q) - x(w_p)$, are known, then $y(v_k)$ and the x-offset between v_k and w_p can be computed; namely, by Eq. 6.6, we have

$$y(v_k) = \frac{x(w_q) - x(w_p) + y(w_q) + y(w_p)}{2}, \tag{6.7}$$

$$x(v_k) - x(w_p) = \frac{x(w_q) - x(w_p) + y(w_q) - y(w_p)}{2}. \tag{6.8}$$

The algorithm consists of three phases. In the first phase, we compute a canonical ordering of the input graph. In the second phase, we add vertices one at a time, according to that canonical ordering: for each added vertex v_k, we compute its y-coordinate and x-offset $x(v_k) - x(w_p)$, update the x-offset of w_q (from its previous value $x(w_q) - x(w_{q-1})$) to $x(w_q) - x(v_k)$, and possibly update the x-offset of w_{p+1}. In the third phase, we suitably traverse the graph starting from v_1 and compute the final x-coordinates of the vertices by accumulating offsets.

We now describe the data structure used to implement the algorithm. For each $4 \leq k \leq n$, the family of sets $L(w_1), \ldots, L(w_t)$ for vertices $w_1, \ldots, w_t$ of C_{k-1} can be viewed as an ordered forest F of trees $L(w_i)$ rooted at vertex $w_i, 1 \leq i \leq t$. When vertex v_k is added and set $L(v_k)$ is created (see Step 4 above), a new tree $L(v_k)$ of F is created out of trees $L(w_{p+1}), \ldots, L(w_{q-1})$ by making v_k the parent of $w_{p+1}, \ldots, w_{q-1}$ (in this order from left to right). A standard way to represent an ordered forest F is by means of a binary tree T: the roots of the trees of F are all considered siblings; the root of T corresponds to the root of the first tree of F; if n_T is a node of T corresponding to a node n_F of F, then the left child of n_T corresponds to the leftmost child of n_F (if any), and the right child of n_T corresponds to the next sibling of n_F (if any).

In our context, the root of T corresponds to $v_1 = w_1$, its right child corresponds to w_2, its right child's right child corresponds to w_3, and so on; thus, the rightmost leaf corresponds to $w_t = v_2$. Tree $L(w_i), 1 \leq i \leq t$, is represented by the node corresponding to w_i and its left subtree. The subtree of T rooted at the node corresponding to w_i represents $\bigcup_{j \geq i} L(w_j)$. For brevity, in the rest of the section, we refer with the same symbol to a vertex of G, the corresponding node of F, and the corresponding node of T.

If u is an ancestor of v in T, the *x-offset* between v and u is defined as $\Delta x(v, u) = x(v) - x(u)$. If u is the parent of v, we simply use the term x-offset of v and the symbol $\Delta x(v)$. With each vertex v of G, we store the following information:

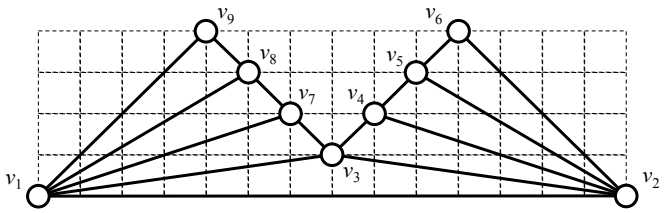


Figure 6.10 Drawing Γ_9 of graph G_9 which consists of vertices $v_1, v_2, \ldots, v_9$.

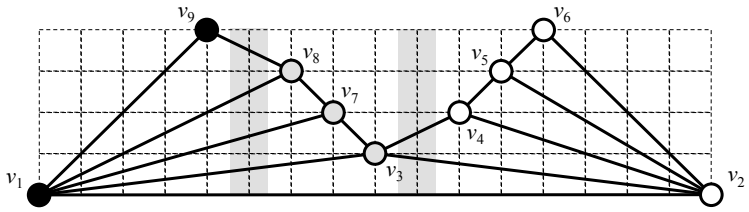


Figure 6.11 Preparing to add vertex v_{10} to drawing Γ_9. Vertex v_{10} has left attachment v_9 and right attachment v_4: the black-filled vertices are shifted to the left by one unit; the gray-filled vertices do not move; and the white-filled vertices are shifted to the right by one unit.

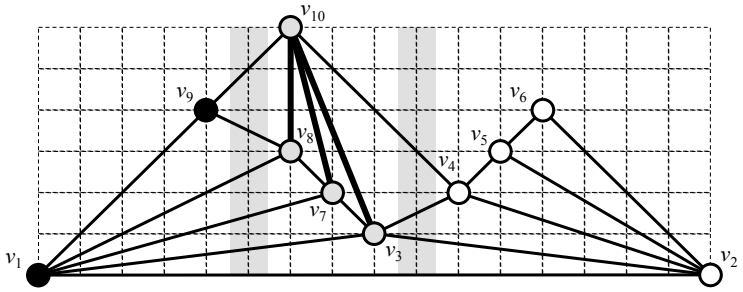


Figure 6.12 Addition of vertex v_{10} and its incident edges, which yields drawing Γ_{10}. Vertex v_{10} covers vertices v_8, v_7, and v_3.

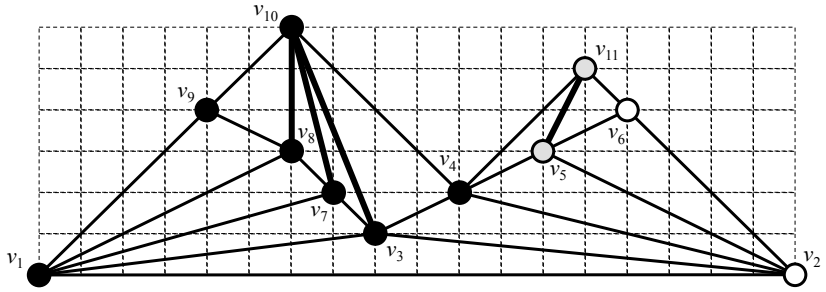


Figure 6.13 Drawing Γ_{11} obtained by adding vertex v_{11} and its incident edges after shifting the black-filled vertices to the left and the white-filled vertices to the right. Vertex v_{11} covers vertex v_5.

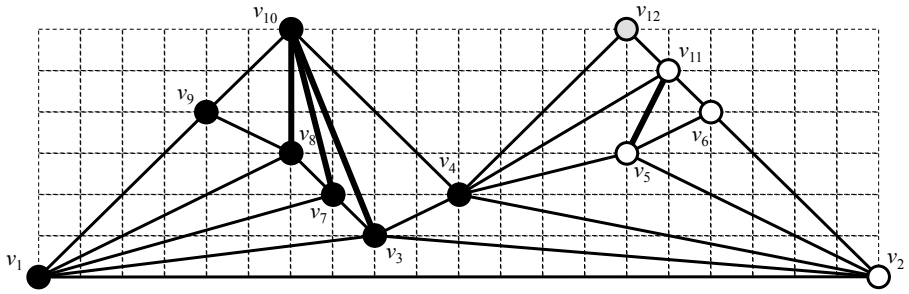


Figure 6.14 Drawing Γ_{12}.

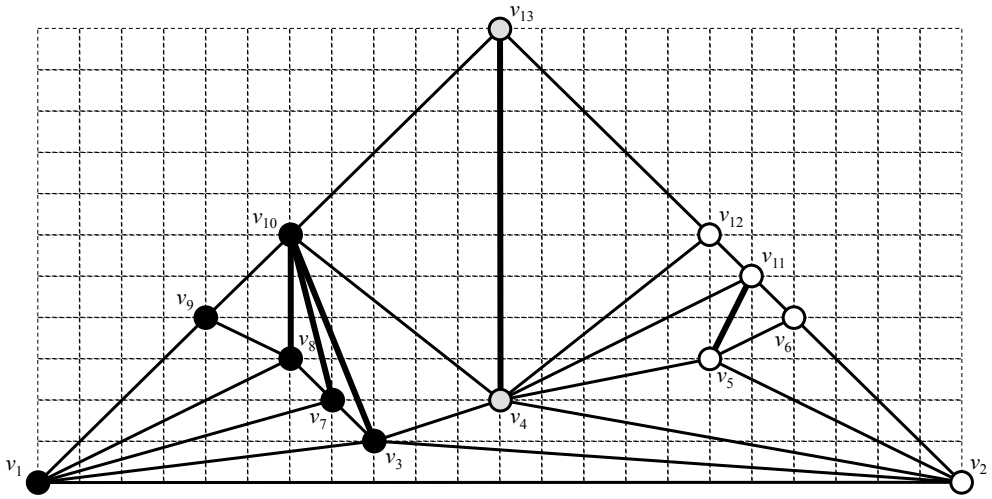


Figure 6.15 Drawing Γ_{13}.

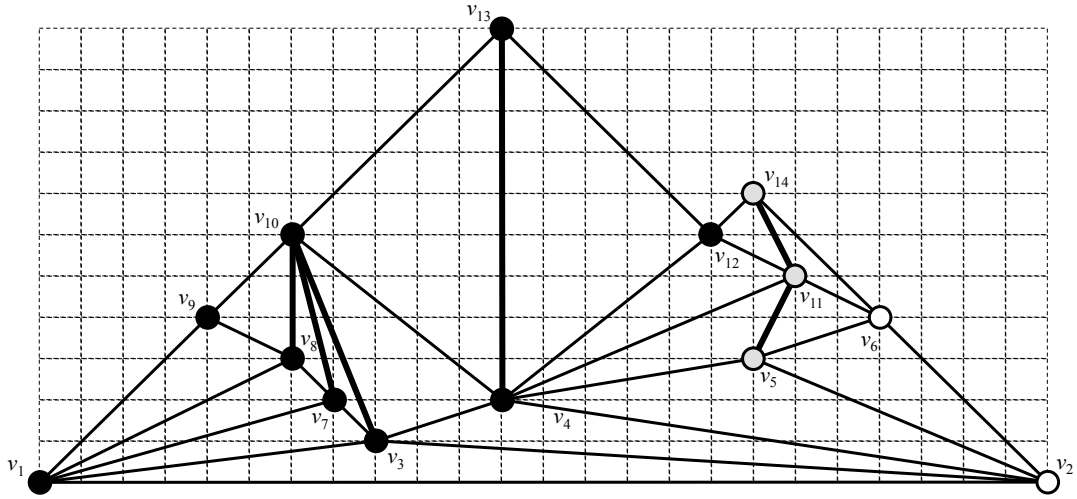


Figure 6.16 Drawing Γ_{14}.

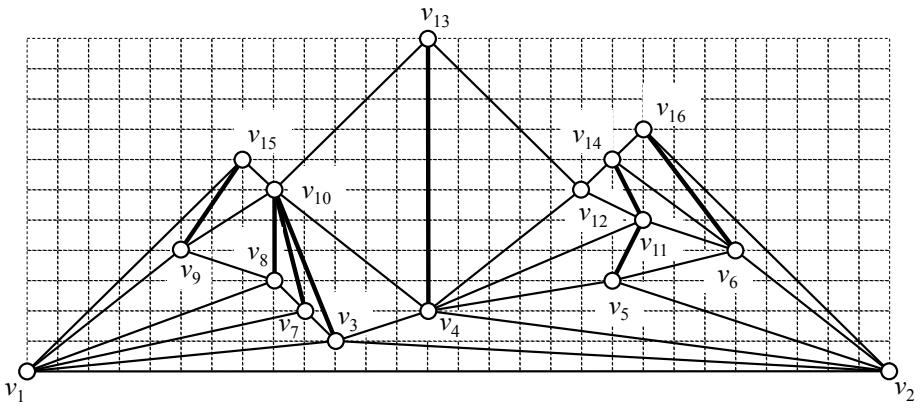


Figure 6.17 Drawing Γ_{16}. Note that we have skipped drawing Γ_{15}. Also, here and in the next two figures we do not fill the vertices to denote the amount of shifting.

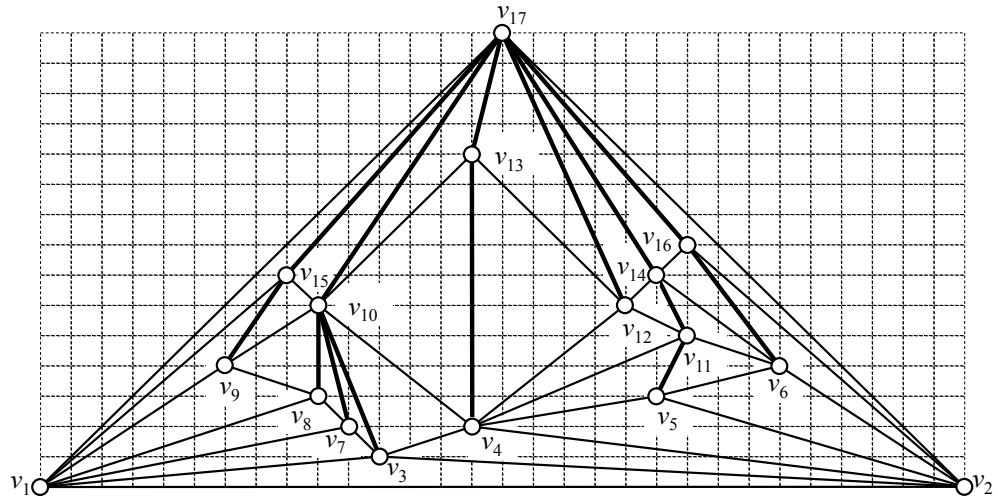


Figure 6.18 Drawing Γ_{17}.

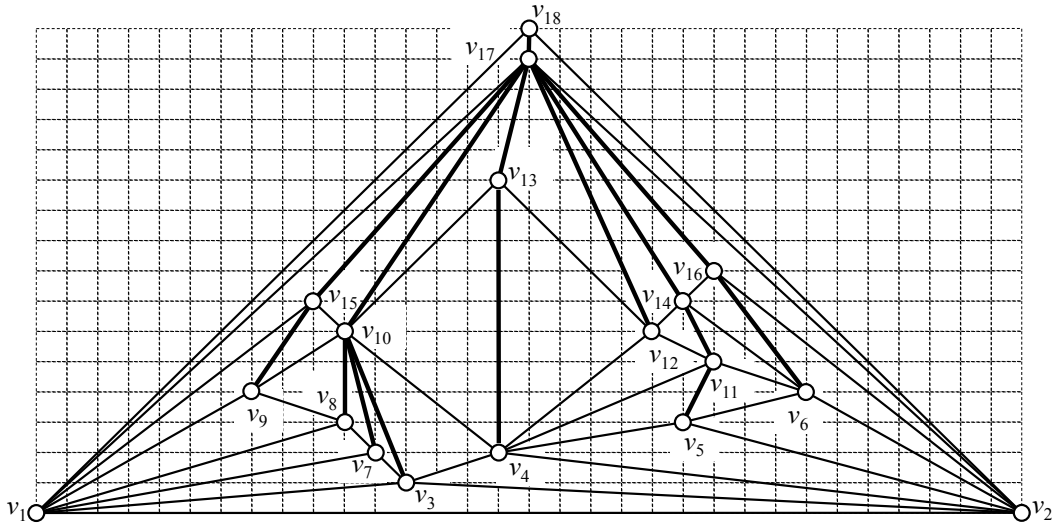


Figure 6.19 Drawing Γ_{18} of the graph of Figure 6.5.

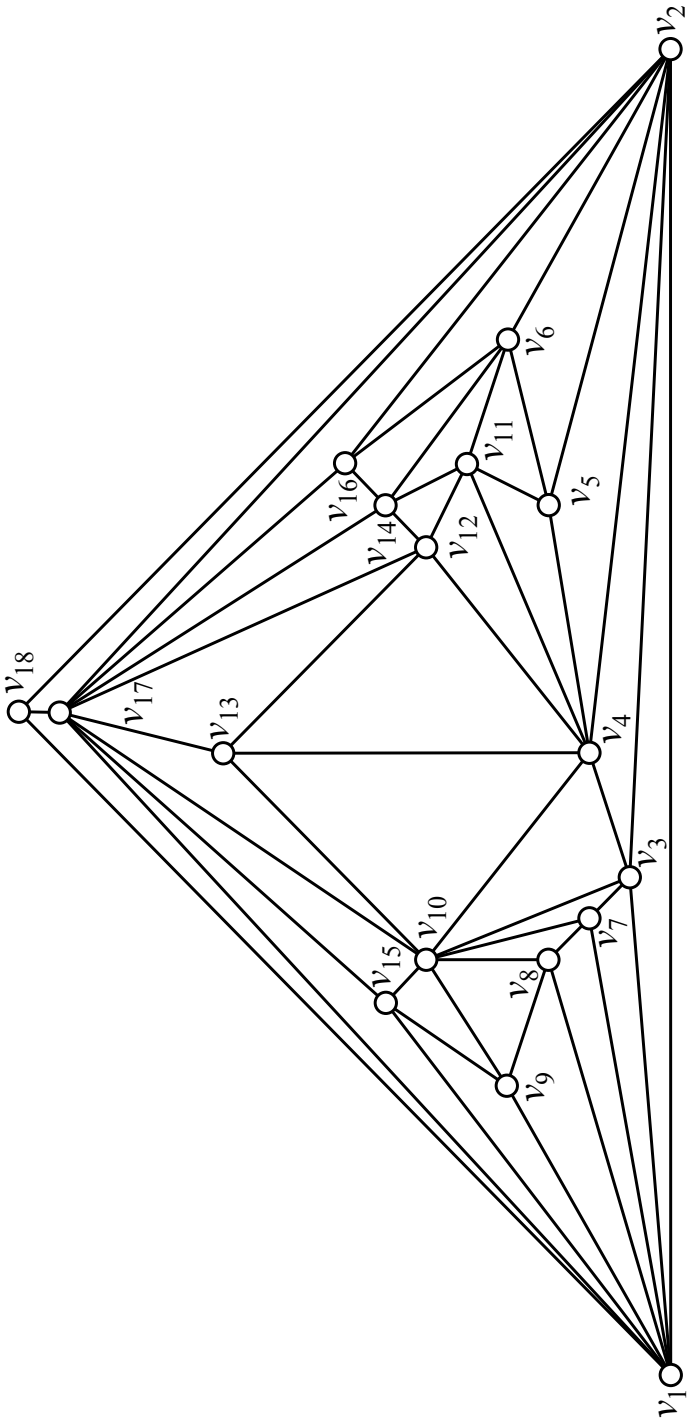


Figure 6.20 Planar straight-line grid drawing of the graph of Figure 6.5 constructed with the shift method by de Fraysseix, Pach, and Pollack (Algorithm MaximalShift shown in Figure 6.21). The graph has $n = 18$ vertices and the drawing has width $2n - 4 = 32$ and height $n - 2 = 16$. Note that the drawing is the same as that of Figure 6.19 except that it has been rotated counterclockwise by 90 degrees and the grid lines have been omitted for better readability.

- $\Delta x(v)$, the x-offset of v;
- $y(v)$, the y-coordinate of v;
- $left(v)$, the left child of v in T;
- $right(v)$, the right child of v in T.

The pseudo-code of the algorithm, which we call MaximalShift, is given in Figure 6.21. The first phase of the algorithm consists of line 1. The second phase consists of lines 2–30; note that, on line 23, the right child of w_{q-1}, which before the insertion of v_k was w_q, is set to *nil* since now $w_{q-1} \notin C_k$ and w_q is the right child of v_k. The third phase consists of lines 31–32, where the x-coordinate of v_1 is set to 0 and recursive procedure AccumulateOffset is called. The pseudo-code of procedure AccumulateOffset is given in Figure 6.22. It performs a preorder visit of T and computes the x-coordinate of each vertex $v \neq v_1$ of G as the sum of the x-coordinate of the parent of v in T and the x-offset of v (line 1).

The following theorem summarizes the area bound and computational complexity of the shift method.

Theorem 6.1 **[dFPP90, CP95]** *Let G be a maximal plane graph with n vertices. The shift method computes a planar straight-line drawing of G on the $(2n-4) \times (n-2)$ grid in $O(n)$ time and space.*

Proof: We refer to Algorithm MaximalShift, shown in Figure 6.21. Clearly, the x-coordinate of each vertex $v \neq v_1$ of G can be computed by adding the x-offset $\Delta x(v, v_1)$ between v and v_1 to the x-coordinate $x(v_1)$ of v_1 (the root of T). Thus, we only have to prove that those x-offsets are correct at the end of the second phase of the algorithm. Note that the only steps of the algorithm where the x-offsets of some vertices of the current graph G_{k-1} are modified are those on lines 7–8 and on lines 26–29.

For the "stretch" step on lines 7–8, we recall that the subtree $T(w_i)$ of T rooted at w_i represents $\bigcup_{j \geq i} L(w_j)$; thus, incrementing $\Delta x(w_i)$ increments the x-offset between each vertex of $T(w_i)$ and v_1, i.e., correctly shifts all vertices in $\bigcup_{j \geq i} L(w_j)$ rightward, or, equivalently, all vertices in $\bigcup_{j < i} L(w_j)$ leftward.

During the "adjust" step on lines 26–29, only $\Delta x(w_q)$ and possibly $\Delta x(w_{p+1})$ are modified. Note that, after the insertion of v_k, w_p is still an ancestor of both w_q and w_{p+1} in T: namely, v_k is the parent of w_{p+1} and w_q, and w_p is the parent of v_k. We now prove that the values of $\Delta x(w_q, w_p)$ and $\Delta x(w_{p+1}, w_p)$ are not modified by the insertion of v_k.

- After the insertion of v_k we have $\Delta x(w_q, w_p) = \Delta x(w_q, v_k) + \Delta x(v_k, w_p) = \Delta x(w_q) + \Delta x(v_k)$, which, by the choice of $\Delta x(w_q)$ on line 26, is clearly equal to the value of $\Delta x(w_q, w_p)$ before the insertion of v_k, computed on line 10.
- If $p + 1 \neq q$, after the insertion of v_k we have $\Delta x(w_{p+1}, w_p) = \Delta x(w_{p+1}, v_k) + \Delta x(v_k, w_p) = \Delta x(w_{p+1}) + \Delta x(v_k)$, which, by the choice of $\Delta x(v_{p+1})$ on line 28, is clearly equal to the value of $\Delta x(w_{p+1})$ before the insertion of v_k.

It follows that, for each vertex $v \neq v_1 \in G_{k-1}$, x-offset $\Delta x(v, v_1)$ is not modified during the "adjust" step. Hence, algorithm MaximalShift is a correct implementation of the shift method.

As for its space and time complexity, the data structure used to implement the algorithm clearly takes $O(n)$ space. By Lemma 6.1, the first phase takes $O(n)$ time and space. The time complexity of the body (lines 6–29) of the main for loop is dominated by the computation of $\Delta x(w_q, w_p)$ on line 10, which takes $O(\deg(v_k))$; thus, by Eq. 6.2, the second phase globally takes $O(n)$ time. The third phase clearly takes $O(n)$ time since at the end of the second phase T has n nodes. □

Input: A maximal plane graph G with n vertices
Output: A planar straight-line drawing of G on the $(2n-4) \times (n-2)$ grid

```
1: compute a canonical ordering v_1, ..., v_n of G
2: (Δx(v_1), y(v_1), left(v_1), right(v_1)) ← (0, 0, nil, v_3)
3: (Δx(v_3), y(v_3), left(v_3), right(v_3)) ← (1, 1, nil, v_2)
4: (Δx(v_2), y(v_2), left(v_2), right(v_2)) ← (1, 0, nil, nil)
5: for 4 ≤ k ≤ n do
6:    /* stretch the L(w_p)-to-L(w_{p+1}) and L(w_{q-1})-to-L(w_q) gaps */
7:    Δx(w_{p+1}) ← Δx(w_{p+1}) + 1
8:    Δx(w_q) ← Δx(w_q) + 1
9:    /* compute Δx(w_q, w_p) */
10:   Δx(w_q, w_p) ← Δx(w_{p+1}) + ··· + Δx(w_q)
11:   /* compute Δx(v_k) and y(v_k); see Eqs. 6.8 and 6.7 */
12:   Δx(v_k) ← (Δx(w_q, w_p) + y(w_q) − y(w_p))/2
13:   y(v_k) ← (Δx(w_q, w_p) + y(w_q) + y(w_p))/2
14:   /* add v_k to T */
15:   right(w_p) ← v_k
16:   if p + 1 ≠ q then
17:      left(v_k) ← w_{p+1}
18:   else
19:      left(v_k) ← nil
20:   end if
21:   right(v_k) ← w_q
22:   if q − 1 ≠ p then
23:      right(w_{q−1}) ← nil
24:   end if
25:   /* adjust Δx(w_q) and Δx(w_{p+1}) */
26:   Δx(w_q) ← Δx(w_q, w_p) − Δx(v_k)
27:   if p + 1 ≠ q then
28:      Δx(w_{p+1}) ← Δx(w_{p+1}) − Δx(v_k)
29:   end if
30: end for
31: x(v_1) ← 0
32: AccumulateOffset(v_1,x(v_1))
```

Figure 6.21 Algorithm MaximalShift.

Input: A vertex v of T and an integer x

```
1: if v ≠ nil then
2:    x(v) ← x + Δx(v)
3:    AccumulateOffset(left(v),x(v))
4:    AccumulateOffset(right(v),x(v))
5: end if
```

Figure 6.22 Procedure AccumulateOffset.

6.6.3 Refinements and Variations

Chrobak and Nakano [CN98] and Brandenburg [Bra08] refined the shift method by de Fraysseix, Pach, and Pollack [dFPP90], thus reducing the area of the drawing.

In the original shift method [dFPP90], we have seen that at each step the drawing satisfies the contour conditions. In the refinement by Chrobak and Nakano [CN98], these conditions are relaxed: $x(w_i) \leq x(w_{i+1}), 1 \leq i \leq t-1$ and the equality may hold only when $y(w_i) < y(w_{i+1})$. Thus, each contour segment $P(w_i)P(w_{i+1})$ belongs to one of the following four types:

vertical $x(w_i) = x(w_{i+l})$ and $y(w_i) < y(w_{i+l})$;
upward $x(w_i) < x(w_{i+l})$ and $y(w_i) < y(w_{i+l})$;
horizontal $x(w_i) < x(w_{i+l})$ and $y(w_i) = y(w_{i+l})$;
downward $x(w_i) < x(w_{i+l})$ and $y(w_i) > y(w_{i+l})$.

The presence of vertical contour segments allows to avoid some shifts, thus obtaining a more compact drawing. The authors present a new combinatorial structure, called a *domino chain*, which allows to partition the vertices into *stable* and *unstable*; a stable vertex v_k can be added to G_{k-1} with edge (w_p, v_k) drawn as a vertical segment and no shift is necessary. Namely, the method avoids making any shifts in approximately $\frac{n}{3}$ steps and results in a drawing of size $\lfloor \frac{2}{3}(n-1) \rfloor \times 4 \lfloor \frac{2}{3}(n-1) \rfloor - 1$.

Brandenburg further improves the shifting strategy and also rotates the drawing to choose the best base edge. This refinement of the shift method results in a drawing of size $\lceil \frac{4}{3}n \rceil \times \lceil \frac{2}{3}n \rceil$. Also, this height and width are necessary if the drawing is constrained to be enclosed by an isosceles right-angled triangle.

Kant [Kan96] presents an algorithm based on the shift method for constructing convex drawings of triconnected plane graphs on the $(2n-4) \times (n-2)$ grid. The size of the grid is reduced to $(n-2) \times (n-2)$ in a successive algorithm by Chrobak and Kant [CK97].

6.7 Realizer Method

An alternative method for drawing maximal planar graphs on an integer grid was presented by Schnyder [Sch90]. The origins of the approach can be found in [Sch89], where it was used to characterize planar graphs as the graphs whose incidence relation is the intersection of at most three total orders[3] (see Theorems 4.1 and 6.2 of [Sch89]).

6.7.1 Realizers

DEFINITION 6.3 A *realizer* of a maximal plane graph G is a triplet of rooted directed spanning trees of G with the following properties[4] (see Figure 6.23):

[3] More formally, a graph $G = (V, E)$ is planar if and only if the order dimension of the poset $(V \cup E, \prec)$, where incidence relation $\prec$ is defined by $v \prec e \Leftrightarrow v \in V, e \in E, v \in e$, is at most 3. The *order dimension* of a poset is the minimum cardinality of its realizers. A *realizer* of a poset $(X, \prec)$ is a nonempty set of total orders on X whose intersection is $\prec$.

[4] This definition of a realizer of a maximal plane graph is slightly different from the one given in [Sch90], as we consider also the external edges; our definition allows to reduce the number of special cases and to generalize the concept of realizer to triconnected plane graphs.

1. In each spanning tree, the edges of G are directed from children to parent.
2. The sinks (roots) of the spanning trees are the three external vertices of G.
3. Each internal edge of G is contained in one spanning tree.
4. Each external edge of G is contained in two spanning trees and it has different directions in the two trees.
5. Consider the edges of G with the directions they have in the three spanning trees (the external edges are considered twice):
   (a) Each non-sink vertex v of G has exactly three *outgoing edges*; the circular order of the outgoing edges around v induces a circular order of the spanning trees around v; all the non-sink vertices of G have the same circular order of the spanning trees.
   (b) For each vertex of G, the *incoming edges* that belong to the same spanning tree appear consecutively between the outgoing edges of the other two spanning trees (for the sink of each spanning tree the first and last incoming edges are coincident with the two outgoing edges).
6. For the sink of each spanning tree, all the incoming edges belong to that spanning tree.

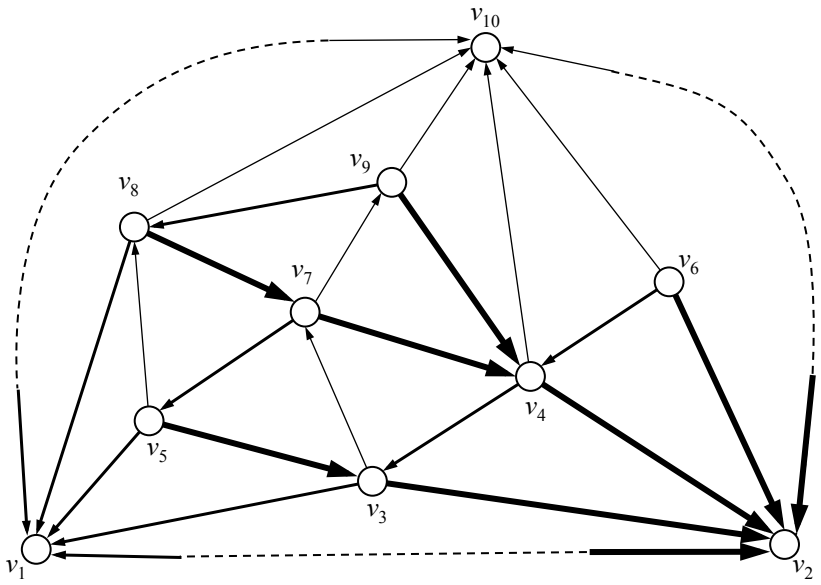


Figure 6.23 A realizer of a maximal plane graph whose vertices are numbered according to a canonical ordering. The edges are thick for the green spanning tree, medium for the blue spanning tree, and thin for the red spanning tree. Note the 2-colored edges on the external face.

Let T_b, T_g, and T_r be the spanning trees forming a realizer of a maximal plane graph G (see Figure 6.23). We assign a color to the edges of G contained in T_b, T_g, and T_r, say, *blue*, *green*, and *red*, respectively. In the figures, we use dark grey for blue, light grey for green, and medium grey for red. According to Property 3 of the realizers, each internal edge of

G is assigned one color and is said to be 1-*colored*, while the three external edges of G are assigned two colors and are said to be 2-*colored*.

In the proof of the following lemma, we present a mechanism for constructing a realizer of a maximal plane graph G based on a canonical ordering of G; this is different from the mechanism based on edge labelings presented in [Sch90].

LEMMA 6.4 Each maximal plane graph has a realizer, which can be computed in linear time and space.

Proof: Let G be a maximal plane graph. A realizer of G can be constructed by assigning colors and directions to the edges of G as follows:

1. a canonical ordering of G is computed;
2. v_1, v_2, and v_n are the sinks of the blue, green, and red tree, respectively;
3. (v_1, v_2) is an outgoing blue edge for v_2 and an outgoing green edge for v_1;
4. for each $3 \leq k \leq n$, let $c_l, \ldots, c_r$ be the consecutive neighbors of v_k on C_{k-1} from left to right; (v_k, c_l) is an outgoing blue edge for v_k; (v_k, c_r) is an outgoing green edge for v_k; each edge $(v_k, c_i), l < i < r$, is an outgoing red edge for c_i (see Figure 6.23);
5. (v_n, v_1) is also an outgoing red edge for v_1, and (v_n, v_2) is also an outgoing red edge for v_2.

Note that v_1 has no outgoing blue edge, v_2 has no outgoing green edge, and v_n has no outgoing red edge. Besides, for each $3 \leq k \leq n$, the following invariants hold for G_k:

- v_k has exactly one outgoing blue edge, exactly one outgoing green edge, and no outgoing red edge; the outgoing blue edge precedes the outgoing green edge in the clockwise circular order of the edges of C_k, and all the (possible) incoming red edges are incident with vertices of C_{k-1};
- for every vertex of C_k the (possible) incoming blue edge of C_k follows the (possible) incoming green edge of C_k in the clockwise circular order of the edges of C_k;
- no vertex of C_{k-1} has an outgoing blue or green edge incident with v_k;
- every vertex of C_{k-1} with no neighbor in $G - G_k$ has exactly one outgoing red edge, while every vertex of C_{k-1} with neighbors in $G - G_k$ has no outgoing red edge;
- G_k contains no cycle such that a common color is assigned to all its edges.

All the properties of a realizer easily follow from these invariants. By Lemma 6.1, the above construction can be carried out in linear time and space. □

From the construction in the proof of Lemma 6.4, it follows that, for every realizer of a maximal plane graph G, all internal edges of G are 1-colored, while the three external edges are 2-colored. Also, for each vertex of G, the colors of the three outgoing edges appear in the following counterclockwise circular order: blue, green, red. Set $\{b, g, r\}$ will be considered accordingly ordered in the rest of the chapter.

In the rest of the section, we consider a maximal plane graph G equipped with a realizer $\{T_b, T_g, T_r\}$. We denote v_1, v_2, and v_n by s_b, s_g, and s_r, respectively. For each vertex v of G, the *blue path* $p_b(v)$ is the path of G along T_b from v to s_b. In the same way, we define the *green path* $p_g(v)$ as the path of G along T_g from v to s_g and the *red path* $p_r(v)$ as the path

of G along T_r from v to s_r. Note that $p_i(s_i), i \in \{b, g, r\}$, is a degenerate path consisting only of s_i. The subpath of $p_i(v), i \in \{b, g, r\}$, from v to the ancestor u of v in T_i is denoted by $p_i(v, u)$. The parent of vertex v in $T_i, i \in \{b, g, r\}$, is denoted by $par_i(v)$. The lowest common ancestor of vertices u and v in $T_i, i \in \{b, g, r\}$, is denoted by $lca_i(u, v)$.

LEMMA 6.5 For each vertex v of G, $p_b(v)$, $p_g(v)$, and $p_r(v)$ have only vertex v in common.

Proof: W.l.o.g., suppose, for a contradiction, that $p_b(v)$ and $p_g(v)$ have a vertex u in common, and that $p_b(v, u) - \{u, v\}$ and $p_g(v, u) - \{u, v\}$ have no vertex in common with each other and with $p_r(v)$. Vertex u has both a blue and a green incoming edge; thus, by Property 6 of the realizers, we have $u \neq s_b, s_g$. Let R be the subgraph of G bounded by $p_b(v, u)$ and $p_g(v, u)$; from the circular order of the outgoing edges at v, we have that $p_b(v, u)$ (resp., $p_g(v, u)$) follows the boundary of R counterclockwise (resp., clockwise). Thus, by Property 5 of the realizers at u and by the planarity of G, $par_b(u) \in R$ (the same is true for $par_g(u)$). Still by the planarity of G, $p_b(par_b(u))$ leaves R at a vertex w; two cases are possible: (i) $w \in p_g(v, u) - \{u, v\}$, but this contradicts Property 5 of the realizers at w, or (ii) $w \in p_b(v, u)$, but this contradicts the acyclicity of T_b. □

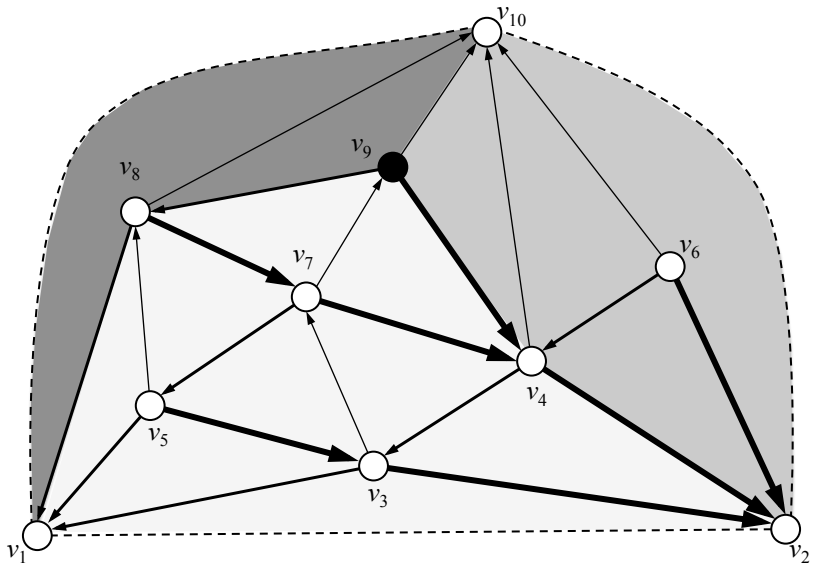


Figure 6.24 The blue (medium), green (thick), and red (thin) paths for vertex v_9 and corresponding blue (medium shaded), green (dark shaded) and red (light shaded) regions of vertex v_9. The coordinates of v_9 in the barycentric representation are the number of faces in the blue, green, and red region, respectively, i.e., $(4, 2, 9)$.

For each vertex v of G, the *blue region* $R_b(v)$ is the subgraph of G bounded by $p_g(v)$, $p_r(v)$ and (s_g, s_r). In the same way, the *green region* $R_g(v)$ is the subgraph of G bounded by $p_b(v)$, $p_r(v)$ and (s_r, s_b), and the *red region* $R_r(v)$ is the subgraph of G bounded by $p_b(v)$, $p_g(v)$ and (s_b, s_g) (see Figure 6.24). Note that $R_b(s_g) = R_b(s_r)$ is a degenerate region consisting only of (s_g, s_r). In the same way, $R_g(s_r) = R_g(s_b) = (s_r, s_b)$ and $R_r(s_b) = R_r(s_g) = (s_b, s_g)$.

LEMMA 6.6 Let u and v be two distinct vertices of G. If $u \in R_k(v), k \in \{b, g, r\}$, then $R_k(u) \subset R_k(v)$.

Proof: W.l.o.g, we assume $k = r$. Two cases are possible: (i) $u \notin p_b(v) \cup p_g(v)$, or (ii) $u \in p_b(v) \cup p_g(v)$. We consider only the first case; the second is similar. By the planarity of G and by Property 5 of the realizers, $p_b(u)$ has no vertex in common with $p_g(v)$, and $p_g(u)$ has no vertex in common with $p_b(v)$. Thus, the region $R_r(v) - R_r(u)$ bounded by $p_b(u, lca_b(u, v))$, $p_g(u, lca_g(u, v))$, $p_b(v, lca_b(u, v))$, and $p_g(v, lca_g(u, v))$ is nonempty; hence, $R_r(u) \subset R_r(v)$. □

Note that, for each $k \in \{b, g, r\}$, the inclusion partial order of the k-regions induces a partial order on the vertices of G defined by $u \prec_k v \Leftrightarrow R_k(u) \subset R_k(v)$. Partial order $\prec_k$ is represented by tree $T_k, k \in \{b, g, r\}$ of the realizer of G. Also, for each edge (u, w) and each vertex $v \neq u, w$ of G, by the planarity of G, (u, w) is in some region $R_k(v), k \in \{b, g, r\}$; hence, $u \prec_k v$ and $w \prec_k v$. Any choice of three linear extensions of $\prec_b$, $\prec_g$, and $\prec_r$, produces a realizer of the poset defined in footnote (3) on page 212.

6.7.2 Barycentric Representation

DEFINITION 6.4 A *barycentric representation* of a graph $G = (V, E)$ is an injective function $f : v \in V \rightarrow (v_b, v_g, v_r) \in \mathbb{Z}^3$ that satisfies the following conditions:

1. For each vertex v of G, $v_b + v_g + v_r = c$, where c is a constant dependent on G.
2. For each edge (u, w) and each vertex $v \neq u, w$ of G, there exists a coordinate $i \in \{b, g, r\}$ such that $v_i > u_i$ and $v_i > w_i$.

One can view v_b, v_g, and v_r as barycentric coordinates of vertex v. Note that these coordinates have a purely combinatorial meaning.

LEMMA 6.7 A barycentric representation $f : v \in V \rightarrow (v_b, v_g, v_r)$ of a graph $G = (V, E)$ is a planar straight-line drawing of G on plane $b + g + r = c$ in $\mathbb{Z}^3$.

Proof: Let π be the plane in $\mathbb{Z}^3$ defined by equation $b + g + r = c$. By Condition 1 of Definition 6.4, all vertices of G are mapped to points of π. In order to prove the planarity of the straight-line drawing, we must prove the following:

- No two vertices are mapped to the same point of π. By definition, since f is injective.
- No vertex overlaps an edge. Let (u, w) be an edge of G and let $max_i = \max\{u_i, v_i\}, i \in \{b, g, r\}$. Let λ_b be the line of π defined by equation $b = max_b$, i.e., the line of π passing through the endpoint of segment $f(u)f(w)$ with maximum b-coordinate and perpendicular to the b axis. Lines λ_g and λ_r are defined in a similar way, and, together with λ_b, identify a (closed) triangle T containing $f(u)f(w)$. Suppose, for a contradiction, that there exists a vertex $v \neq u, w$ of G such that $f(v)$ overlaps $f(u)f(w)$. Clearly, $f(v)$ is contained by T, i.e., $v_i \leq max_i$, for each $i \in \{b, g, r\}$. But this contradicts Condition 2 of Definition 6.4.
- No two edges cross. Let $e_1 = (u, w)$ and $e_2 = (x, y)$ be two nonincident edges of G, and let T_1 and T_2 be the two (closed) triangles containing $f(u)f(w)$ and $f(x)f(y)$,

respectively, identified as above. Suppose, for a contradiction, that $f(u)f(w)$ and $f(x)f(y)$ cross. Then, either T_1 contains $f(x)$ or $f(y)$, or T_2 contains $f(u)$ or $f(v)$. But this again contradicts Condition 2 of Definition 6.4.

□

Lemma 6.7 implies that only a planar graph can have a barycentric representation. For each vertex v of G, we denote by $l_b(v)$, $l_g(v)$, and $l_r(v)$ the number of faces in $R_b(v)$, $R_g(v)$, and $R_r(v)$, respectively. Note that $0 \leq l_b(v), l_g(v), l_r(v) \leq 2n-5$ and

$$l_b(v) + l_g(v) + l_r(v) = 2n - 5.$$

We have that these values yield barycentric coordinates (see Figures 6.24 and 6.25), as shown by the following lemma.

LEMMA 6.8 Let $G = (V, E)$ be a maximal plane graph equipped with a realizer. Function $f : v \in V \rightarrow (l_b(v), l_g(v), l_r(v))$ is a barycentric representation of G.

Proof: The injectivity of f follows from Lemma 6.6. Condition 1 of Definition 6.4 is trivially satisfied since for each vertex v, $v_b + v_g + v_r = 2n - 5$. As for Condition 2, let (u, w) and $v \neq u, w$ be an edge and a vertex of G, respectively. W.l.o.g., let $u \in R_r(v)$; by the planarity of G, $w \in R_r(v)$, as well. By Lemma 6.6, $R_r(u) \subset R_r(v)$ and $R_r(w) \subset R_r(v)$. Hence, $v_r > u_r$ and $v_r > w_r$. □

Let Γ be the planar straight-line drawing resulting from the barycentric representation of Lemma 6.8. By that lemma and by Lemma 6.7, Γ is a planar straight-line drawing of G on plane $b + g + r = 2n - 5$ in $\mathbb{Z}^3$. In particular, vertices s_b, s_g, and s_r are mapped to points $(2n-5, 0, 0)$, $(0, 2n-5, 0)$, and $(0, 0, 2n-5)$, respectively. A planar straight-line drawing of G on the $(2n-5) \times (2n-5)$ grid in $\mathbb{Z}^2$ can be obtained by projecting Γ, e.g., by dropping, for each vertex v, the red coordinate v_r, as illustrated in Figure 6.26.

As for the time and space complexity, by Lemma 6.4, a realizer of G can be constructed in linear time and space. The coordinates of the vertices of G can also be computed in linear time and space.

It is possible to obtain more compact drawings by relaxing the constraints imposed on the vertex coordinates by Definition 6.4. Given two ordered pairs (a, b) and (c, d), the $>_{lex}$ relation is defined by $(a, b) >_{lex} (c, d) \Leftrightarrow a > c \vee (a = c \wedge b > d)$.

DEFINITION 6.5 A *weak barycentric representation* of a graph $G = (V, E)$ is an injective function $f : v \in V \rightarrow (v_b, v_g, v_r) \in \mathbb{Z}^3$ that satisfies the following conditions:

1. For each vertex v of G, $v_b + v_g + v_r = c$, where c is a constant dependent on G.
2. For each edge (u, w) and each vertex $v \neq u, w$ of G, there exist two consecutive coordinates i and j in the circularly ordered set $\{b, g, r\}$ such that $(v_i, v_j) >_{lex} (u_i, u_j)$ and $(v_i, v_j) >_{lex} (w_i, w_j)$.

The following lemma can be proved similarly to Lemma 6.7 and implies that only a planar graph can have a weak barycentric representation.

LEMMA 6.9 **[Sch90]** A weak barycentric representation $f : v \in V \rightarrow (v_b, v_g, v_r)$ of a graph $G = (V, E)$ is a planar straight-line drawing of G on plane $b + g + r = c$ in $\mathbb{Z}^3$.

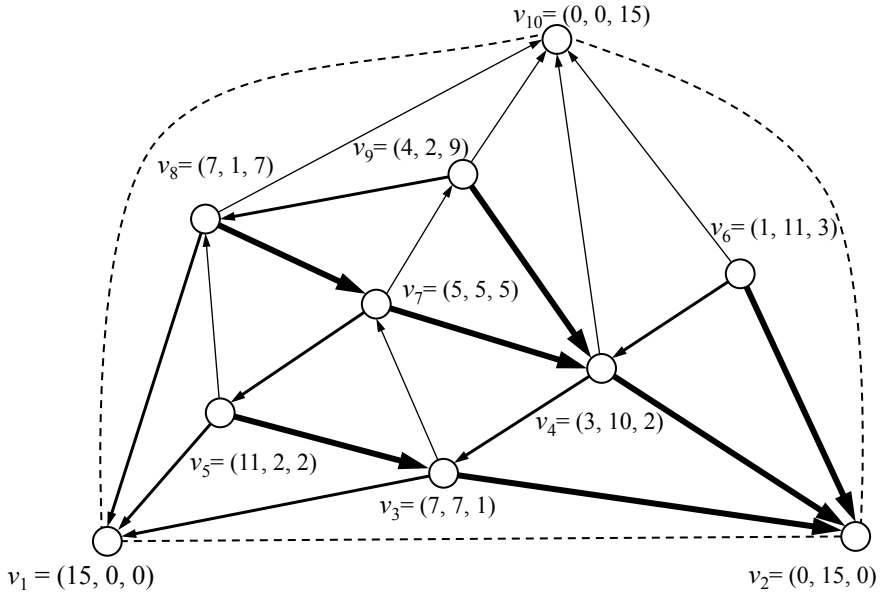


Figure 6.25 Barycentric coordinates obtained from a realizer.

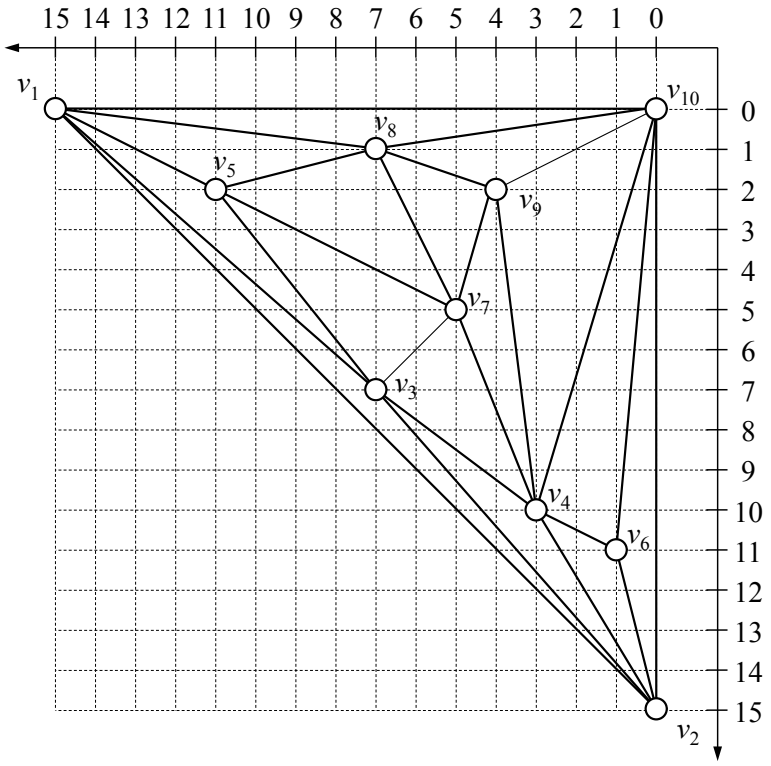


Figure 6.26 Planar straight-line grid drawing obtained from the barycentric coordinates of Figure 6.25 by dropping the third (red) coordinate. The horizontal and vertical axes are shown reversed to maintain the visual correspondence with the drawing of Figure 6.25.

For each vertex v of G, we denote by $n_b(v)$, $n_g(v)$, and $n_r(v)$ the number of vertices in $R_b(v) - p_r(v)$, $R_g(v) - p_b(v)$, and $R_r(v) - p_g(v)$, respectively. Note that $0 \leq n_b(v), n_g(v), n_r(v) \leq n-2$ and $n_b(v) + n_g(v) + n_r(v) = n-1$.

LEMMA 6.10 Let u and v be two distinct vertices of G, and let i and j be two consecutive coordinates in the circularly ordered set $\{b, g, r\}$. If $u \in R_i(v)$, then $(n_i(v), n_j(v)) >_{lex} (n_i(u), n_j(u))$.

Proof: W.l.o.g, we assume $i = r$ and thus $j = b$. Two cases are possible:

1. $u \notin p_g(v)$; by Lemma 6.6, $R_r(u) \subset R_r(v)$, and thus $p_g(u)$ is in $R_r(v)$; since $u \in p_g(u)$, we have $u \notin R_r(u) - p_g(u)$ while $u \in R_r(v) - p_g(v)$; thus, $R_r(u) - p_g(u) \subset R_r(v) - p_g(v)$; hence, $n_r(v) > n_r(u)$;
2. $u \in p_g(v)$; two subcases are possible:
   (a) $R_r(u) - p_g(u) \subset R_r(v) - p_g(v)$; hence, $n_r(v) > n_r(u)$;
   (b) $R_r(u) - p_g(u) = R_r(v) - p_g(v)$ (this subcase occurs if $par_b(u) = par_b(v)$); hence, $n_r(v) = n_r(u)$; however, $u \in R_b(v)$ and $u \notin p_r(v)$; by the same argument used for Case 1, $n_b(v) > n_b(u)$.

Thus, $(n_r(v), n_b(v)) >_{lex} (n_r(u), n_b(u))$. □

LEMMA 6.11 Let $G = (V, E)$ be a maximal planar graph equipped with a realizer. Function $f : v \in V \to (n_b(v), n_g(v), n_r(v))$ is a weak barycentric representation of G.

Proof: Injectivity of f follows from Lemma 6.10. Condition 1 of Definition 6.5 is trivially satisfied, since, for each vertex v, $v_b + v_g + v_r = n - 1$. As for Condition 2, let (u, w) and $v \neq u, w$ be an edge and a vertex of G, respectively. W.l.o.g., let $u \in R_r(v)$; by the planarity of G, $w \in R_r(v)$, as well. Hence, by Lemma 6.10, $(v_r, v_b) >_{lex} (u_r, u_b)$ and $(v_r, v_b) >_{lex} (w_r, w_b)$. □

6.7.3 Implementation

Let Γ be the straight-line drawing of G resulting from the weak barycentric representation of Lemma 6.11. By that lemma and by Lemma 6.9, Γ is a planar straight-line drawing of G on plane $b + g + r = n - 1$ in $\mathbb{Z}^3$. In particular, vertices s_b, s_g, and s_r are mapped to points $(n-2, 1, 0)$, $(0, n-2, 1)$, and $(1, 0, n-2)$, respectively. A planar straight-line drawing of G on the $(n-2) \times (n-2)$ grid in $\mathbb{Z}^2$ can be obtained by projecting Γ, e.g., by dropping, for each vertex v, the red coordinate v_r.

We now consider the time and space complexity. By Lemma 6.4, a realizer of G can be constructed in linear time and space. Next, we show that the coordinates for the vertices of G can be computed in linear time and space. In particular, we show how to compute, for each vertex v of G, coordinate v_r; coordinates v_b and v_g can be computed similarly.

From the planarity of G and Property 5 of the realizers, it follows that, for each vertex $u \neq v \in R_r(v)$, (i) the subtree $T_r(u)$ of T_r rooted at u is contained by $R_r(v)$, and (ii) $p_r(u)$ has exactly one vertex w in common with $p_b(v) \cup p_g(v)$ (note that $u \in T_r(w)$).

First, we compute, for each vertex v of G, the number of its descendants in T_r, including v itself, and store it in variable $numdesc_r(v)$; this can be done by a postorder visit of T_r.

Second, we compute, for each vertex v of G, the number of its ancestors in T_g, including v itself, and store it in variable $numanc_g(v)$; this can be done by a preorder visit of T_g. Finally, we compute, for each vertex v of G, $\sum_{w \in p_b(v)} numdesc_r(w)$ and $\sum_{w \in p_g(v)} numdesc_r(w)$; this can be done by a preorder visit of T_b and T_g, respectively.

For each vertex v of G, the number $n_r(v)$ of vertices in $R_r(v) - p_g(v)$, i.e., coordinate v_r, is given by the expression

$$\sum_{w \in p_b(v)} numdesc_r(w) + \sum_{w \in p_g(v)} numdesc_r(w) - numdesc_r(v) - numanc_g(v)$$

It follows that the coordinates for the vertices of G can be computed by a constant number of traversals of T_b, T_g, and T_r, and thus globally in $O(n)$ time. Furthermore, the additional variables used in the tree traversals clearly take $O(n)$ space.

Thus, we obtain the following theorem that summarizes the area bound and computational complexity of the realizer method.

Theorem 6.2 **[Sch90]** *Let G be a maximal plane graph with n vertices. The realizer method computes a planar straight-line drawing of G on the $(n-2) \times (n-2)$ grid in $O(n)$ time and space.*

6.7.4 Refinements and Variations

Zhang and He [ZH03] discovered some new properties of Schnyder's realizers and were able to further reduce the grid size (in most cases).

Di Battista, Tamassia, and Vismara [DTV99] extend the realizer method to construct in linear time a convex grid drawing of a triconnected plane graph on the $(f-1) \times (f-1)$ grid, where f is the number of faces of the graph. The same result had been claimed by Schnyder and Trotter [ST92] without proof and is independently obtained by Felsner [Fel01] with different techniques. A method that further improves the grid size was developed by Bonichon, Felsner, and Mosbah [BFM07].

Acknowledgment

Roberto Tamassia contributed to the writing of this chapter.

References

[BFM07] Nicolas Bonichon, Stefan Felsner, and Mohamed Mosbah. Convex drawings of 3-connected plane graphs. *Algorithmica*, 47:399–420, 2007.

[Bra08] Franz J. Brandenburg. Drawing planar graphs on $\frac{8}{9}n^2$ area. *Electronic Notes in Discrete Mathematics*, 31:37–40, 2008.

[CG95] I. F. Cruz and A. Garg. Drawing graphs by example efficiently: Trees and planar acyclic digraphs. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes Comput. Sci.*, pages 404–415. Springer-Verlag, 1995.

[CK97] M. Chrobak and G. Kant. Convex grid drawings of 3-connected planar graphs. *Internat. J. Comput. Geom. Appl.*, 7(3):211–223, 1997.

[CN98] Marek Chrobak and S. Nakano. Minimum-width grid drawings of plane graphs. *Comput. Geom. Theory Appl.*, 11:29–54, 1998.

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

[CP95] M. Chrobak and T. Payne. A linear-time algorithm for drawing planar graphs. *Inform. Process. Lett.*, 54:241–246, 1995.

[CYN84] N. Chiba, T. Yamanouchi, and T. Nishizeki. Linear algorithms for convex drawings of planar graphs. In J. A. Bondy and U. S. R. Murty, editors, *Progress in Graph Theory*, pages 153–173. Academic Press, New York, NY, 1984.

[DF13] Giuseppe Di Battista and Fabrizio Frati. Drawing trees, outerplanar graphs, series-parallel graphs, and planar graphs in a small area. In J. Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 121–166. 2013.

[dFPP90] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.

[Dji95] H. N. Djidjev. On drawing a graph convexly in the plane. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes Comput. Sci.*, pages 76–83. Springer-Verlag, 1995.

[DTV99] G. Di Battista, R. Tamassia, and L. Vismara. Output-sensitive reporting of disjoint paths. *Algorithmica*, 23(4):302–340, 1999.

[Fár48] I. Fáry. On straight lines representation of planar graphs. *Acta Univ. Szeged. Sect. Sci. Math.*, 11:229–233, 1948.

[Fel01] Stefan Felsner. Convex drawings of planar graphs and the order dimension of 3-polytopes. *Order*, 18:19–37, 2001.

[HT73] J. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973.

[Kan96] G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16:4–32, 1996. (special issue on Graph Drawing, edited by G. Di Battista and R. Tamassia).

[NR04] Takao Nishizeki and Md. Saidur Rahman. *Planar Graph Drawing*. World Scientific, 2004.

[RT86] P. Rosenstiehl and R. E. Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete Comput. Geom.*, 1(4):343–353, 1986.

[Sch89] W. Schnyder. Planar graphs and poset dimension. *Order*, 5:323–343, 1989.

[Sch90] W. Schnyder. Embedding planar graphs on the grid. In *Proc. 1st ACM-SIAM Sympos. Discrete Algorithms*, pages 138–148, 1990.

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

[ST92] W. Schnyder and W. T. Trotter. Convex embeddings of 3-connected plane graphs. *Abstracts of the AMS*, 13(5):502, 1992.

[Ste51] S. K. Stein. Convex maps. *Proc. Amer. Math. Soc.*, 2(3):464–466, 1951.

[Tho80] C. Thomassen. Planarity and duality of finite and infinite planar graphs. *J. Combin. Theory Ser. B*, 29(2):244–271, 1980.

[Tho84] C. Thomassen. Plane representations of graphs. In J. A. Bondy and U. S. R. Murty, editors, *Progress in Graph Theory*, pages 43–69. Academic Press, New York, NY, 1984.

[Tut60] W. T. Tutte. Convex representations of graphs. *Proceedings London Mathematical Society*, 10(38):304–320, 1960.

[Tut63] W. T. Tutte. How to draw a graph. *Proceedings London Mathematical Society*, 13(52):743–768, 1963.

[Wag36] K. Wagner. Bemerkungen zum Vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.

[ZH03] Huaming Zhang and Xin He. Compact visibility representation and straight-line grid embedding of plane graphs. In *Algorithms and Data Structures*, volume 2748 of *Lecture Notes in Computer Science*, pages 493–504. Springer, 2003.