Chapter 2

Plane Embeddings

Graphs can be represented in various ways, for instance, as an adjacency matrix or using adjacency lists. In this chapter we explore another class of representations that are quite different in nature, namely *geometric* representations. In a geometric representation, vertices and edges are represented by geometric objects, for example points and curves. This approach is appealing because it succinctly visualizes a graph along with its many properties. We have many degrees of freedom in selecting the geometric objects and the details of their geometry. This freedom allows us to tailor the representation to meet specific goals, such as emphasizing certain structural aspects of the graph at hand or reducing the complexity of the obtained representation.

The most common geometric graph representation is a *drawing*, where vertices are mapped to points and edges to curves in $\mathbb{R}^2$. It is desirable to make such a map injective by avoiding edge crossings, both from a mathematically aesthetic viewpoint and for the sake of the practical readability. Those graphs that allow such an *embedding* into the Euclidean plane are known as *planar*. Our goal is to study the interplay between abstract planar graphs and their plane embeddings. Specifically, we want to answer the following questions:

- What is the combinatorial complexity (that is, the number of edges and faces) of planar graphs?
- Under which conditions are plane embeddings unique (up to a certain sense of equivalence)?
- How can we represent plane embeddings in a data structure?
- What is the geometric complexity (that is, the encoding size of the geometric objects used to represent vertices and edges) of plane embeddings?

Most definitions we use directly extend to multigraphs. But for simplicity, we use the term “graph” throughout.

2.1 Drawings, Embeddings and Planarity

A **curve** is a set $C \subset \mathbb{R}^2$ of the form $\{\gamma(t) : 0 \leqslant t \leqslant 1\}$, where $\gamma : [0,1] \to \mathbb{R}^2$ is a continuous function. The function γ is called a *parameterization* of C. The points $\gamma(0)$ and $\gamma(1)$ are the *endpoints* of the curve. A curve is *closed* if $\gamma(0) = \gamma(1)$. A curve is *simple* if it admits a parameterization γ that is injective on $[0,1]$; for a closed simple curve we allow as an exception that $\gamma(0) = \gamma(1)$. The following famous theorem describes an important property of the plane. A proof can, for instance, be found in the book of Mohar and Thomassen [24].

Theorem 2.1 (Jordan). *Any simple closed curve C partitions the plane into exactly two regions (connected open sets), each bounded by C.*

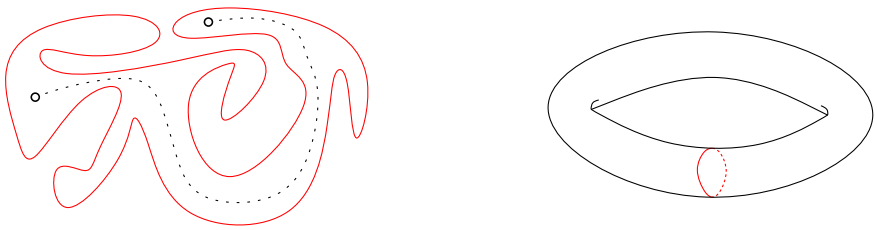


Figure 2.1: *Left: a simple closed curve in the plane and two points in one of its faces. Right: a simple closed curve that does not disconnect the torus.*

Observe that, for instance, on the torus there are simple closed curves that do not disconnect the surface, and thus the theorem does not hold there.

Drawings. As a first criterion for a reasonable geometric representation of a graph, we would like to have a clear separation between different vertices and also between a vertex and nonincident edges. Formally, a *drawing* of a graph $G = (V, E)$ in the plane is a function f that assigns

- a point $f(v) \in \mathbb{R}^2$ to every vertex $v \in V$ and
- a simple curve $f(uv)$ with endpoints $f(u)$ and $f(v)$ to every edge $uv \in E$,

such that

(1) f is injective on V and

(2) $f(uv) \cap f(V) = \{f(u), f(v)\}$, for every edge $uv \in E$.

A common point $f(e) \cap f(e')$ between two curves that represent distinct edges $e, e' \in E$ is called a *crossing* if it is not a common endpoint of e and e'.

Commonly, when discussing a drawing of a graph $G = (V, E)$, we do not differentiate a vertex/an edge from its geometric realization. That is, a vertex $v \in V$ is identified with the point $f(v)$, and an edge $e \in E$ is identified with the curve $f(e)$. For instance, the last

sentence in the previous paragraph may be phrased as "A common point of two edges is called a crossing if it is not their common endpoint."

Often it is convenient to make additional assumptions about edge intersections in a drawing. For example, we may demand *nondegeneracy* in the sense that no three edges can meet at a single crossing, or that any two edges can intersect at only finitely many points.

Planar vs. plane. A graph is *planar* if it admits a drawing in the plane without crossings. Such a drawing is also called a *crossing-free* drawing or a (plane) *embedding* of the graph. A planar graph together with a particular plane embedding is called a *plane* graph. Note the distinction between "planar" and "plane": the former refers to an abstract graph and indicates the possibility of an embedding, whereas the latter refers to a concrete embedding (Figure 2.2).

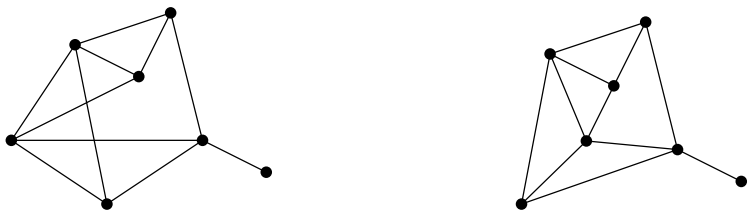


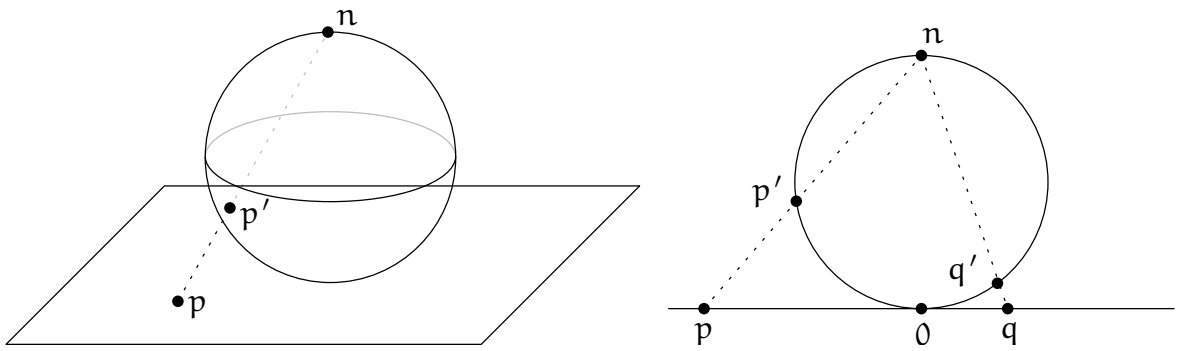
Figure 2.2: *A planar graph (left) and a plane embedding of it (right).*

A *geometric graph* is a graph together with a drawing in which all edges are straight-line segments. Note that such a drawing is fully determined by the vertex positions. A plane graph which is also geometric is called a *plane straight-line graph* (PSLG). On the other hand, a plane graph whose edges are arbitrary simple curves is emphasized as *topological plane graph*.

The *faces* of a plane graph G are the maximally connected regions of $\mathbb{R}^2 \setminus G$, that is, the plane without the points occupied by the embedding (as the image of a vertex or an edge). Each embedding of a finite graph has exactly one *unbounded face*, also called *outer* or *infinite* face. Using stereographic projection, we could show that any face can be swapped out to serve as the unbounded face:

Theorem 2.2. *If a graph G has a plane embedding in which some face is bounded by a cycle $(v_1, \dots, v_k)$, then G also has a plane embedding in which the unbounded face is bounded by the cycle $(v_1, \dots, v_k)$.*

Proof Sketch. Take a plane embedding Γ of G and map it to the sphere using *stereographic projection*: Imagine $\mathbb{R}^2$ being the x/y-plane in $\mathbb{R}^3$ and place a unit sphere S whose south pole touches the origin. We establish a bijection between $\mathbb{R}^2$ and $S \setminus \{n\}$, where $n := (0, 0, 2)$ is the north pole position: A point $p \in \mathbb{R}^2$ is mapped to the intersection p' of the segment $\overline{pn}$ and S, see Figure 2.3. The map is continuous, so it preserves incidence between vertices, edges and faces.

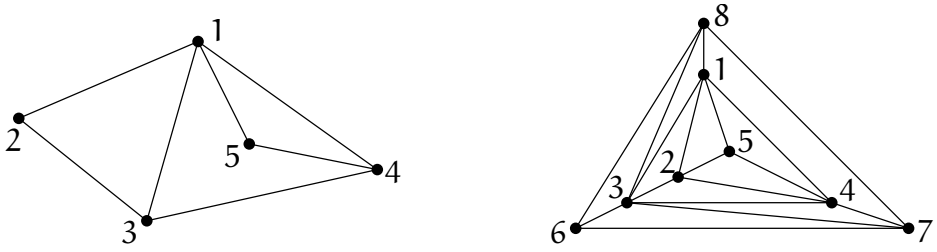


(a) Three-dimensional view. (b) Cross-section view.

Figure 2.3: *Stereographic projection.*

Consider the resulting embedding Γ' of G on S: The infinite face of Γ corresponds to the face of Γ' that contains the north pole n of S. Now rotate the embedding Γ' on S such that the desired face contains n. Mapping back to the plane using stereographic projection results in an embedding in which the desired face is the outer face. □

Exercise 2.3. *Consider the plane graphs depicted in Figure 2.4. For both graphs give a plane embedding in which the cycle* $(1, 2, 3)$ *bounds the outer face.*



(a) (b)

Figure 2.4: *Make* $(1, 2, 3)$ *bound the outer face.*

Duality. Every plane graph G has a *dual* G^* whose vertices are the faces of G. For every edge in G, we connect its two incident faces by an edge in the dual G^*. Note that in general, G^* is a multigraph (with loops and multiple edges) and may depend on the embedding. So an abstract planar graph G may have several nonisomorphic duals; see Figure 2.5 for an example. If G is a connected plane graph, then $(G^*)^* = G$. We will see later in Section 2.3 that the dual of a 3-connected planar graph is unique (up to isomorphism).

The Euler Formula and its ramifications. One of the most important tools for planar graphs (and more generally, graphs embedded on a surface) is the Euler–Poincaré Formula.

Figure 2.5: *Two plane drawings G_1 and G_2 of the same abstract planar graph and their duals G_1^* and G_2^* with $G_1^* \not\cong G_2^*$. (To see this, for instance, count the number of vertices of degree greater than three.)*

Theorem 2.4 (Euler's Formula). *For every connected plane graph with n vertices, e edges, and f faces, we have $n - e + f = 2$.*

Proof. Let G be a connected plane graph with n vertices, e edges, and f faces. Note that $e \geqslant n - 1$ as G is connected.

We prove the statement by induction on $e - n$. In the base case $e - n = -1$, the graph G is a (plane) tree and contains exactly one (unbounded) face, and so $n - e + f = 1 + 1 = 2$ as claimed.

In the general case, fix a spanning tree T of G, pick an arbitrary edge e of $G \setminus T$, and consider the graph $G^- = G \setminus e$. By construction it has n vertices and $e - 1$ edges. We claim that it has $f - 1$ faces. To see this observe that $G^- \supset T$ is connected. In particular, the endpoints of e are connected by a path in G^-, which together with e forms a cycle in G. So in G, any two points sufficiently close to but on opposite sides of e are in different faces, whereas they are in the same face of G^-. In other words, the two incident faces of e are distinct in G but merged into one in G^-. All other faces remain untouched. It follows that G^- has $f - 1$ faces, as claimed. Then by the inductive assumption on G^-, we have $n - e + f = n - (e - 1) + (f - 1) = 2$, which concludes the induction. □

In particular, this shows that every plane embedding of a planar graph has the same number of faces. In other words, the number of faces is an invariant of an abstract planar graph. It also follows (as the corollary below) that planar graphs are *sparse*, that is, they have a linear number of edges and faces only. So the asymptotic complexity of a planar graph is already determined by its number of vertices.

Corollary 2.5. *A simple planar graph on $n \geqslant 3$ vertices has at most $3n - 6$ edges and at most $2n - 4$ faces.*

Proof. Without loss of generality we may assume that G is connected. (If not, add edges between components of G until the graph is connected. The number of edges increases and the number of faces remains unchanged.) The statement is easily checked for $n = 3$, where G is either a triangle or a path and therefore has no more than $3 \leqslant 3 \cdot 3 - 6$ edges and no more than $2 \leqslant 2 \cdot 3 - 4$ faces. Next consider a simple connected planar graph G

on $n \geqslant 4$ vertices, and fix any plane embedding of it. Denote by E its set of edges and by F its set of faces. Let

$$X = \{(e, f) \in E \times F : e \text{ bounds } f\}$$

denote the set of incident edge-face pairs. We count X in two different ways.

First note that each edge bounds at most two faces and so $|X| \leqslant 2 \cdot |E|$.

Second note that every face is bounded by at least three edges: If G contains a cycle, then the boundary of every face shall contain a cycle and hence at least three edges. If G is acyclic, then it must be a tree since we assumed it to be connected. Its only face (the outer face) is bounded by all edges; and there are at least three since G contains at least four vertices. In both cases we have $|X| \geqslant 3 \cdot |F|$.

Therefore $3|F| \leqslant 2|E|$. Using Euler's Formula we conclude that

$$\begin{aligned} 4 &= 2(n - |E| + |F|) \leqslant 2n - 3|F| + 2|F| = 2n - |F| \quad \text{and} \\ 6 &= 3(n - |E| + |F|) \leqslant 3n - 3|E| + 2|E| = 3n - |E| \,, \end{aligned}$$

which yield the claimed bounds. □

Corollary 2.5 implies that the degree of a "typical" vertex in a planar graph is a small constant.

Corollary 2.6. *The average vertex degree in a simple planar graph is less than six.*

Exercise 2.7. *Prove Corollary 2.6.*

There exist several variations of this statement, a few more of which we will encounter during this course.

Exercise 2.8. *Show that neither K_5 (the complete graph on five vertices) nor $K_{3,3}$ (the complete bipartite graph where both classes have three vertices) is planar.*

Exercise 2.9. *Let P be a set of $n \geqslant 3$ points in the plane such that the distance between every pair of points is at least one. Show that there are at most $3n - 6$ pairs of points in P at distance exactly one.*

Characterizing planarity. The classical theorems of Kuratowski and Wagner provide a characterization of planar graphs in terms of forbidden substructures. A *subdivision* of a graph $G = (V, E)$ is obtained from G by replacing each edge with a path.

Theorem 2.10 (Kuratowski [22, 31]). *A graph is planar if and only if it does not contain a subdivision of $K_{3,3}$ or K_5.*

A *minor* of a graph $G = (V, E)$ is obtained from G using zero or more edge contractions, edge deletions, and/or vertex deletions.

Theorem 2.11 (Wagner [34]). *A graph is planar if and only if it does not contain $K_{3,3}$ or K_5 as a minor.*

In some sense, Wagner's Theorem is a special instance[1] of a much more general theorem.

Theorem 2.12 (Graph Minor Theorem, Robertson/Seymour [28]). *Every minor-closed family of graphs can be described in terms of a finite set of forbidden minors.*

Being *minor-closed* means that any minor of any graph from the family also belongs to the family. For instance, the family of planar graphs is minor-closed because planarity is preserved under removal of edges and vertices and under edge contractions.

Exercise 2.13. *A graph is 1-planar if it admits a drawing in the plane in which every edge has at most one crossing. Prove or disprove: The family of 1-planar graphs is minor-closed.*

The Graph Minor Theorem is a celebrated result established by Robertson and Seymour in a series of twenty papers, see also the survey by Lovász [23]. They also describe an $O(n^3)$ algorithm (with horrendous constants, though) to decide whether a graph on n vertices contains a fixed (constant-size) minor. As a consequence, every minor-closed property can be tested in polynomial time. Later, Kawarabayashi et al. [20] showed that this problem can be solved in $O(n^2)$ time.

Unfortunately, the Graph Minor Theorem is nonconstructive in the sense that in general we do not know how to obtain the set of forbidden minors for a given family. For instance, for the family of toroidal graphs (graphs that can be embedded without crossings on the torus) more than 16′000 forbidden minors are known, and the theorem tells us that the number is finite, but we still do not know the concrete number. So while we know that there exists a quadratic time algorithm to test membership for minor-closed families, we have no idea what such an algorithm looks like in general.

Graph families other than planar graphs for which the forbidden minors are known include forests (free of K_3 minors) and outerplanar graphs (free of $K_{2,3}$ and K_4 minors). A graph is *outerplanar* if it admits a plane embedding in which all vertices appear on the outer face (Figure 2.6).

Figure 2.6: *An outerplanar graph (left) and a plane embedding of it in which all vertices are incident to the outer face (right).*

Exercise 2.14. *(a) Give an example of a 6-connected planar graph or argue that no such graph exists.*

[1] It is more than just a special instance because it also specifies the forbidden minors explicitly.

(b) Give an example of a 5-connected planar graph or argue that no such graph exists.

(c) Give an example of a 3-connected outerplanar graph or argue that no such graph exists.

Planarity testing. To test a given graph for planarity we do not have to contend ourselves with a quadratic-time algorithm. In fact, there exist a number of different linear time algorithms that decide if a given abstract graph is planar; all of them—from a very high-level point of view—can be regarded as an annotated depth-first-search. The first such algorithm was described by Hopcroft and Tarjan [19], while the current state-of-the-art is probably among the "path searching" method by Boyer and Myrwold [6] and the "LR-partition" method by de Fraysseix et al. [14]. Although the overall idea in all these approaches is easy to convey, many technical details make an in-depth discussion rather painful to go through.

2.2 Graph Representations

There are two standard representations for an abstract graph $G = (V, E)$ on $n = |V|$ vertices. For the *adjacency matrix* representation we consider the vertices to be ordered as $V = \{v_1, \ldots, v_n\}$. The adjacency matrix of an undirected graph is a symmetric $n \times n$-matrix $A = (a_{ij})_{1 \leqslant i,j \leqslant n}$ where $a_{ij} = a_{ji} = 1$, if $\{v_i, v_j\} \in E$, and $a_{ij} = a_{ji} = 0$ otherwise. Storing such a matrix explicitly requires $\Omega(n^2)$ space, but it allows testing in constant time whether or not two given vertices are adjacent.

In an *adjacency list* representation, we store for each vertex a list of its neighbors in G. This requires only $O(n+|E|)$ storage, which is better than for the adjacency matrix in case that $|E| = o(n^2)$. On the other hand, the adjacency test for two given vertices is not a constant-time operation, because it requires a search in one of the lists. Depending on the implementation of the lists, the search time ranges from $O(d)$ (for an unsorted list) to $O(\log d)$ (for a sorted dynamic data structure such as a balanced search tree), where d is the minimum degree of the two vertices.

Both representations have their merits. The choice typically depends on what one wants to do with the graph. When dealing with embedded graphs, however, additional information about the embedding is needed beyond the pure incidence structure of the graph. The next section discusses a standard data structure to represent embedded graphs.

2.2.1 The Doubly-Connected Edge List

The *doubly-connected edge list* (DCEL) is a data structure to represent a plane graph in such a way that it is easy to traverse and to manipulate. To avoid complications, let us discuss only connected graphs that contain at least two vertices. It is not hard to extend the data structure to be able to represent all plane graphs. We also assume

that we deal with a straight-line embedding and thus the geometry of edges is defined by the positions of their endpoints already. For more general embeddings, the geometric description of edges has to be stored in addition.

The main building block of a DCEL is a list of *halfedges*. Every actual edge is split into two halfedges going in opposite direction, and these are called *twins*, see Figure 2.7. Along the boundary of each face, halfedges are oriented counterclockwise, that is, the face always stays to the left.

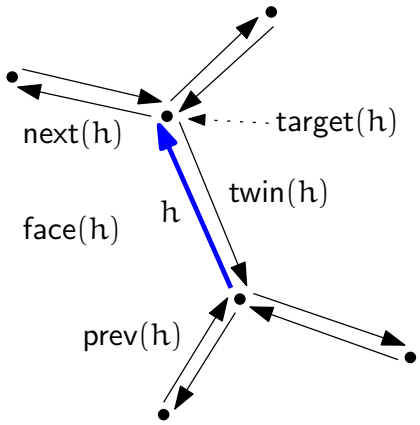


Figure 2.7: *A halfedge in a DCEL.*

A DCEL also stores a list of vertices and a list of faces. These three lists are unordered but interconnected by various pointers. A vertex v stores a pointer $\text{halfedge}(v)$ to an arbitrary halfedge originating from v. Every vertex also records its coordinates $\text{point}(v)$, that is, the point it is mapped to in the embedding. A face f stores a pointer $\text{halfedge}(f)$ to an arbitrary halfedge within the face. A halfedge h stores *five* pointers:

- a pointer $\text{target}(h)$ to its target vertex,
- a pointer $\text{face}(h)$ to its incident face,
- a pointer $\text{twin}(h)$ to its twin halfedge,
- a pointer $\text{next}(h)$ to the halfedge following h along the boundary of $\text{face}(h)$, and
- a pointer $\text{prev}(h)$ to the halfedge preceding h along the boundary of $\text{face}(h)$.

A constant amount of information is stored for every vertex, (half-)edge, and face of the graph. Therefore the whole DCEL needs storage proportional to $|V| + |E| + |F|$, which is $O(n)$ for a plane graph with n vertices by Corollary 2.5.

This information is sufficient for most tasks. For example, traversing all edges around a face f can be done as follows:

$s \leftarrow \text{halfedge}(f)$
$h \leftarrow s$
do

something with h
$h \leftarrow \mathsf{next}(h)$
while $h \neq s$

Exercise 2.15. *Give pseudocode to traverse all edges incident to a given vertex v of a DCEL.*

Exercise 2.16. *Why is the previous halfedge* $\mathsf{prev}(\cdot)$ *stored explicitly whereas the source vertex of a halfedge is not?*

2.2.2 Manipulating a DCEL

In many applications, plane graphs do not just appear as static objects but rather evolve over the course of an algorithm. Therefore the data structure must allow for efficient updates. These include, but are not limited to, appending new vertices, edges and faces to the corresponding list within the DCEL and—symmetrically—the ability to delete an existing entity.

First, it should be easy to add a new vertex v to the graph within a given face f and (as we maintain a connected graph) connect v to an existing vertex u. For such a connection to be valid, we require that the open line segment $\overline{uv}$ lies completely in f. Given that we need access to both f and u, it would be convenient to pass the already existing halfedge h that satisfies $\mathsf{face}(h) = f$ and $\mathsf{target}(h) = u$ as an argument. Assuming that $\mathsf{point}(v)$ has already been set to the desired location of the new vertex, our operation then becomes

$\mathsf{add\text{-}vertex\text{-}at}(v, h)$
Precondition: the open line segment $\overline{\mathsf{point}(v)\mathsf{point}(u)}$, where $u := \mathsf{target}(h)$, lies completely in $f := \mathsf{face}(h)$.
Postcondition: the new vertex v has been inserted into f, connected by an edge to u.

(a) before **(b)** after

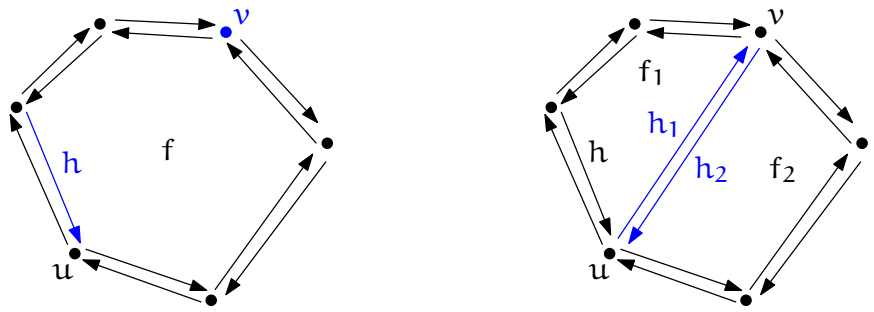
Figure 2.8: *Add a new vertex connected to an existing vertex u.*

See also Figure 2.8. It can be realized by manipulating a constant number of pointers as follows.

```
add-vertex-at(v, h) {
    h1 ← a new halfedge
    h2 ← a new halfedge
    halfedge(v) ← h2
    twin(h1) ← h2
    twin(h2) ← h1
    target(h1) ← v
    target(h2) ← u
    face(h1) ← f
    face(h2) ← f
    next(h1) ← h2
    next(h2) ← next(h)
    prev(h1) ← h
    prev(h2) ← h1
    next(h) ← h1
    prev(next(h2)) ← h2
}
```

Similarly, it should be possible to add an edge between two existing vertices u and v, provided the open line segment $\overline{uv}$ lies completely within a face f of the graph, see Figure 2.9. Since such an edge insertion splits f into two faces, the operation is called *split-face*. Again we pass as an argument the halfedge h satisfying $\text{face}(h) = f$ and $\text{target}(h) = u$.

split-face(h, v)
Precondition: v is incident to $f := \text{face}(h)$ but not adjacent to $u := \text{target}(h)$.
The open line segment $\overline{\text{point}(v)\text{point}(u)}$ lies completely in f.
Postcondition: f has been split by a new edge uv.



(a) before **(b)** after

Figure 2.9: *Split a face by an edge* uv.

The implementation is slightly more complicated compared to add-vertex-at above, because the face f is destroyed and so we have to update the face information of all incident

halfedges. In particular, this is not a constant time operation and has complexity proportional to the size of f.

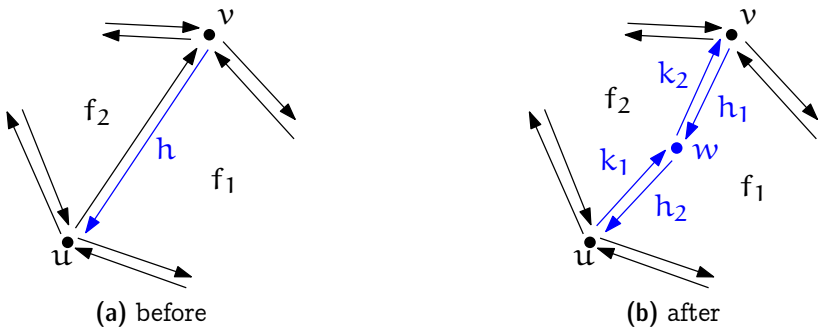
```
split-face(h, v) {
    f1 ← a new face
    f2 ← a new face
    h1 ← a new halfedge
    h2 ← a new halfedge
    halfedge(f1) ← h1
    halfedge(f2) ← h2
    twin(h1) ← h2
    twin(h2) ← h1
    target(h1) ← v
    target(h2) ← u
    next(h2) ← next(h)
    prev(next(h2)) ← h2
    prev(h1) ← h
    next(h) ← h1
    i ← h2
    loop
        face(i) ← f2
        if target(i) = v break the loop
        i ← next(i)
    endloop
    next(h1) ← next(i)
    prev(next(h1)) ← h1
    next(i) ← h2
    prev(h2) ← i
    i ← h1
    do
        face(i) ← f1
        i ← next(i)
    until target(i) = u
    delete the face f
}
```

In a similar fashion one can realize the inverse operation join-face(h) that removes the edge represented by h, thereby joining the faces face(h) and face(twin(h)).

It is easy to see that every connected plane graph on at least two vertices can be constructed using the operations add-vertex-at and split-face, starting from an embedding of K_2 (two vertices connected by an edge).

Exercise 2.17. *Give pseudocode for the operation join-face(h). Specify preconditions if needed.*

Exercise 2.18. *Give pseudocode for the operation split-edge*(h)*, that splits the edge represented by* h *into two by a new vertex* w*, see Figure 2.10.*



(a) before **(b)** after

Figure 2.10: *Split an edge by a new vertex.*

2.2.3 Graphs with Unbounded Edges

In some cases it is convenient to consider plane graphs in which some edges are not mapped to a line segment but to an unbounded curve, such as a ray. This setting is not really much different from the one we studied before, except that one special vertex is placed "at infinity". One way to think of it is in terms of *stereographic projection* (see the proof of Theorem 2.2). The further away a point in $\mathbb{R}^2$ is from the origin, the closer its image on the sphere S gets to the north pole n of S. But there is no way to reach n except in the limit. Therefore, we can imagine drawing the graph on S instead of in $\mathbb{R}^2$ and putting the "infinite vertex" at n.

All this is just for the sake of a proper geometric interpretation. As far as a DCEL of such a graph is concerned, there is no need to consider spheres or anything beyond what we have discussed. The only difference to the case with all finite edges is that there is this special infinite vertex, which does not have any point/coordinates associated to it. Other than that, the infinite vertex is treated in exactly the same way as the finite vertices: it has in- and out-going halfedges along which the unbounded faces can be traversed (Figure 2.11).

Remarks. It is actually not so easy to point exactly to where the DCEL data structure originates from. Often Muller and Preparata [25] are credited, but while they use the term DCEL, the data structure they describe is different from what we discussed above and from what people usually consider a DCEL nowadays. Overall, there are a large number of variants of this data structure, which appear under the names *winged edge* data structure [3], *halfedge* data structure [35], or *quad-edge* data structure [16]. Kettner [21] provides a comparison of all these with some additional references.

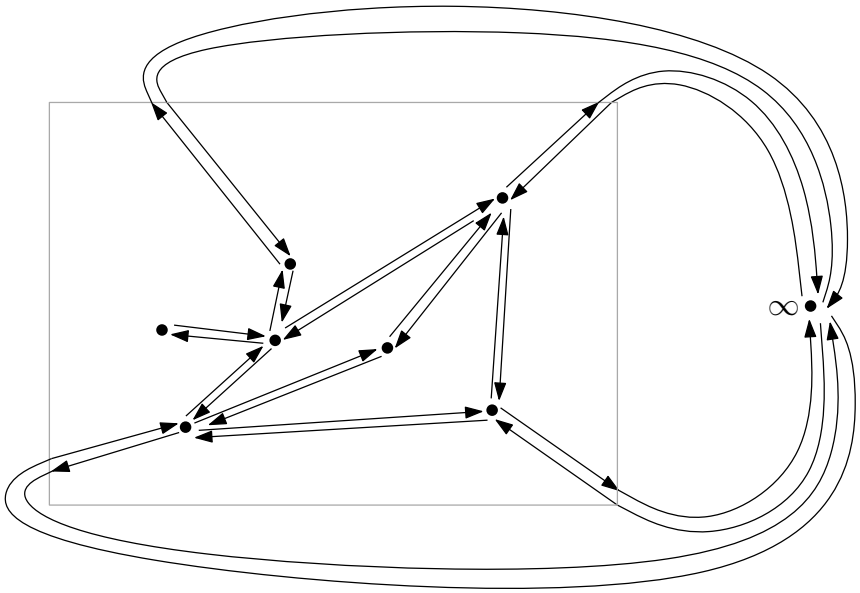


Figure 2.11: *A DCEL with unbounded edges. Usually, we will not show the infinite vertex and draw all edges as straight-line segments. This yields a geometric drawing, like the one within the gray box.*

2.2.4 Combinatorial Embeddings

The basic DCEL omits geometric aspects, such as coordinates of a vertex or the shape of an edge, and only stores incidences and adjacencies between vertices, edges, and faces of an embedding. We call such information the *combinatorial embedding* of the actual plane graph. Conventionally, we write it as a set of face boundaries, where each boundary is encoded as a circular sequence of vertices in counterclockwise order. For instance, the combinatorial embeddings of the plane graphs in Figure 2.12a are

$$\begin{aligned}
&(a) \;:\; \{(1,2,3),(1,3,6,4,5,4),(1,4,6,3,2)\},\\
&(b) \;:\; \{(1,2,3,6,4,5,4),(1,3,2),(1,4,6,3)\}, \text{and}\\
&(c) \;:\; \{(1,4,5,4,6,3),(1,3,2),(1,2,3,6,4)\}.
\end{aligned}$$

Note that a vertex can appear several times along the boundary of a face (if it is a cut-vertex).

This view allows us to compare embeddings easily. Two embeddings (plane graphs) are *combinatorially equivalent* if their combinatorial embeddings are equal up to a global change of orientation (reversing the order of all sequences simultaneously). For example, (b) is not equivalent to (a) nor (c), because it is the only one with a face bounded by seven vertices. However, (a) and (c) turn out to be equivalent: after reverting orientations f_1 takes the role of h_2, f_2 takes the role of h_1, and f_3 takes the role of h_3.

Exercise 2.19. *Let G be a planar graph with vertex set $\{1,\dots,9\}$. Try to find an embedding corresponding to the following list of circular sequences of faces:*

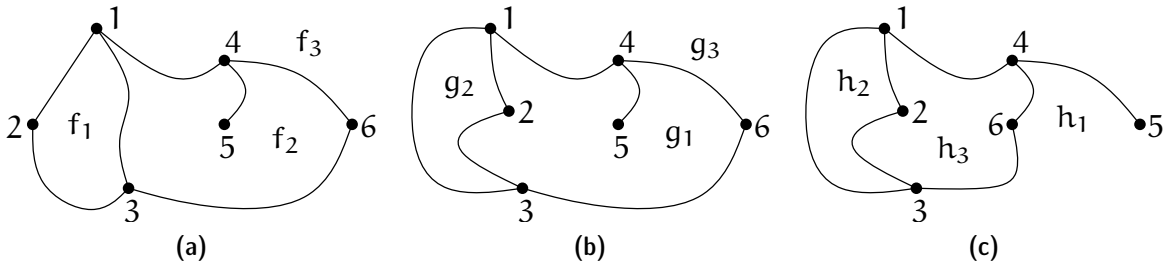


Figure 2.12: *Equivalent embeddings?*

(a) $\{(1,4,5,6,3),(1,3,6,2),(1,2,6,7,8,9,7,6,5),(7,9,8),(1,5,4)\}$

(b) $\{(1,4,5,6,3),(1,3,6,2),(1,2,6,7,8,9,7,6,5),(7,9,8),(1,4,5)\}$

Combinatorial embeddings are not only used to categorize plane graphs. They also play a role in algorithm design. Quite often, algorithms dealing with planar graphs do not need a full-fledged embedding to proceed. It is sufficient to operate on a combinatorial embedding, which is more efficient to handle.

Many people prefer a dual representation which, instead of listing face boundaries, enumerates the neighbors of v in cyclic order for each vertex v. It can avoid the issue of a vertex appearing multiple times in the sequence. However, the following lemma shows that such an issue does not arise when dealing with biconnected graphs.

Lemma 2.20. *In a biconnected plane graph every face is bounded by a cycle.*

We leave the proof as an exercise. Intuitively the statement is clear, but we believe it is instructive to think about a formal argument. An easy consequence is stated below, whose proof is also an exercise.

Corollary 2.21. *For any vertex v in a 3-connected plane graph, there is a cycle that contains all neighbors of v.*

Exercise 2.22. *Prove Lemma 2.20 and Corollary 2.21.*

Given Lemma 2.20, one might wonder the converse question: Which cycles in a planar graph G bound a face (in some plane embedding of G)? Such cycles are said to be *facial*; see Figure 2.13.

Exercise 2.23. *Describe a linear time algorithm that, given an abstract planar graph G and a cycle C in G, tests whether C is a facial cycle. (You may assume that planarity can be tested in linear time.)*

Exercise 2.24. *Let $G = (V, E)$ be a biconnected graph and let $\mathcal{C}$ be a collection of directed cycles of G. Assume that for every directed edge (u, v) whose underlying undirected edge $\{u, v\}$ is in E, there is exactly one cycle $C \in \mathcal{C}$ that contains (u, v). Further assume that for every vertex $v \in V$, the $\deg(v)$ cycles in $\mathcal{C}$ that contain v*

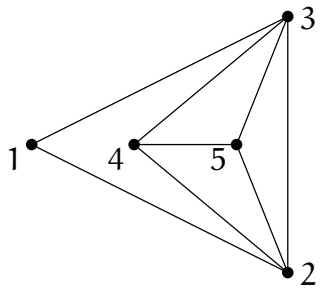


Figure 2.13: *The cycles* $(2,3,5)$ *and* $(1,2,5,3)$*, for example, are both facial. One can show that* $(2,4,3,5)$ *is not.*

can be ordered as $C_0, \dots, C_{\deg(v)-1}$ *such that for every pair* $u, v \in V$ *and* $i \in \mathbb{N}$ *with* $(u,v),(v,w) \in C_i$ *there exists an* $x \in V$ *such that* $(w,v),(v,x) \in C_{i+1}$ *(where indices are taken modulo* $\deg(v)$*). Prove or disprove: The collection* $\mathcal{C}$ *describes a combinatorial plane embedding of* G*.*

2.3 Unique Embeddings

As we have seen, an abstract planar graph may admit many different embeddings, even in the combinatorial sense. Under what condition does it admit a unique combinatorial embedding?

To answer the question, we start by studying cycles that bound a face in *every* plane embedding of G. (Note that this is stronger than being facial.) The lemma below provides a complete characterization of these cycles. Let us agree on some terminology about a cycle C in a graph G. A *chord* of C is an edge in $E(G) \setminus E(C)$ that connects two vertices of C. The cycle C is *induced* if it does not have any chord. It is *separating* if $G \setminus C$ is not connected.

Lemma 2.25. *Let* G *be a planar graph which is neither a cycle, nor a cycle plus a single chord. Then a cycle* C *in* G *bounds a face in every plane embedding of* G *if and only if* C *is induced and not separating.*

Proof. "$\Leftarrow$": Consider any plane embedding Γ of G. By the Jordan Curve Theorem, the cycle C splits the plane into an interior and an exterior region. As $G \setminus C$ is connected, it lies either entirely in the interior or entirely in the exterior. In either case, the other region is bounded by C because C does not have any chord.

"$\Rightarrow$": Using contraposition, suppose that (1) C is not induced or (2) C is separating. We aim to find a plane embedding of G in which C does not bound a face. To this end, let us start from an arbitrary plane embedding Γ of G. If C does not bound a face in Γ then we are done. So next we assume that C bounds a face in Γ.

(1) If C is not induced, then it has a chord c. As $G \neq C \cup c$, the graph G either has some vertex $v \notin C$ or another chord $d \neq c$ of C. We modify Γ by rerouting the

chord c inside the face C and obtain an embedding in which C does not bound a face: one of the two regions split by the Jordan curve C contains the chord c, and the other contains either the vertex v or another chord d.

(2) If C is separating, then $G \setminus C$ is not connected. If $G \setminus C = \emptyset$ then G is either C (which is excluded by assumption) or C plus some chords (which is handled by Case (1)). So from now on we assume $G \setminus C \neq \emptyset$ has two components A and B; see Figure 2.14a. Γ induces plane embeddings Γ_A of $A \cup C$ and Γ_B of $B \cup C$; the cycle C bounds a face in both of them. By the transformation in Theorem 2.2 we can make C bounding the outer face in Γ_A yet an inner face in Γ_B. Then we can glue the two embeddings at C, that is, extend Γ_B by adding Γ_A within the (inner) face bounded by C (Figure 2.14b). The result is a plane embedding of G in which C does not bound a face.

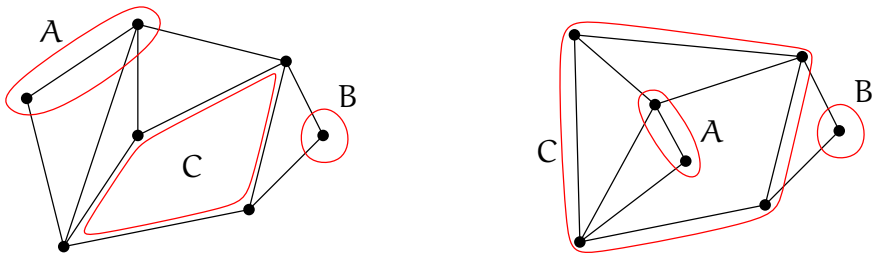


Figure 2.14: *A plane embedding in which C does not bound a face, in Case (2).*

□

For those special graphs G excluded in Lemma 2.25, it is easy to see that all cycles in G bound a face in every plane embedding. This completes the characterization. Since these special graphs are not 3-connected, we have

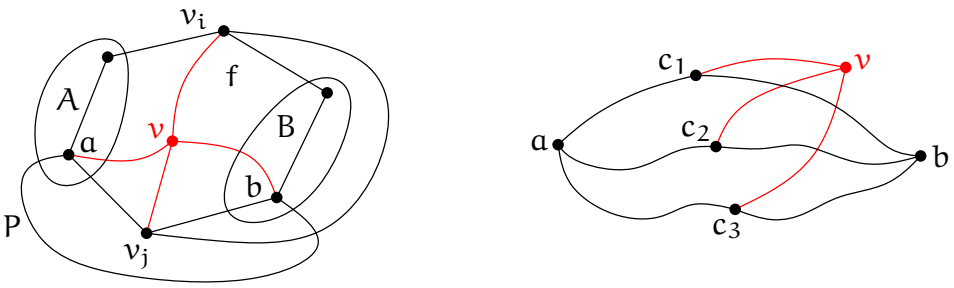
Corollary 2.26. *A cycle C of a 3-connected planar graph G bounds a face in every plane embedding of G if and only if C is induced and not separating.* □

The following theorem tells us that a wide range of graphs have little choice when embedded into the plane, from a combinatorial point of view. Geometrically, though, there is still much freedom.

Theorem 2.27 (Whitney [36]). *A 3-connected planar graph has a unique combinatorial plane embedding (up to equivalence).*

Proof. Let G be a 3-connected planar graph and suppose there exist two embeddings Φ_1 and Φ_2 of G that are not equivalent. So there is a cycle $C = (v_1, \dots, v_k)$ in G that, say, bounds a face f in Φ_1 but does not bound any face in Φ_2. By Corollary 2.26 there are only two options:

Case 1: C has a chord $\{v_i, v_j\}$. Denote $A = \{v_x : i < x < j\}$ and $B = \{v_x : x < i \vee j < x\}$ and observe that both A and B are nonempty because $\{v_i, v_j\}$ is a chord and so v_i and v_j are not adjacent in C. Given that G is 3-connected, there is at least one path P from A to B that avoids both v_i and v_j. Let a denote the last vertex of P that is in A, and let b denote the first vertex of P that is in B. As C bounds f in Φ_1, we can add a new vertex v inside f and connect it to each of v_i, v_j, a and b by four pairwise internally disjoint curves. The result would be a plane graph that contains a K_5 subdivision with branch vertices v, v_i, v_j, a, and b. This contradicts Kuratowski's Theorem (Theorem 2.10).



(a) Case 1. **(b)** Case 2.

Figure 2.15: *Illustration of the two cases in Theorem 2.27.*

Case 2: C is induced and separating. Since C is induced and G is 3-connected, we must have $G \setminus C \neq \emptyset$. So $G \setminus C$ contains two distinct components A and B. Choose vertices $a \in A$ and $b \in B$ arbitrarily. Applying Menger's Theorem (Theorem 1.5) on the 3-connected graph G, there exist three paths $\alpha_1, \alpha_2, \alpha_3$, pairwise internally vertex-disjoint, from a to b. Let c_i be some vertex where α_i intersects C, for $1 \leqslant i \leqslant 3$. Note that c_1, c_2, c_3 exist because C separates A and B, and they are pairwise distinct because $\alpha_1, \alpha_2, \alpha_3$ are pairwise internally (vertex-)disjoint. Therefore, $\{a, b\}$ and $\{c_1, c_2, c_3\}$ form branch vertices of a $K_{2,3}$ subdivision in G. We can add a new vertex v inside f and connect it to each of c_1, c_2 and c_3 by three pairwise internally disjoint curves. The result would be a plane graph that contains a $K_{3,3}$ subdivision. This contradicts Kuratowski's Theorem (Theorem 2.10).

In both cases we arrived at a contradiction and so there does not exist such a cycle C. Thus Φ_1 and Φ_2 are equivalent. □

Whitney's Theorem does not provide a characterization of unique embeddability in general, as there are biconnected graphs with unique combinatorial plane embedding (such as cycles) as well as those with several, non-equivalent combinatorial plane embeddings (such as a triangulated pentagon).

Exercise 2.28. *Describe a family of biconnected planar graphs with exponentially many combinatorial plane embeddings. That is, show that there exists a constant* $c \in \mathbb{R}$

such that for every $n \in \mathbb{N}$ *there exists a biconnected planar graph on* n *vertices that has at least* c^n *different combinatorial plane embeddings.*

2.4 Triangulating a Planar Graph

We like to study worst case scenarios not so much to dwell on "how bad things could get" but rather—phrased positively—because worst case examples provide universal bounds of the form "things are always at least this good". Most questions related to embeddings get harder when the graph contains more edges because every additional edge poses an increasing danger of crossing. So let us study the worst case: planar graphs such that adding any edge shall break its planarity. These graphs are called *maximal planar*. Corollary 2.5 tells us that every (hence also maximal) planar graph on n vertices has at most $3n - 6$ edges. Yet we would like to learn a bit more about how these graphs look like.

Lemma 2.29. *A maximal planar graph on* $n \geqslant 3$ *vertices is biconnected.*

Proof. Consider a maximal planar graph $G = (V, E)$. Note that G is connected because adding an edge between two distinct components of a planar graph maintains planarity. Now if G is not biconnected, then it has a cut-vertex v. Take a plane drawing Γ of G. As $G \setminus v$ is disconnected, removal of v also splits $N_G(v)$ into at least two components. Hence there are two vertices $a, b \in N_G(v)$, consecutive in the circular order around v in Γ, that are in different components of $G \setminus v$. In particular, $ab \notin E$ and we can add this edge to G (routing it very close to the path (a, v, b) in Γ) without violating planarity. This is in contradiction to G being maximal planar, so G must be biconnected. □

Lemma 2.30. *In any embedding of a maximal planar graph on* $n \geqslant 3$ *vertices, all faces are topological triangles, that is, every face is bounded by exactly three edges.*

Proof. Consider a maximal planar graph $G = (V, E)$ and a plane drawing Γ of G. By Lemma 2.29 we know that G is biconnected and so by Lemma 2.20 every face of Γ is bounded by a cycle. Suppose that there is a face f in Γ bounded by a cycle $(v_0, \ldots, v_{k-1}, v_k = v_0)$ of $k \geqslant 4$ vertices. We claim that at least one of the edges v_0v_2 or v_1v_3 is not in E.

Suppose to the contrary that $\{v_0v_2, v_1v_3\} \subseteq E$. Then we can add a new vertex v' in the interior of f and connect it to each of v_0, v_1, v_2, v_3 by a curve inside f without introducing a crossing. In other words, given G is planar, the graph $G' = (V \cup \{v'\}, E \cup \{v'v_i : i \in \{0, 1, 2, 3\}\})$ is also planar. However, v_0, v_1, v_2, v_3, v' are branch vertices of a K_5 subdivision in G': v' is connected to all other vertices within f, each vertex v_i is connected to both $v_{(i-1) \bmod 4}$ and $v_{(i+1) \bmod 4}$ along the boundary of f, and the two missing connections are provided by the edges v_0v_2 and v_1v_3 (Figure 2.16a). This contradicts Kuratowski's Theorem. Therefore, one of the edges v_0v_2 or v_1v_3 must be absent from E, as claimed.

So assume without loss of generality that $v_1v_3 \notin E$. But then we can route a curve from v_1 to v_3 inside f in Γ without introducing a crossing (Figure 2.16b). It follows that

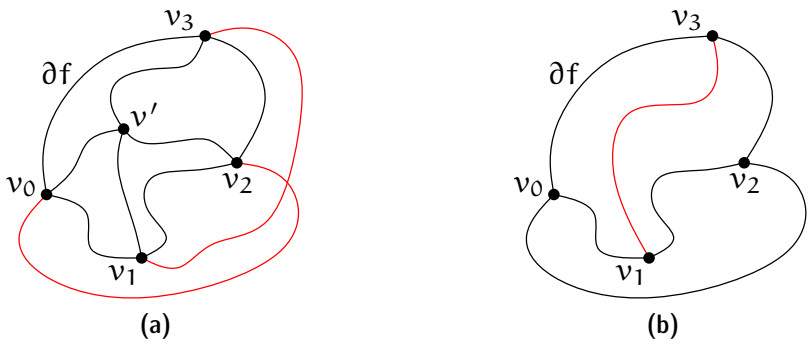


Figure 2.16: *Every face of a maximal planar graph is a topological triangle.*

the edge v_1v_3 can be added to G without sacrificing planarity, which is in contradiction to G being maximal planar. Therefore, there is no such face f bounded by four or more vertices. □

Theorem 2.31. *A maximal planar graph on $n \geqslant 4$ vertices is 3-connected.*

Exercise 2.32. *Prove Theorem 2.31.*

Exercise 2.33. *(a) A* minimal nonplanar *graph is a non-planar graph G which contains an edge e such that $G \setminus e$ is planar. Prove or disprove: Every minimal nonplanar graph contain an edge e such that $G \setminus e$ is maximal planar.*

(b) A maximal-plus-one planar *graph is a graph G that contains an edge e such that $G \setminus e$ is maximal planar. Prove or disprove: Every maximal-plus-one planar graph can be drawn with at most one crossing.*

Many questions about graphs are formulated only for connected graphs because it is easy to add edges to disconnected graphs and make them connected. For similar reason, many questions about planar embeddings are formulated only for maximal planar graphs because it is easy to augment planar graphs and make them maximal planar. Well, this last statement is not entirely obvious. Let us look at it in more detail.

An augmentation of a given planar graph $G = (V, E)$ to a maximal planar graph $G' = (V, E')$ where $E' \supseteq E$ is also called a *topological triangulation*. The proof of Lemma 2.30 already contains the basic algorithmic idea to topologically triangulate a plane graph.

Theorem 2.34. *For a given connected plane graph $G = (V, E)$ on n vertices one can compute in $O(n)$ time and space a maximal plane graph $G' = (V, E')$ with $E \subseteq E'$.*

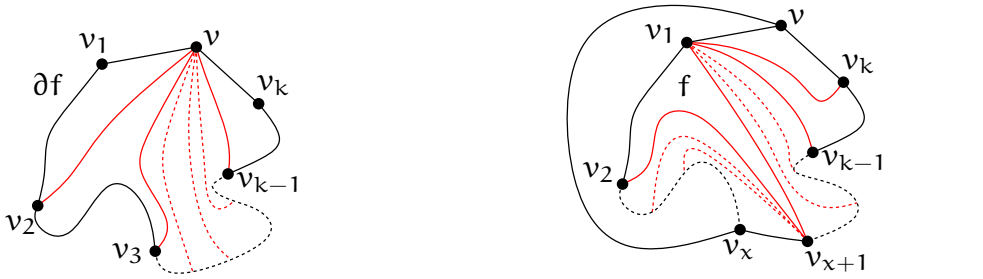
Proof. Suppose, for instance, that G is represented as a DCEL[2], from which one can easily extract the face boundaries. As a clean-up, we walk along the boundary of each

[2] If you wonder how the possibly complicated curves are represented: they do not need to be, since here we need a representation of the combinatorial embedding only.

face. Whenever we see a vertex twice (or more), it must be a cut vertex. We fix this by adding an edge between its current predecessor and successor along the walk, and then continue the walk. Since the total number of traversed edges and vertices of all faces is proportional to $|E|$, which by Corollary 2.5 is linear, the clean-up finishes in $O(n)$ time. Henceforth we may suppose that all faces of G are bounded by cycles.

Every face that is bounded by more than three vertices selects an arbitrary vertex on its boundary. Conversely, every vertex keeps a list of all faces that have selected it. Then we process every vertex $v \in V$ as follows:

1. Mark all neighbors of v.
2. For each face f that selected v, scan its boundary $\partial f = (v, v_1, \ldots, v_k)$ counterclockwise, where $k \geqslant 3$, and find the first marked vertex $v_x \notin \{v_1, v_k\}$.
   - If there is no such vertex, we can safely triangulate f using a star from v, that is, by adding the edges vv_i, for $i \in \{2, \ldots, k-1\}$ (Figure 2.17a). We then mark the new neighbors of v accordingly.
   - Otherwise, the edge vv_x as a curve embedded outside f prevents any vertex in $\{v_1, \ldots, v_{x-1}\}$ from connecting to any vertex in $\{v_{x+1}, \ldots, v_k\}$ by an edge in G. (The reasoning copies the one we made for the edges v_0v_2 and v_1v_3 in the proof of Lemma 2.30 above; see Figure 2.16a.) So we can safely triangulate f using a bi-star from v_1 and v_{x+1}, that is, by adding the edges v_1v_i, for $i \in \{x+1, \ldots, k\}$, and v_jv_{x+1}, for $j \in \{2, \ldots, x-1\}$ (Figure 2.17b).
3. After finishing all faces that seleted v, we conclude the processing of v by clearing all marks on its neighbors.



(a) Case 1: v does not have any neighbor on ∂f other than v_1 and v_k.

(b) Case 2: v has a neighbor v_x on ∂f other than v_1 and v_k.

Figure 2.17: *Topologically triangulating a plane graph.*

Regarding the runtime bound, note that every face is visited only twice: one time when selecting its representative vertex, the other time when scanning its boundary. In this way, each edge is touched a constant number of times in step 2 overall. The marking/unmarking (steps 1 and 3) cost $\sum_{v \in V} \deg(v) = 2|E|$ time by the Handshaking

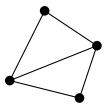
Lemma. Therefore, the total time can be bounded by $O(n + |F| + |E|) = O(n)$ by Corollary 2.5. □

Using any of the standard planarity testing algorithms we can obtain a combinatorial embedding of a planar graph in linear time. Together with Theorem 2.34 this yields:

Corollary 2.35. *For a given planar graph $G = (V, E)$ on n vertices one can compute in $O(n)$ time and space a maximal planar graph $G' = (V, E')$ with $E \subseteq E'$.* □

The results discussed in this section can serve as a tool to fix the combinatorial embedding for a given graph G: augment G using Theorem 2.34 to a maximal planar graph G', whose combinatorial embedding is unique by Theorem 2.27.

Being maximal planar is a property of an abstract graph. In contrast, a geometric graph to which no straight-line edge can be added without crossing is called a *triangulation*. Not every triangulation is maximal planar, as the example depicted to the right shows.



It is also possible to triangulate a geometric graph in linear time. But this problem is much more involved. Triangulating a single face of a geometric graph amounts to what is called "triangulating a simple polygon". This can be done in near-linear[3] time using standard techniques, and in linear time using Chazelle's famous algorithm, whose description spans a fourty pages paper [9].

Exercise 2.36. *We discussed the DCEL structure to represent plane graphs in Section 2.2.1. An alternative way to represent an embedding of a maximal planar graph is the following: For each triangle, store pointers to its three vertices and to its three neighboring triangles. Compare both approaches. Discuss different scenarios where you would prefer one over the other. In particular, analyze the space requirements of both.*

Connectivity serves as an important indicator for properties of planar graphs. Already Wagner showed that a 4-connected graph is planar if and only if it does not contain K_5 as a minor. That is, assuming 4-connectivity the second forbidden minor $K_{3,3}$ becomes "irrelevant". For subdivisions this is a different story. Independently Kelmans and Semour conjectured in the 1970s that 5-connectivity allows to consider K_5 subdivisions only. This conjecture was proven only recently[4] by Dawei He, Yan Wang, and Xingxing Yu.

Theorem 2.37 (He, Wang, and Yu [18]). *Every 5-connected nonplanar graph contains a subdivision of K_5.*

Exercise 2.38. *Give a 4-connected nonplanar graph that does not contain a subdivision of K_5.*

[3] $O(n \log n)$ or—using more elaborate tools—$O(n \log^* n)$ time.
[4] The result was announced in 2015 and published in 2020.

Another example that illustrates the importance of connectivity is the following famous theorem of Tutte that provides a sufficient condition for Hamiltonicity.

Theorem 2.39 (Tutte [32]). *Every 4-connected planar graph is Hamiltonian.*

Moreover, for a given 4-connected planar graph a Hamiltonian cycle can also be computed in linear time [10].

2.5 Compact Straight-Line Drawings

As a next step we consider geometric plane embeddings, where every edge is drawn as a straight-line segment. A classical theorem of Wagner and Fáry states that this is not a restriction to plane embeddability.

Theorem 2.40 (Fáry [13], Wagner [33]). *Every planar graph has a plane straight-line embedding.*

This is quite surprising, considering how much more freedom a simple curve allows, compared to a line segment which is completely determined by its endpoints. To further increase the level of appreciation, let us remark that a similar "straightening" is generally not possible if we fix the point set on which the vertices are to be embedded: On the one hand, Pach and Wenger [27] showed that a given planar graph G on n vertices $v_1, \ldots, v_n$ and a given point set $\{p_1, \ldots, p_n\} \subset \mathbb{R}^2$, one can always find a plane embedding of G such that $v_i \mapsto p_i$, for all $i \in \{1, \ldots, n\}$. On the other hand, this is not possible in general with a plane *straight-line* embedding. For instance, K_4 does not admit a plane straight-line embedding on a set of points that form a convex quadrilateral, such as a rectangle. In fact, it is NP-hard to decide whether a given planar graph admits a plane straight-line embedding on a given point set [7].

Exercise 2.41. *Show the following:*

(a) For every natural number $n \geqslant 4$, there exist a planar graph G on n vertices and a set $P \subset \mathbb{R}^2$ of n points in general position (no three points are collinear) so that G does not admit a plane straight-line embedding on P.

(b) For every natural number $n \geqslant 6$, there exist a planar graph G on n vertices and a set $P \subset \mathbb{R}^2$ of n points in general position (no three points are collinear) so that (1) G does not admit a plane straight-line embedding on P; and (2) there are three points in P forming a triangle that contains all other points from P.

Exercise 2.42. *Show that for every set $P \subset \mathbb{R}^2$ of $n \geqslant 3$ in general position (no three points are collinear) the cycle on n vertices admits a plane straight-line embedding on P.*

Although Fáry-Wagner's theorem has a nice inductive proof, we do not discuss it here. Instead we will soon prove a stronger statement that implies the theorem.

A very desirable property of straight-line embeddings is that they are easy to represent: only the points/coordinates for the vertices are needed. But from an algorithmic and complexity point of view it is also important to learn the space requirement for the coordinates, since it affects the input and output size of algorithms that work on embedded graphs. While the Fáry-Wagner Theorem guarantees the existence of a plane straight-line embedding for every planar graph, it does not bound the size of the coordinates. The following strengthening provides such bounds, via an explicit algorithm that embeds (without crossing) a given planar graph on a linear size integer grid.

Theorem 2.43 (de Fraysseix, Pach, Pollack [15]). *Every planar graph on $n \geqslant 3$ vertices has a plane straight-line drawing on a $(2n-3) \times (n-1)$ integer grid. In fact, it can be computed in $O(n)$ time.*

2.5.1 Canonical Orderings

The key concept behind the algorithm is the notion of a canonical ordering, which is a vertex order that allows building the plane drawing inside out (hence canonical). Reading it backwards one may imagine a shelling or peeling order that destructs the graph from the outside. A canonical ordering also provides a succinct representation for the combinatorial embedding.

Definition 2.44. *A plane graph G is* **internally triangulated** *if it is biconnected and every bounded face is a (topological) triangle. We denote by $C_\circ(G)$ its outer cycle, that is, the cycle bounding its outer face.*

Definition 2.45. *Let G be an internally triangulated plane graph. A permutation $\pi = (v_1, v_2, \ldots, v_n)$ of $V(G)$ is a* **canonical ordering** *for G if for all $k \in \{3, \ldots, n\}$ we have*

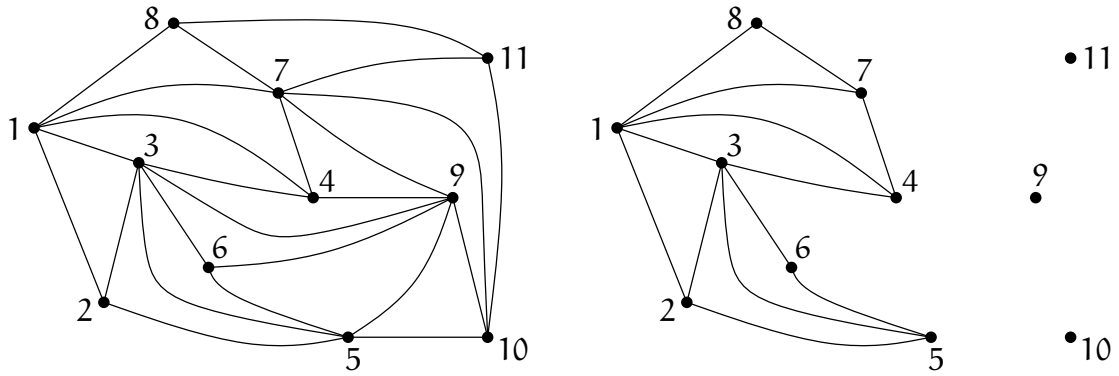
(CO1) G_k is internally triangulated;

(CO2) $v_1v_2 \in C_\circ(G_k)$; and

(CO3) v_k is located in the outer face of G_{k-1},

where $G_k := G[\{v_1, \ldots, v_k\}]$ denotes the induced drawing on the first k vertices.

Figure 2.18 shows an example with canonical ordering $(1, 2, \ldots, 8)$. Note that not every permutation is a valid canonical ordering. For instance, if π chooses its first seven vertices from $\{1, 2, 3, 5, 6, 7, 8\}$, then the induced subgraph $G[\{1, 2, 3, 5, 6, 7, 8\}]$ is not biconnected since 1 is a cut vertex. Thus π is not a canonical ordering. Alternatively we may think about it backwards: Suppose we choose the initial three removals from $\{9, 10, 11\}$ as shown in Figure 2.18b, then the next removal cannot be 4 because its removal would create a cut vertex in the resulting graph.



(a) G (b) G_8

Figure 2.18: *An internally triangulated plane graph with one of its canonical orderings* $(1, 2, \ldots, 8)$.

Theorem 2.46. *For every internally triangulated plane graph* G *and every edge* v_1v_2 *on its outer cycle, there exists a canonical ordering for* G *that starts with* v_1, v_2*. Moreover, such an ordering can be computed in linear time.*

Proof. Induction on n, the number of vertices. For a triangle, any ordering is valid and so the statement holds. Now consider an internally triangulated plane graph $G = (V, E)$ on $n \geqslant 4$ vertices. *Assume* we find a vertex $v_n \in C_\circ(G) \setminus \{v_1, v_2\}$ such that the plane graph $G_{n-1} := G \setminus \{v_n\}$ is internally triangulated. (We will show later that such a vertex always exists.) Then we can apply induction on G_{n-1} to obtain a canonical ordering $(v_1, v_2, \ldots, v_{n-1})$ for G_{n-1}. The extended ordering $(v_1, v_2, \ldots, v_n)$ satisfies (CO1)–(CO3) for $k \in \{3, \ldots, n-1\}$ by the induction hypothesis, but also for $k = n$ by definition of v_n. Hence the induction is complete, assuming the existence of v_n.

It remains to argue that v_n exists. We will show this in two steps:

(1) we can find a $v_n \in C_\circ(G) \setminus \{v_1, v_2\}$ that is not incident to a chord of $C_\circ(G)$; and

(2) such a vertex v_n guarantees that $G_{n-1} := G \setminus \{v_n\}$ is internally triangulated.

First we show (1). If $C_\circ(G)$ does not have any chord, this is obvious because every cycle has at least three vertices, one of which is neither v_1 nor v_2. So suppose that $C_\circ(G)$ has a chord c. The endpoints of c split $C_\circ(G)$ into two paths, one of which does not have v_1 nor v_2 as an internal vertex. We call this path the path *associated* to c. (Such a path has at least two edges because there is always at least one vertex "behind" a chord.) Among all chords of $C_\circ(G)$ we select c such that its associated path has minimal length. Then by this choice of c its associated path together with c forms an induced cycle in G. In particular, none of the (at least one) interior vertices of the path associated to c is incident to a chord of $C_\circ(G)$ because such a chord would either cross c or it would have an associated path that is strictly shorter than the one associated to c. So we can select v_n from these vertices. By definition the path associated to c does not contain v_1 nor v_2, hence this procedure does not select either of these vertices.

Then we look at (2). The way G_{n-1} is obtained from G, every bounded face f of G_{n-1} also appears as a bounded face of G. As G is internally triangulated, f is a triangle. It remains to show that G_{n-1} is biconnected.

Consider the circular sequence of neighbors around v_n in G and break it into a linear sequence $u_1, \dots, u_m$, for some $m \geqslant 2$, that starts and ends with the neighbors of v_n in $C_\circ(G)$. As G is internally triangulated, each of the bounded faces spanned by v_n, u_i, u_{i+1}, for $i \in \{1, \dots, m-1\}$, is a triangle and hence $u_i u_{i+1} \in E$. The boundary of the outer face of G_{n-1} is obtained from $C_\circ(G)$ by replacing v_n with the (possibly empty) sequence $u_2, \dots, u_{m-1}$. As v_n is not incident to a chord of $C_\circ(G)$ (and so none of $u_2, \dots, u_{m-1}$ appeared along $C_\circ(G)$ already), the resulting sequence forms a cycle, indeed. Add a new vertex v in the outer face of G_{n-1} and connect v to every vertex of $C_\circ(G_{n-1})$ to obtain a maximal planar graph $H \supset G_{n-1}$. By Theorem 2.31 the graph H is 3-connected and so G_{n-1} is biconnected, as desired. This also completes the proof of the claim.

Regarding the runtime bound, we maintain for each vertex v whether it is on the current outer cycle and what is the number of incident chords with respect to the current outer cycle. Given a combinatorial embedding of G, it is straighforward to initialize this information in linear time. (Every edge is considered at most twice, once for each endpoint on the outer cycle.) We also maintain an unordered list of the *eligible* vertices, that is, those vertices that are on the outer cycle and not incident to any chord. This list is straightforward to maintain: Whenever a vertex information is updated, check before and after the update whether it is eligible and correspondingly add it to or remove it from the list of eligible vertices. We store with each vertex a pointer to its position in the list (*nil* if it is not eligible currently) so that we can remove it from the list in constant time if needed.

When removing a vertex v_n from G, there are two cases: Either v_n has two neighbors u_1 and u_2 only (Figure 2.19a), in which case the edge u_1u_2 ceases to be a chord. Thus, the chord count for u_1 and u_2 has to be decremented by one. Otherwise, there are $m \geqslant 3$ neighbors $u_1, \dots, n_m$ (Figure 2.19b) and (1) all vertices $u_2, \dots, u_{m-1}$ are new on the outer cycle, and (2) every edge incident to u_i, for $i \in \{2, \dots, m-1\}$, and some other vertex on the outer cycle other than u_{i-1} or u_{i+1} is a new chord. These latter changes have to be reflected in the chord counters at the vertices. So to update these counters, we inspect all edges incident to one of $u_2, \dots, u_{m-1}$. For each such edge, we check whether the other endpoint is on the outer cycle and, if so, increment the counter.

During the course of the algorithm every vertex appears once as a new vertex on the outer cycle. At this point all incident edges (in the current graph G_i) are examined. Similarly, when a vertex v_k is removed from G_K, all edges incident to v_k in G_k are inspected so as to discover if and which new vertices appear on the outer cycle. Finally, each vertex is removed at most once. Therefore, every edge is inspected at most three times: when one of its two endpoints appears first on the outer cycle, and when the first endpoint (and therefore the edge) is removed. Altogether this takes linear time because the number of edges in G is linear by Corollary 2.5. □

Using one of the linear time planarity testing algorithms, we can obtain a combinato-

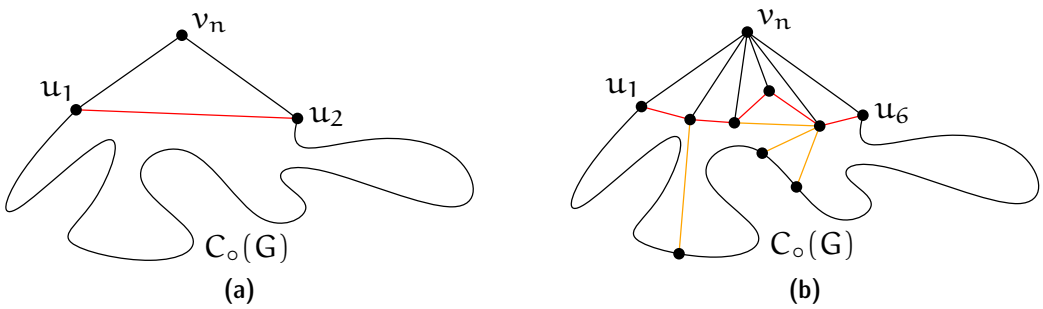
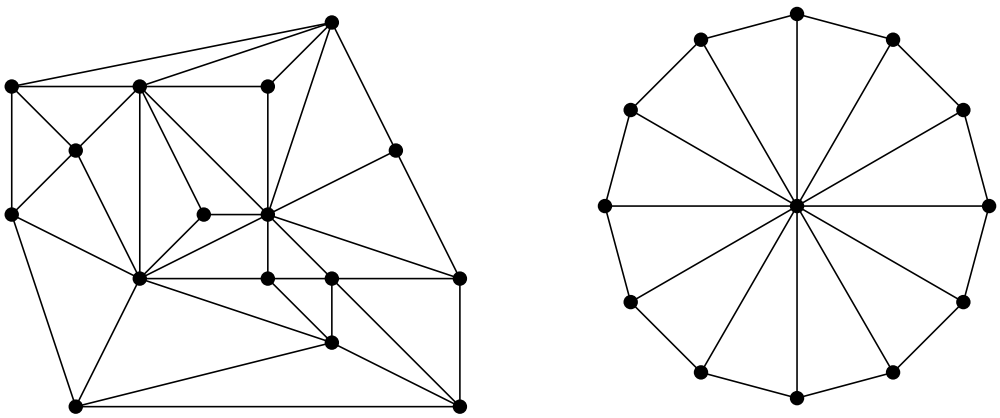


Figure 2.19: *Processing a vertex when computing a canonical ordering.*

rial embedding for a given maximal planar graph G. As every maximal planar graph is 3-connected (Theorem 2.31), this embedding is unique (Theorem 2.27). Then, as every maximal plane graph is also internally triangulated, we can use Theorem 2.46 to provide us with a canonical ordering for (the unique embedding of) G, in overall linear time.

Corollary 2.47. *Every maximal planar graph admits a canonical ordering. Moreover, such an ordering can be computed in linear time.* □

Exercise 2.48. *(a) Compute a canonical ordering for the following internally triangulated plane graphs:*



(b) Design an infinite family of internally triangulated plane graphs on $2k$ vertices with at least $k!$ canonical orderings.

(c) Design an infinite family of internally triangulated plane graphs, along with specific choices for v_1, v_2, so that each graph in the family has a unique canonical ordering starting from v_1, v_2.

Exercise 2.49. *(a) Describe a plane graph G with n vertices that can be embedded (while preserving the outer face) in straight-line on a grid of size $(2n/3) \times (2n/3)$, but not on a smaller grid.*

(b) Can you draw G *on a smaller grid if you are allowed to change the outer face?*

As simple as they may appear, canonical orderings are a powerful and versatile tool to work with plane graphs. As an example, consider the following partitioning theorem.

Theorem 2.50 (Schnyder [30]). *For every maximal planar graph* G *on at least three vertices and every fixed face* f *of* G*, the multigraph obtained from* G *by doubling the (three) edges of* f *can be partitioned into three spanning trees.*

Exercise 2.51. *Prove Theorem 2.50.* Hint: *Fix a canonical ordering; for every vertex* v_k *take the edge to its first neighbor on* $C_\circ(G_{k-1})$*; argue that the edges form a spanning tree.*

Of a similar flavor is the following question.

Problem 2.52 (In memoriam Ferran Hurtado (1951–2014)).
Can every complete geometric graph on $n = 2k$ vertices (in general position) be partitioned into k plane spanning trees?

There are several positive results for special point sets [1, 5], and it is also known that there are always $\lfloor n/3 \rfloor$ edge disjoint plane spanning trees [4]. The general statement above has been refuted very recently [26]. However, it remains open if there always exists a partition into $k+1$ plane trees—or more generally, what is the minimum number of plane trees that always suffices.

2.5.2 The Shift-Algorithm

Let $(v_1, \ldots, v_n)$ be a canonical ordering of maximal planar graph G. The plan is to insert vertices in this order and extend the embedding incrementally, starting from the triangle $P(v_1) = (0,0)$, $P(v_3) = (1,1)$, $P(v_2) = (2,0)$; see Figure 2.20.

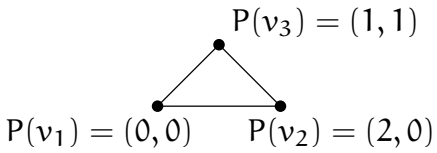


Figure 2.20: *Initialization of the shift algorithm.*

At each step, some vertices are shifted to the right, making room for the insertion of a new vertex. For each vertex v_k inserted, we define a "frozen" set $F(v_k)$ to collect all vertices that move rigidly with v_k in the future. For the first three vertices we define $F(v_i) = \{v_i\}$, for $1 \leqslant i \leqslant 3$. Once defined, the set will not change anymore.

We ensure the following invariants after Step k (that is, after we have inserted v_k):

(i) We obtain a straight-line embedding of $G_k := G[\{v_1, \ldots, v_k\}]$ on the integer grid, combinatorially equivalent to the one considered in the canonical ordering. Moreover, $P(v_1) = (0,0)$ and $P(v_2) = (2k-4, 0)$.

(ii) Denote the outer cycle by $C_\circ(G_k) =: (w_1, \ldots, w_t)$ where $w_1 = v_1$ and $w_t = v_2$. The x-coordinates of $w_1, \ldots, w_t$ are strictly increasing.[5]

(iii) Each edge $w_i w_{i+1}$, for $1 \leqslant i < t$, of $C_\circ(G_k)$ is drawn as a line segment with slope ± 1. In particular, the Manhattan distance[6] between any two points on $C_\circ(G_k)$ is even.

(iv) The sets $F(w_1), \ldots, F(w_t)$ partition $\{v_1, \ldots, v_k\}$.

Clearly these invariants hold for G_3, embedded as described above.

Idea for Step $k+1$. We are about to place vertex v_{k+1}. Its neighbors $w_\ell, \ldots, w_r$ lie consecutively on $C_\circ(G_k)$ by the property of canonical ordering. Put v_{k+1} at position $\mu(P(w_\ell), P(w_r))$, where

$$\mu((x_\ell, y_\ell), (x_r, y_r)) := \left(\frac{x_\ell - y_\ell + x_r + y_r}{2}, \frac{-x_\ell + y_\ell + x_r + y_r}{2} \right)$$

is the intersection between the line $y = x - x_\ell + y_\ell$ of slope 1 through (x_ℓ, y_ℓ) and the line $y = x_r - x + y_r$ of slope -1 through (x_r, y_r).

Proposition 2.53. *If the Manhattan distance between $P(w_\ell)$ and $P(w_r)$ is even, then $\mu(P(w_\ell), P(w_r))$ is on the integer grid.*

Proof. By (ii) we know that $x_\ell < x_r$. Suppose without loss of generality that $y_\ell \leqslant y_r$. The Manhattan distance of the two points is $d := x_r - x_\ell + y_r - y_\ell$, an even number by assumption. Adding an even number $2x_\ell$ to d yields the even number $x_r + x_\ell + y_r - y_\ell$, half of which is the x-coordinate of $\mu((x_\ell, y_\ell), (x_r, y_r))$. Adding an even number $2y_\ell$ to d yields the even number $x_r - x_\ell + y_r + y_\ell$, half of which is the y-coordinate of $\mu((x_\ell, y_\ell), (x_r, y_r))$. □

However, $\mu(P(w_\ell), P(w_r))$ may be unable to "see" all of $w_\ell, \ldots, w_r$, in case that the slope of $w_\ell w_{\ell+1}$ is 1 and/or the slope of $w_{r-1} w_r$ is -1 (Figure 2.21).

In order to resolve these problems, we shift some points to the right so that $w_{\ell+1}$ no longer lies on the line of slope 1 through w_ℓ, and that w_{r-1} no longer lies on the line of slope -1 through w_r. The actual Step $k+1$ then reads:

1. Shift $\bigcup_{i=\ell+1}^{r-1} F(w_i)$ to the right by one unit.
2. Shift $\bigcup_{i=r}^{t} F(w_i)$ to the right by two units.
3. $P(v_{k+1}) := \mu(P(w_\ell), P(w_r))$.
4. $F(v_{k+1}) := \{v_{k+1}\} \cup \bigcup_{i=\ell+1}^{r-1} F(w_i)$.

[5] The notation is a bit sloppy because both t and the w_i depend on k. So in principle we should write w_i^k instead of w_i. But as the k would just make a constant appearance throughout, we omit it to avoid clutter.

[6] The *Manhattan distance* of two points (x_1, y_1) and (x_2, y_2) is $|x_2 - x_1| + |y_2 - y_1|$.

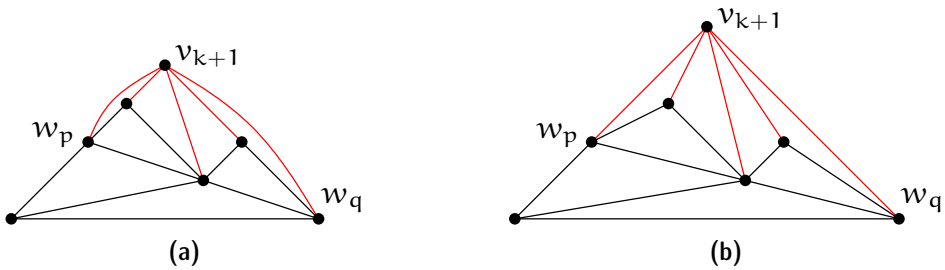


Figure 2.21: *(a) The new vertex v_{k+1} is adjacent to all of $w_\ell, \dots, w_r$. If we place v_{k+1} at $\mu(P(w_\ell), P(w_r))$, then some edges may overlap, in case that $w_{\ell+1}$ lies on the line of slope 1 through w_ℓ or w_{r-1} lies on the line of slope -1 through w_r; (b) shifting $w_{\ell+1}, \dots, w_{r-1}$ by one and $w_r, \dots, w_t$ by two units to the right solves the problem.*

Next we argue that the invariants (i)–(iv) are maintained after Step $k+1$.

For (i), note that the shifting always starts from $w_{\ell+1}$ onward. So $w_1 = v_1$ is never moved and stays at $P(v_1) = (0,0)$. On the other hand, we shift every vertex by two starting from (and including) w_r, hence v_2 moves two units to $P(v_2) = (2(k+1)-4, 0)$.

Also, observe that the Manhattan distance between w_ℓ and w_r remains even because the shift increases their horizontal distance by two and leaves the y-coordinates unchanged. Therefore by Proposition 2.53 the vertex v_{k+1} is embedded on the integer grid indeed.

After shifting, the absolute slopes of the edges $w_\ell w_{\ell+1}$ and $w_{r-1}w_r$ (possibly the same edge) become < 1, and the absolute slopes of all other edges on $C_\circ(G_k)$ remain 1. In contrast, the edges $v_{k+1}w_\ell$ and $v_{k+1}w_r$ both have absolute slope 1, and all edges from v_{k+1} to $w_{\ell+1}, \dots, w_{r-1}$ have absolute slopes > 1. Hence, for all $i \in \{\ell, \dots, r\}$, the edge $v_{k+1}w_i$ intersects $C_\circ(G_k)$ in exactly one point, which is w_i. In other words, these new edges will not cross anything in G_k.

Of course, to conclude that the drawing is plane, we also need to argue that the edges originally in G_k do not clash with each other after shifting. But as this is intuitively clear, we postpone the formal argument for later. Now (i) is complete.

For (ii), clearly both the shifts and the insertion of v_{k+1} maintain the strict order along the outer cycle. For (iii), note that the edges $w_\ell w_{\ell+1}$ and $w_{r-1}w_r$ (possibly equal) are the only edges on the outer cycle $C_\circ(G_k)$ whose slope is changed. But neither edge appears on $C_\circ(G_{k+1})$ any more, as they are covered by the two new edges $v_{k+1}w_\ell$ and $v_{k+1}w_r$; these new edges have slope 1 and -1, respectively. Regarding (iv), the set $F(v_{k+1})$ by definition includes the new vertex v_{k+1} and encompasses the sets of all outer cycle vertices that it covers (that is, they are on $C_\circ(G_k)$ but not on $C_\circ(G_{k+1})$). So the sets on $C_\circ(G_{k+1})$ partition $\{v_1, \dots, v_{k+1}\}$.

So (i)–(iv) are invariants of the algorithm, indeed. Let us look at the consequences. During the entire procedure, invariants (i)(ii) and the definition of μ ensures that each point is placed on a $(2n-3) \times (n-2)$ integer grid. In fact, the final vertex v_n is always

placed at $\mu(P(v_1), P(v_2)) = \mu((0,0),(2n-4,0)) = (n-2, n-2)$ since both v_1 and v_2 are its neighbors.

Finally, we return to provide a formal argument that the "interior part" of the drawing remains plane under shifts.

Lemma 2.54. *Let G_k, $k \geqslant 3$, be straight-line embedded on grid as described by the algorithm. Assume $C_\circ(G_k) = (w_1, \ldots, w_t)$, and let $\delta_1 \leqslant \ldots \leqslant \delta_t$ be nonnegative integers. If for each i we shift $F(w_i)$ by δ_i to the right, then the resulting straight-line drawing is plane.*

Proof. Induction on k. For the base case G_3 this is obvious. Now for G_k, assume $v_k = w_\ell$, where $2 \leqslant \ell < t$. Denote its $m \geqslant 2$ neighbors as $u_1, \ldots, u_m$ where $u_1 = w_{\ell-1}$ and $u_m = w_{\ell+1}$. Then we have

$$C_\circ(G_{k-1}) = (w_1, \ldots, w_{\ell-1}, \underbrace{u_2, \ldots, u_{m-1}}_{\text{could be empty}}, w_{\ell+1}, \ldots, w_t).$$

Recall that the algorithm defines $F(v_k) = \{v_k\} \cup \bigcup_{i=1}^{m} F(u_i)$. Hence, to shift each $F(w_i)$ by δ_i is equivalent to applying the sequence

$$\Delta := (\delta_1, \ldots, \delta_{\ell-1}, \underbrace{\delta_\ell, \ldots, \delta_\ell}_{m-2 \text{ times}}, \delta_{\ell+1}, \ldots, \delta_t)$$

to G_{k-1} and then shifting v_k by δ_ℓ.

Clearly Δ is monotonically increasing, so by the inductive assumption the shifted drawing of G_{k-1} is plane. After shifting v_k by δ_ℓ, the drawing of G_k is plane: Vertex v_k moves rigidly with its neighbors $u_2, \ldots, u_{m-1}$, and the two extreme neighbors u_1 and u_m move relatively to the left and right, respectively. The corresponding edges cannot cross anything during this movement. □

Linear time. The challenge in implementing the shift algorithm efficiently lies in the eponymous shift operations, which modify the x-coordinates of potentially many vertices. In fact, it is not hard to see that a naive implementation—which keeps track of all coordinates explicitly—may use quadratic time. De Fraysseix et al. described an implementation of the shift algorithm that uses $O(n \log n)$ time. Then Chrobak and Payne [11] observed how to improve the runtime to linear, using the following ideas.

Recall that $P(v_{k+1})$ is placed at the coordinates

$$x = \frac{x_\ell - y_\ell + x_r + y_r}{2} \quad \text{and} \quad y = \frac{(x_r - x_\ell) + y_\ell + y_r}{2}, \tag{2.55}$$

and thus

$$x - x_\ell = \frac{(x_r - x_\ell) + y_r - y_\ell}{2}. \tag{2.56}$$

In other words, to determine the y-coordinate and the x-offset relative to the leftmost neighbor w_ℓ, we only need to know the y-coordinates of $P(w_\ell)$ and $P(w_r)$ together with x-offset of $P(w_r)$ relative to $P(w_\ell)$.

To exploit these relations, we represent the geometric embedding of G_k as follows. For each vertex v_i, for $1 \leqslant i \leqslant k$, we store three integers:

1. the y-coordinate $y(v_i)$, which is determined when v_i is inserted and remains the same throughout the algorithm,
2. an index (or pointer) to a parent vertex $p(v_i) \in V(G_k)$, and
3. the x-offset $\Delta(v_i)$ of $P(v_i)$ relative to $P(p(v_i))$.

The *parent vertex* $p(v_i)$ of a vertex v_i on $C_\circ(G_k)$ is its predecessor on $C_\circ(G_k)$, that is, if $C_\circ(G_k) = (w_1, \ldots, w_t)$, then $p(w_j) = w_{j-1}$, for $j > 1$, and $p(w_1) = \mathit{nil}$. For a vertex v_i in $G_k \setminus C_\circ(G_k)$ its reference vertex $p(v_i)$ is the vertex v_x, where $i < x \leqslant k$, that covered v_i, that is, such that v_i is on $C_\circ(G_{x-1})$ but not on $C_\circ(G_x)$. Note that in the latter case $p(v_i)$ is can equivalently be described as the neighbor of v_i in G that appears last in the canonical ordering used.

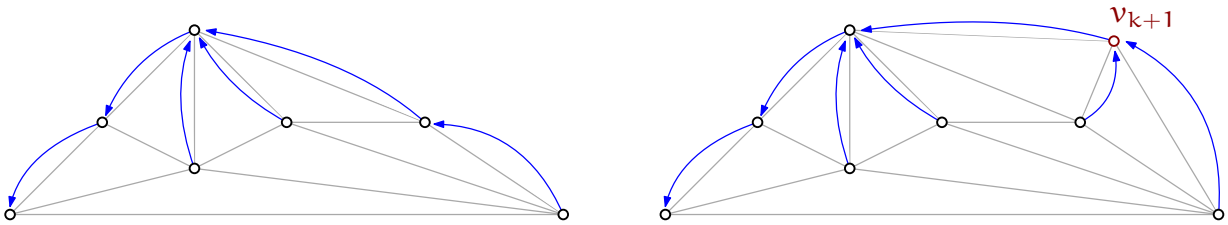


Figure 2.22: *Maintaining the reference tree structure under insertion of a vertex* v_{k+1}.

The parent structure represents a tree that is rooted at v_1, see Figure 2.22 for an example. Knowing the x-coordinate of $P(p(v))$ and the offset $\Delta(v)$, we can easily compute the x-coordinate of $P(v)$. So fixing the x-coordinate of $P(v_1)$, which is zero by Invariant (i), fixes the x-coordinates of all vertices in the drawing. Thus, using this information for $G = G_n$, we can compute all x-coordinates in the final embedding using a simple loop in $O(n)$ time:

```
x[1] = 0
for i = n downto 2:
  x[i] = x[p[i]] + delta[i]
```

Algorithmically, we compute both $p(v_i)$ and $\Delta(v_i)$ in the step when v_i is covered by v_{k+1}, and then these values remain the same for the rest of the algorithm, as the edge $v_i v_{k+1}$ is frozen. Let $w_{\ell+1}, \ldots, w_{r-1}$ denote the vertices covered by inserting v_{k+1}. If $r = \ell + 1$, no vertex is covered and there is nothing to do. So suppose that $r - \ell \geqslant 2$.

We first update the parent information and compute the x-offset of w_r relative to w_ℓ, by iterating over $w_{\ell+1}, \ldots, w_r$, setting their parent to v_{k+1} and summing up the individual offsets. This allows us to compute $\Delta(v_{k+1})$ using (2.56). Then we iterate a second time over $w_{\ell+1}, \ldots, w_r$ to compute the x-offset $\lambda(w_j)$ of $P(w_j)$ relative to $P(w_\ell)$, for all $j \in \{\ell+1, \ldots, r\}$ and use these values to rebase the offset to be relative to $P(v_{k+1})$ instead, as shown in the code fragment below.

```
lambda = 0
for j = l+1 to r:
  lambda = lambda + delta[j]
  delta[j] = lambda - delta[k+1]
```

This concludes the description of the algorithm and its runtime analysis.

2.5.3 Remarks and Open Problems

From a geometric complexity point of view, Theorem 2.43 provides very good news for planar graphs in a similar way that the Euler Formula does from a combinatorial complexity point of view. Euler's Formula tells us that we can obtain a combinatorial representation (for instance, as a DCEL) of any plane graph using $O(n)$ space, where n is the number of vertices. Now the shift algorithm tells us that for any planar graph we can even find a geometric plane (straight-line) representation using $O(n)$ space. In addition to the combinatorial information, we only have to store $2n$ numbers from the range $\{0, 1, \ldots, 2n-4\}$.

When we make such claims regarding space complexity we implicitly assume the so-called *word RAM model*. In this model each memory cell stores a *word* of b bits, which may represent any integer in $\{0, \ldots, 2^b - 1\}$. One also assumes that b is sufficiently large, in our case $b \geqslant \log n$.

There are also different models such as the *bit complexity model*, where one is charged for every bit used to store information. In our case that would already incur an additional factor of $\log n$ for the combinatorial representation: for instance, for each halfedge we store its endpoint, which is an index from $\{1, \ldots, n\}$.

Edge lengths. Theorem 2.43 shows that planar graphs admit a plane straight-line drawing where all vertices have integer coordinates. It is an open problem whether a similar statement can be made for edge lengths.

Problem 2.57 (Harborth's Conjecture [17]). Every planar graph admits a plane straight-line drawing where all Euclidean edge lengths are integral.

Without the planarity restriction such a drawing is possible because for every $n \in \mathbb{N}$ one can find a set of n points in the plane, not all collinear, such that their distances are all integral. In fact, such a set of points can be constructed to lie on a circle of integral radius [2]. When mapping the vertices of K_n onto such a point set, all edge lengths are

integral. In the same paper it is also shown that there exists no infinite set of points in the plane so that all distances are integral, unless all of these points are collinear. Unfortunately, collinear point sets are not very useful for drawing graphs. The existence of a dense subset of the plane where all distances are rational would resolve Harborth's Conjecture. However, it is not known whether such a set exists, and in fact the suspected answer is "no".

Problem 2.58 (Erdős–Ulam Conjecture [12]). There is no dense set of points in the plane whose Euclidean distances are all rational.

Generalizing the Fáry-Wagner Theorem. As discussed earlier, not every planar graph on n vertices admits a plane straight-line embedding on every set of n points. But Theorem 2.40 states that for every planar graph G on n vertices there *exists* a set P of n points in the plane so that G admits a plane straight-line embedding on P. It is an open problem whether this statement can be generalized to hold for several graphs, in the following sense.

Problem 2.59. What is the largest number $k \in \mathbb{N}$ for which the following statement holds? For every collection of k planar graphs $G_1, \ldots, G_k$ on n vertices each, there exists a set P of n points so that G_i admits a plane straight-line embedding on P, for every $i \in \{1, \ldots, k\}$.

By Theorem 2.40 we know that the statement holds for $k = 1$. Already for $k = 2$ it is not known whether the statement holds. However, it is known that k is finite [8]. Specifically, there exists a collection of 49 planar graphs on 11 vertices each so that for every set P of 11 points in the plane at least one of these graphs does not admit a plane straight-line embedding on P [29]. Therefore we have $k \leqslant 49$.

Questions

1. *What is an embedding? What is a planar/plane graph?* Give the definitions and explain the difference between planar and plane.
2. *How many edges can a planar graph have? What is the average vertex degree in a planar graph?* Explain Euler's formula and derive your answers from it.
3. *How can plane graphs be represented on a computer?* Explain the DCEL data structure and how to work with it.
4. *How can a given plane graph be (topologically) triangulated efficiently?* Explain what it is, including the difference between topological and geometric triangulation. Give a linear time algorithm, for instance, as in Theorem 2.34.
5. *What is a combinatorial embedding? When are two combinatorial embeddings equivalent? Which graphs have a unique combinatorial plane embedding?* Give the definitions, explain and prove Whitney's Theorem.

6. *What is a canonical ordering and which graphs admit such an ordering? For a given graph, how can one find a canonical ordering efficiently?* Give the definition. State and prove Theorem 2.46.
7. *Which graphs admit a plane embedding using straight line edges? Can one bound the size of the coordinates in such a representation?* State and prove Theorem 2.43.

References

[1] Oswin Aichholzer, Thomas Hackl, Matias Korman, Marc van Kreveld, Maarten Löffler, Alexander Pilz, Bettina Speckmann, and Emo Welzl, Packing plane spanning trees and paths in complete geometric graphs. *Inform. Process. Lett.*, 124, (2017), 35–41.

[2] Norman H. Anning and Paul Erdős, Integral distances. *Bull. Amer. Math. Soc.*, 51/8, (1945), 598–600.

[3] Bruce G. Baumgart, A polyhedron representation for computer vision. In *Proc. AFIPS Natl. Comput. Conf.*, vol. 44, pp. 589–596, AFIPS Press, Alrington, Va., 1975.

[4] Ahmad Biniaz and Alfredo García, Packing plane spanning trees into a point set. *Comput. Geom. Theory Appl.*, 90, (2020), 101653.

[5] Prosenjit Bose, Ferran Hurtado, Eduardo Rivera-Campo, and David R. Wood, Partitions of complete geometric graphs into plane trees. *Comput. Geom. Theory Appl.*, 34/2, (2006), 116–125.

[6] John M. Boyer and Wendy J. Myrvold, On the cutting edge: simplified O(n) planarity by edge addition. *J. Graph Algorithms Appl.*, 8/3, (2004), 241–273.

[7] Sergio Cabello, Planar embeddability of the vertices of a graph using a fixed point set is NP-hard. *J. Graph Algorithms Appl.*, 10/2, (2006), 353–363.

[8] Jean Cardinal, Michael Hoffmann, and Vincent Kusters, On universal point sets for planar graphs. *J. Graph Algorithms Appl.*, 19/1, (2015), 529–547.

[9] Bernard Chazelle, Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6/5, (1991), 485–524.

[10] Norishige Chiba and Takao Nishizeki, The Hamiltonian cycle problem is linear-time solvable for 4-connected planar graphs. *J. Algorithms*, 10/2, (1989), 187–211.

[11] Marek Chrobak and Thomas H. Payne, A linear-time algorithm for drawing a planar graph on a grid. *Inform. Process. Lett.*, 54, (1995), 241–246.

[12] Paul Erdős, Ulam, the man and the mathematician. *J. Graph Theory*, 9/4, (1985), 445–449.

[13] István Fáry, On straight lines representation of planar graphs. *Acta Sci. Math. Szeged*, 11/4, (1948), 229–233.

[14] Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl, Trémaux trees and planarity. *Internat. J. Found. Comput. Sci.*, 17/5, (2006), 1017–1030.

[15] Hubert de Fraysseix, János Pach, and Richard Pollack, How to draw a planar graph on a grid. *Combinatorica*, 10/1, (1990), 41–51.

[16] Leonidas J. Guibas and Jorge Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graph.*, 4/2, (1985), 74–123.

[17] Heiko Harborth and Arnfried Kemnitz, Plane integral drawings of planar graphs. *Discrete Math.*, 236/1–3, (2001), 191–195.

[18] Dawei He, Yan Wang, and Xingxing Yu, The Kelmans-Seymour conjecture IV: A proof. *J. Combin. Theory Ser. B*, 144, (2020), 309–358.

[19] John Hopcroft and Robert E. Tarjan, Efficient planarity testing. *J. ACM*, 21/4, (1974), 549–568.

[20] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed, The disjoint paths problem in quadratic time. *J. Combin. Theory Ser. B*, 102/2, (2012), 424–435.

[21] Lutz Kettner, *Software design in computational geometry and contour-edge based polyhedron visualization*. Ph.D. thesis, ETH Zürich, Zürich, Switzerland, 1999.

[22] Kazimierz Kuratowski, Sur le problème des courbes gauches en topologie. *Fund. Math.*, 15/1, (1930), 271–283.

[23] László Lovász, Graph minor theory. *Bull. Amer. Math. Soc.*, 43/1, (2006), 75–86.

[24] Bojan Mohar and Carsten Thomassen, *Graphs on surfaces*, Johns Hopkins University Press, Baltimore, 2001.

[25] David E. Muller and Franco P. Preparata, Finding the intersection of two convex polyhedra. *Theoret. Comput. Sci.*, 7, (1978), 217–236.

[26] Johannes Obenaus and Joachim Orthaber, Edge Partitions of Complete Geometric Graphs (Part 1). *CoRR*, abs/2108.05159, (2021), 1–32.

[27] János Pach and Rephael Wenger, Embedding planar graphs at fixed vertex locations. *Graphs Combin.*, 17, (2001), 717–728.

[28] Neil Robertson and Paul Seymour, Graph Minors. XX. Wagner's Conjecture. *J. Combin. Theory Ser. B*, 92/2, (2004), 325–357.

[29] Manfred Scheucher, Hendrik Schrezenmaier, and Raphael Steiner, A Note on Universal Point Sets for Planar Graphs. *J. Graph Algorithms Appl.*, 24/3, (2020), 247–267.

[30] Walter Schnyder, Planar graphs and poset dimension. *Order*, 5, (1989), 323–343.

[31] Carsten Thomassen, Kuratowski's Theorem. *J. Graph Theory*, 5/3, (1981), 225–241.

[32] William T. Tutte, A theorem on planar graphs. *Trans. Amer. Math. Soc.*, 82/1, (1956), 99–116.

[33] Klaus Wagner, Bemerkungen zum Vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46, (1936), 26–32.

[34] Klaus Wagner, Über eine Eigenschaft der ebenen Komplexe. *Math. Ann.*, 114/1, (1937), 570–590.

[35] Kevin Weiler, Edge-based data structures for solid modeling in a curved surface environment. *IEEE Comput. Graph. Appl.*, 5/1, (1985), 21–40.

[36] Hassler Whitney, Congruent graphs and the connectivity of graphs. *Amer. J. Math.*, 54/1, (1932), 150–168.