Drawing Planar Graphs

Lucie Martinet

November 29, 2010

1 Introduction

The field of planar graph drawing has become more and more important since the late 1960's. Although its first uses were mainly industrial, nowadays the field of readable representations of graph has applications in various domains such as medical science, software engineering, cartography, chemistry or natural language processing. Today, we not only try to draw planar graph but we want to draw them aesthetically and/or with certains geometrical properties.

The increasing interest in graph drawing has naturally draw the attention of many researchers, first on the planarity decision problem, and then on the planar embedding problem. Since then, lots of very different algorithms have been designed to solve these problems. The complexity of the algorithms has also been an issue, with many algorithms running in $O(n^2)$ time. Hopefully, some linear time algorithm have also been designed; the first one is due to Hopcroft and Tarjan. The variety of approaches has made planar drawing a very rich and flourishing field of research.

Because of this variety, we decided to focus on one particular technique to solve the planar decision problem. Then we gives a property of planar graphs drawing. Finally, we present one particular kind of planar embedding called orthogonal drawing.

2 Testing planarity

There exist various drawing standards to draw graphs. Vertices are usually represented by symbols like points or boxes and edges by curves. In this report, the vertices will be points and edges will be polilyne curves.

2.1 Drawing Planarity

We choose to present an algorithm to decide the graph planarity presented in [Wes96] because this algorithm is simple, relatively instinctive and was founded early (1964) by French people. This algorithm was first presented by Demoucron, Malgrange and Pertuiset.

This algorithm uses the notion of H-fragments which is following.

Definition 2.1.1 (H-fragments)
Let $G = (V, E)$ be a graph and $H = (V_H, E_H)$ a subgraph of G. An *H-fragment* is either:

- an edge not in H which endpoints are in H
- a connected component of $G \setminus V_H$ together with the edges (and the *vertices of attachment*) that connect it to H

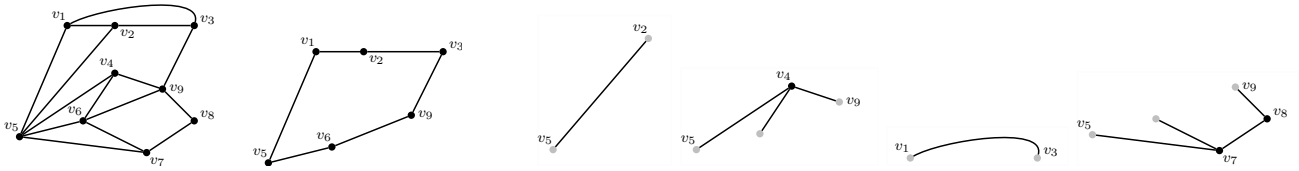


Figure 1: A complete graph G, with a subgraph H of G and its associated H-fragments

Definition 2.1.2 (conflict fragments)
Let be C a cycle in a graph G. Two C-fragment A a,d B *conflict*:

- if they have three common vertices of attachment to C or
- if there are four vertices v_1, v_2, v_3, v_4 in cyclic order on C such that v_1, v_3 are vertices of attachment of A and v_2, v_4 are vertices of attachment of B.

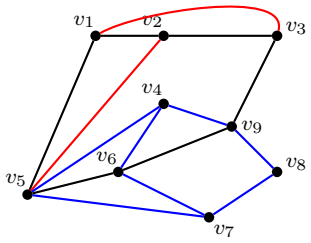


Figure 2: To kind of conflicting fragments

Definition 2.1.3
A path between two vertices of attachment, in a given face F, is called α-*path*.

2.1.1 Description of the algorithm of Demoucron et all

The main idea of this algorithm is to iteratively construct a planar embedding starting with a subgraph of G.

The algorithm uses the fact that if G is planar and if we have a planar embedding of a subgraph H of G, we can extend it to a planar embedding of G. We thus start with an H which is any cycle in G to ensure that the first H is planar. Then, we extend H iteratively as follows:

- we determine the H-fragments
- we select an H-fragment B and a face F which can accept B. A face accepts an H-fragment if and only if it contains all the vertices of attachment of B.
- We choose an *alpha*-path to embed B in F.

This algorithm works only with 2-connected graphs.

Definition 2.1.4
A *2-connected graph* G is graph which the minimum size of the set S of vertices such the $G \setminus S$ is disconnected or has only one vertex.

Definition 2.1.5
We call *block* a maximum subset of G which is 2-connected.

Thus, to know if a graph G is planar, we successively gives the block of G to the algorithm, after having computing this blocks. We can compute these blocks using a Depth-First-Search algorithm according to [Wes96].

Algorithme 1 : Testing planarity algorithm

Input : A 2-connected graph G
Output : true if G is planar, false otherwise

```
1  begin
2      if |E| ≥ 3|V| − 6 then
3          return false
4      else
5          H ← any cycle C of G
6          F ← inner and outer faces of C
7          while H ≠ G do
8              B ← all H-fragments of G
9              foreach B ∈ B do
10                 F(B) ← {F ∈ F | F contains all vertices of attachment of B}
11             if ∃B, |F(B)| = 0 then return false
12             if ∃B, |F(B)| = 1 then B_0 ← B
13             else B_0 ← any B ∈ B
14             P ← any α-path P of B_0
15             H ← H ∪ P with P embedded into some F ∈ F(B_0)
16             F ← update faces of H
17         return true
18 end
```

2.1.2 Proof of the correctness of the algorithm

In this section, we prove the correctness of the algorithm.

Theorem 2.1.1
The algorithm of Demoucron, Malgrange and Pertuiset for the computation of a planar embedding of a graph is correct.

Lemma 2.1.1
If A and B are conflicting fragments such that $|F(A)| \geq 2$ and $|F(B)| \geq 2$ then $F(A) = F(B)$ and $|F(B)| = 2 = |F(A)|$.

Proof. (By contradiction)
Assume that $F(A) \neq F(B)$, then, there exist three distinct faces f, g, h such that:

- $f \in F(A)$
- $g \in F(B)$

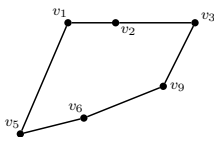
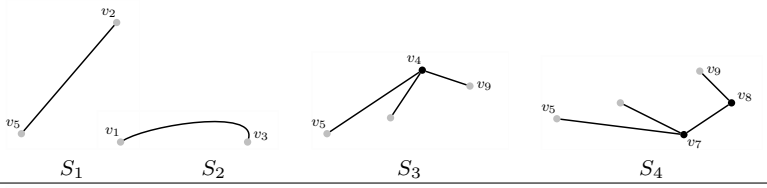
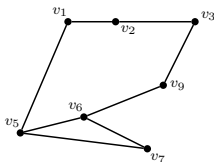
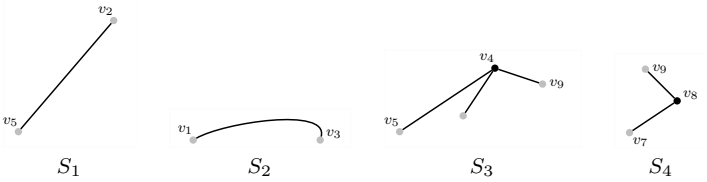
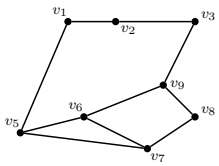
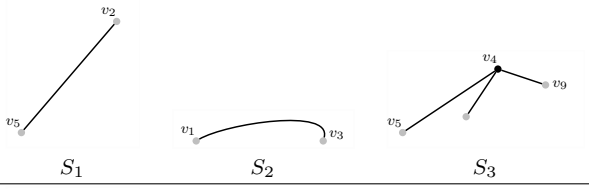
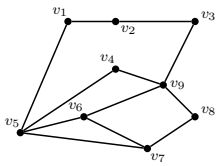
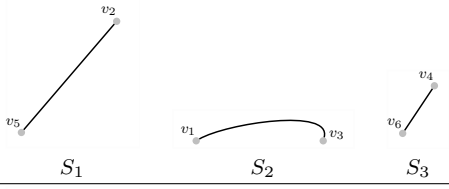
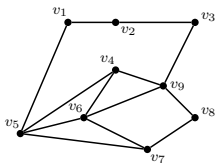
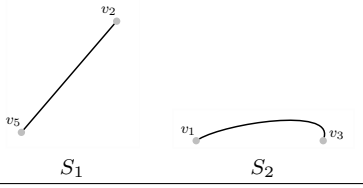
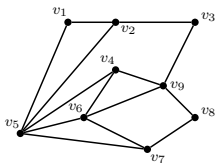
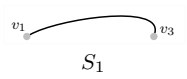
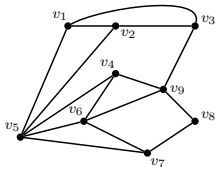
	$F_1 = 1234965$ $F_2 = 1234965$		$F(S_1) = 1, 2$ $F(S_2) = 1, 2$ $F(S_3) = 1, 2$ $F(S_4) = 1, 2$
	$F_1 = 123965$ $F_2 = 567$ $F_3 = 1239675$		$F(S_1) = 1, 3$ $F(S_2) = 1, 3$ $F(S_3) = 1, 3$ $F(S_4) = 3$ c
	$F_1 = 123965$ $F_2 = 567$ $F_3 = 6987$ $F_4 = 1239875$		$F(S_1) = 1, 4$ $F(S_2) = 1, 4$ $F(S_3) = 1$
	$F_1 = 123945$ $F_2 = 567$ $F_3 = 6987$ $F_4 = 5496$ $F_5 = 1239875$		$F(S_1) = 1, 5$ $F(S_2) = 1, 5$ $F(S_3) = 2$
	$F_1 = 123945$ $F_2 = 567$ $F_3 = 6987$ $F_4 = 465$ $F_5 = 496$ $F_6 = 1239875$		$F(S_1) = 1, 6$ $F(S_2) = 1, 6$
	$F_1 = 125$ $F_2 = 23945$ $F_3 = 567$ $F_4 = 6987$ $F_5 = 465$ $F_6 = 496$ $F_7 = 1239875$		$F(S_1) = 7$
	$F_1 = 125$ $F_2 = 23945$ $F_3 = 567$ $F_4 = 6987$ $F_5 = 465$ $F_6 = 496$ $F_7 = 123$ $F_8 = 139875$		

Table 1: Application of the algorithm

- $h \in F(A) \cap F(B)$, since A and B conflict.
- Each α-path $L \subset A$ is embeddable into face f
- Each α-path $M \subset B$ is embeddable into face g.

So, each pair (L, M) is embeddable outside face h and is also embeddable into h due to the starting assumption, which is a contradiction with the assumption that A and B conflict. So, because we ca not have three different faces, $F(A) = F(B)$. $|F(A)| = 2$ because we assume that $F(A)$ and $F(B)$ conflict, hence these fragments have each at least two vertices of attachment are on the same α-path, which is a boundary between two faces. These two faces are the only possible faces to embed fragment A. See the constructed figure 3. □

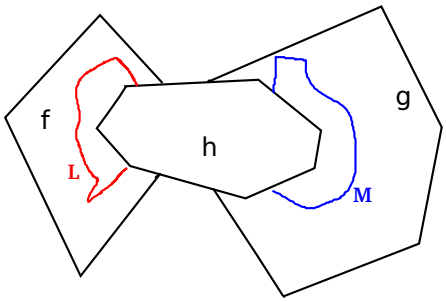


Figure 3: Faces with competing fragments

To justify that the algorithm is actually an incremental algorithm, we use the notion of *partial embedding*.

Definition 2.1.6
An embedding of H is a *partial embedding* of a planar G if we can get this embedding from an embedding of G by removing edges and vertices.

Definition 2.1.7 (fragment graph)
The *fragment graph* $S(H)$ is a graph which vertices are H-fragments. There is an edge between any pair of conflicting fragments.

Lemma 2.1.2 (the fragment graph are bipartite)
If H is a partial embedding of G and if $|F(A)| \geq 2$ for all fragments then $S(H)$ is bipartite.

Proof. (by contradiction)
Assume that $S(H)$ is not bipartite. Then there is a cycle $C = (fr_1, \ldots, fr_n)$ of odd length (thus n is even). By applying lemma 2.1.1 to fr_i and fr_{i+1} we get that $F(fr_i) = F(fr_{i+1})$ and there are exactly 2 admissible faces F_1 and F_2 for all the fragments in cycle C. Thus the only way to embed the fragments is to put the α-paths $P_i \subset fr_i$ into F_1 when i is odd and into F_2 when i is even (or vice-versa). But, fr_n has an even i and $fr_n = fr_1$ and fr_1 has an odd i. Thus we cannot embed any more paths into H, so H is not a partial embedding of G. □

Theorem 2.1.2 (each step of the algorithm produces a partial embedding)
If G is planar, each iteration of the algorithm produces a partial embedding H.

Proof. (by induction on the number of iterations)
Let n be the number of iterations.

- **Base case**: $n = 1$.

  H is a cycle of G and can be reconstructed by removing edges and vertices from any planar embedding of G.

- **Induction from G_n to G_{n+1}.**

  Assuming that we start with a planar embedding of H, let analyze the behavior of the algorithm. As G is planar, there are admissible faces for all H-fragments. Otherwise, there would be an α-path in a fragment connecting different faces of the partial embedding, which would be a partial embedding of a non-planar embedding of G.

  The first phases of the algorithm computes the set of the fragments to allow. The partial embedding G_{n+1} is build during the steps 12-13-14. There are two cases which are analyzed as follows.

  1. There is a fragment A with only one admissible face. In this particular case, there is only one way to embed an α-path $P \subset A$, so the algorithm produces a partial embedding $H \cup P$.
  2. All fragments have more than one admissible face. We have to choose any α-path $P \subset A$, for any G_n-fragment A. There are now two cases to consider:
     - we choose the same face as in the planar embedding of G; we still have a partial embedding of G
     - we choose the other alternative; but the fragment graph $S(G_n)$ is bipartite (see lemma 2.1.2) and there only are two admissible faces f and g for all fragments in the connected component of A. So no fragment in f conflicts with a fragment in g. Then we can swap these two faces in the planar embedding of G and get a new planar embedding of G, of which H is a partial embedding.

□

2.1.3 Complexity

The algorithm presented in section 2.1.1 runs in polynomial time, $O(n^2)$.

initialization First, we have to compute the different blocks of G. Using a Depth-First-Search, we can compute them in $O(n)$.

Then, the initialization requires us to find a cycle in the graph. In the general case, finding a cycle has a complexity of $O(n + e)$. In our case, since $|E| < 3|V| - 6$, the complexity is reduced to $O(n)$. We consider here that we dispose of the following data structures:

- an adjacency list to store $G = (V, E)$
- a matrix $n * n$ (where $n = |V|$), which stores the edges of H. We do a transversal of this matrix only once, to initialize it.
- a table of sorted lists to store the name of the faces of each vertex in G.

Number of iteration
Let first remind Euler's theorem .

Theorem 2.1.3 (Euler)
If a connected plane graph G has exactly n vertices, e edges, and f faces, then $n - e + f = 2$.

Thanks to the previous theorem, we deduce that the number of iterations is exactly equal to $|E| - |V| + 1$, and so it is less than $2|V| - 5$, since $|E| \leq 3|V| - 6$. Indeed, at each iteration, the algorithm add a face to the drawing. So, the number of iteration is the number of faces of the graph minus one, because the algorithm starts with one face already embedded.

Complexity of one iteration
Each step of the algorithm has a complexity no more than $O(n)$:

- The step concerning the computation of the H-fragments can be achieved in $O(n)$. Indeed, it amounts to compute the component of the graph, and it can be achieved in one graph transversal, so in $O(n)$. We have to store the vertices of attachment of each H-fragment.
- Finding the potential faces of the H-fragments consists in doing a traversal of the vertices of attachment of each H-fragment and do the intersection of faces containing each of these fragments. We can find the result of the intersection in $O(n)$, doing a traversal of each attachment vertex and of their list of faces, conserving the result in a temporary list of faces. If this list does contain a face which does not appear in the list of one attachment vertex, we remove this face of the temporary list. As explained previously, this list can not be longer as 2 faces after having checked two vertices of attachment.
- the two tests can be achieved in $O(n)$ because the number of fragments is less than n and they have each at most 2 faces.
- finding a path is achieved in one traversal.
- adding a path to H amount to add $|P| \leq n$ edges in the matrix storing H. Each addition is achieved in $O(1)$.
- To find the new constructing faces, we only need to split the previous existing face into two subfaces. This can be done in less than $O(n)$ time.

3 Straight-line drawing

A *straight-line drawing* of a planar graph is a drawing in which each edge is drawn as a straight-line segment without crossing edges.

Theorem 3.0.4 (Fary)
Every planar graph has a straight line embedding.

To prove this theorem, it is sufficient to prove it on maximal connected planar graphs, in which each face is a triangle. Indeed, if it is not the case, we can add dummy edges to complete the graph and work on a maximal connected graph. And then, remove the dummy edges does not change the properties of the planar graph.

Proof. By induction on the number n of vertices in graph G.

Let assume that we have a maximal connected planar graph G.

Base case: $n = 3$, because the graph is maximal connected, we have 2 faces: the outer one and the internal one which is a triangle. This triangle can be drawn as a polygon with 3 sides. These sides are straight-lines.

Induction case: we assume that every planar graph with $n - 1$ vertices has a straight-line embedding. Let choose three vertices a, b, c be the outer face of the drawing of $G = (V, E)$, with $|V| = n$ and $|E| = m$. We will use the fact that there exists a vertex $v \in V \setminus \{a, b, c\}$ which has

strictly less than 6 incident edges. Indeed, the sum of the degrees of these three outer points are at least 7: $2 * 3$ outer-edges and at least an edge connected to the rest of the graph. Moreover, we saw in class that every planar graph has the following property: $3n - 6 \geq m$. It implies the following inequation: $6n - 12 \geq 2m$ and $2m = \sum_{v \in V} deg(v)$. Let call $restDeg$ the rest of the combined degrees of the other vertices $n-3$. Then $6n-12 \geq restDeg+7 \Leftrightarrow 6n-19 \geq restDeg$. It implies that there are at least one vertex $v \in V \setminus a, b, c$ with a degree at most 5.

Let remove such a vertex with degree $k = deg(v) \geq 5$, together with the k incident edges. As G is maximally connected, when we remove a vertex v and its incident edges. We obtain G', which we make maximal adding $k-3$ edges. Indeed, let x be the number of edges to add: $|E'| = |E| - k + x \Leftrightarrow x = |E'| - |E| + k \Leftrightarrow x = 3(n-1) - 6 - [3n-6] + k \Leftrightarrow x = k - 3$.

By induction, G' has a straight-line embedding in which a, b, c are the outer vertices. Remove the added $k-3$ edges of the straight-line embedding of G'. We obtain a polygonal face with at most 5 sides, in which we have to embed vertex v. Le present the Chvátal's Art Gallery theorem to conclude.

Theorem 3.0.5 (Chvátal's Art Gallery Theorem)
Let P be a polygon with r internal angles of P greater than π formed at vertex v, $r \geq 1$. Then r guardians are always sufficient and occasionally necessary to guard P.

Because of this theorem, each pentagon needs at most one guardian to guard P, what means, that it is always possible to find a point in a pentagon from which we can draw a straight-line to join any point of the boundary of P. Then, it is possible to embed v in our polygon, such that we can join all the vertices belonging the polygon with a straight-line.

□

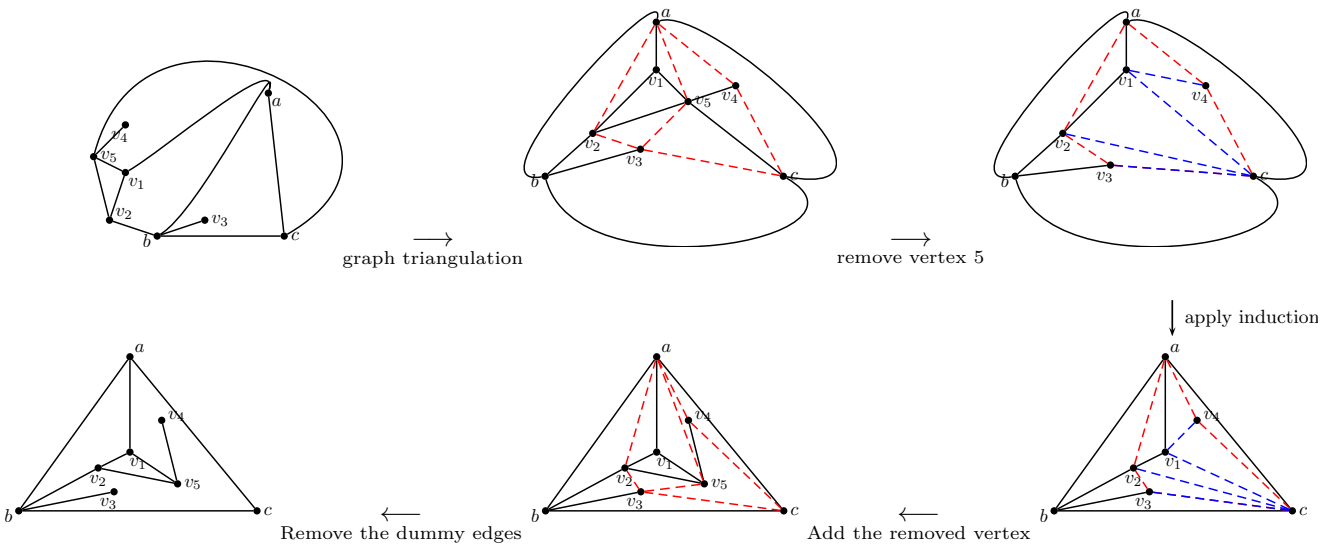


Figure 4: A planar can always be drawn with non crossing straight-line edges.

Until now, an algorithm to find a non-specific drawing of graph G has been presented and we know that each planar graph admits a straight-line drawing. Nonetheless, we sometimes need to give some additional constraints to draw planar graphs, and especially in the field of circuits conception. For instance, we can require a drawing with polyline edges which are horizontal or vertical segments series. We call such representations **box-orthogonal drawing**. In such

a drawing, each face is a rectilinear polygon. Some particular drawing are not always possible for any planar graph. Let present in the following section the main idea for drawing such representations of planar graphs.

4 Specific drawing: orthogonal drawing

As for many planar graph drawing, there exists a lot of distinct algorithms to find a box-orthogonal drawing for planar graph. Some are linear, some are polynomial. One of these algorithm reduces the orthogonal drawing in a flow network model. The goal of this algorithm is to draw the orthogonal drawing with a minimum of changes of direction for the edges. In this aim, they call an algorithm of minimum flow cost problem (see the report about the minimum cost flow).

Let present here how they reduce the orthogonal drawing problem into a minimum flow cost problem, and how they model the orthogonal drawing.

4.1 Basic facts

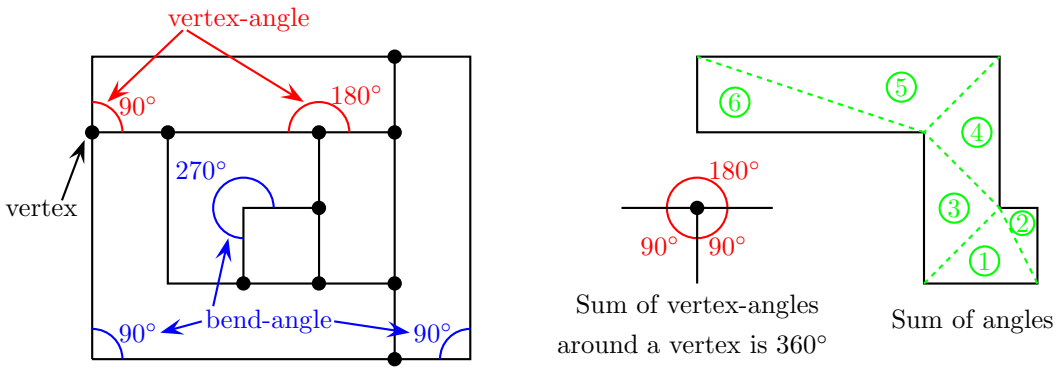


Figure 5: Example of an orthogonal drawing with notations and facts

There are two kinds of angles:

- a *vertex-angle* is an angle formed by two edges incident to a vertex
- a *bend-angle* is an angle formed by two line segments of an edge

Figure 5 sums up the definitions and gives an example of an orthogonal drawing of a planar graph. Clearly, all angles are $k \cdot 90°$ for some $k \in \{1, 2, 3, 4\}$.

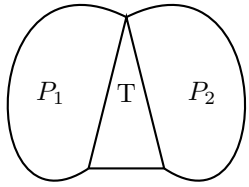
Now notice the two following facts about angles:

Lemma 4.1.1 (Sum of vertex-angles)
The sum of vertex-angles around any vertex is 360°.

Proof. See figure 5. □

Lemma 4.1.2
The sum of the angles inside any polygonal face is $(2p-4)90°$, and the sum of the angles of the outer polygonal face is $(2p+4)90°$, where p is the number of angles of the polygon.

Proof. Consider a division of a polygon in r non-overlapping triangles $T_1, \ldots, T_r$ such as in figure 5. Clearly the sum Σ of all angles inside the polygon is the sum of angles inside the triangles. But the sum of all angles inside a triangle is $180°$. So $\Sigma = r \cdot 180°$. Let call p the number of angles of the polygon. Then the polygon has p sides and it can inductively be shown that $r = p - 2$:

$D =$ 

$$P_1 \begin{cases} p_1 & \textit{sides} \\ p_1 - 2 & \textit{triangles} \end{cases} \qquad P_2 \begin{cases} p_2 & \textit{sides} \\ p_2 - 2 & \textit{triangles} \end{cases} \qquad P \begin{cases} p = (p_1 - 1) + (p_2 - 1) - 1 & \textit{sides} \\ p - 2 = p_1 + p_2 - 1 & \textit{triangles} \end{cases}$$

So finally $\Sigma = (p-2)180° = (2p-4)90°$.

Consider an outer polygon P. Now consider it's complement $\bar{P}$. Then $\bar{P}$ is a polygon. Furthermore, both P and $\bar{P}$ have p angles. Now notice that if P has an angle at point (x, y) of value α, then $\bar{P}$ has an angle at point (x, y) of value $360° - \alpha$. So we get that $\Sigma(P) = 360° \cdot p - \Sigma(\bar{P})$. Now apply the previous result to P and we get that $\Sigma(\bar{P}) = (p-2)180°$ which yields:

$$\Sigma(P) = 360° \cdot p - (p-2)180° = (p+2)180° = (2p+4)90°$$

□

4.2 Orthogonal representation

We now introduce an *orthogonal representation* of an orthogonal drawing D in terms of *bends* occurring along edges and angles formed by edges. Assume that for each face F, we already have an ordered list $P(F) = (e_{i_1}, \ldots, e_{i_k})$ of edges forming the face F (see figure 6 for an example). An *orthogonal representation* R of D is a set of circular ordered lists $R(F)$, one for each face F of D. Each element r of $R(F)$ is a triple (e_r, s_r, a_r) where:

- e_r is an edge of F
- s_r is a bit string providing information about the bends along e_r: the k^{th} bit of s_r describe the k^{th} bend on the **right side** of e_r, 0 indicates a $90°$ bend and 1 indicates a $270°$ bend. An empty string ε means there is no bend, so it indicates a line segment
- $a_r \in \{90°, 180°, 270°, 360°\}$ indicates the angle between e_r and $e_{r'}$ where r' is the element following r is the circular list $R(F)$

Clearly, R describes exactly **the shape of D without considering lengths of line segments** and hence describes an equivalent class of orthogonal drawings of G with "similar shape". Conversely, the following theorem gives necessary and sufficient conditions for a set R to actually describe an orthogonal drawing.

Theorem 1
A set R of circular lists as described above is an orthogonal representation of an orthogonal drawing D of G **if and only if** the following conditions hold:

(1) There is some planar graph which planar representation is given by the e-fields of the lists in R

(2) For each pair of element $r, r' \in R$ with $e_r = e_{r'}$, $s_r = \overline{rev(s_{r'})}$, that is the string s_r can be obtained by applying bitwise negation to the reversal of $s_{r'}$

(3) For each element r, define the rotation $\rho(r)$ as follows:

$$\rho(r) = |s_r|_0 - |s_r|_1 + \left(2 - \frac{a_r}{90^\circ}\right)$$

where $|s_r|_i$ is the number of i in s_r. Then

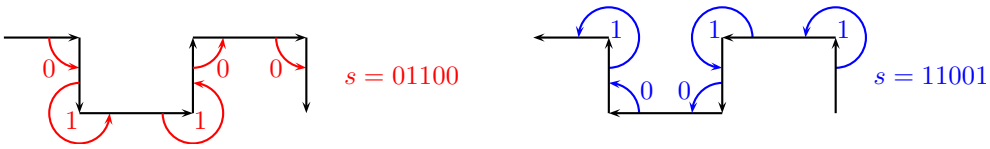
$$\sum_{r \in R(F)} \rho(r) = \begin{cases} +4 & \text{if } F \text{ is an inner face} \\ -4 & \text{if } F \text{ is the outer face } F_0 \end{cases}$$

(4) For each vertex v, the sum of the vertex-angles around v given by the a-field in R is equal to 360°

Proof. $\Rightarrow$ First assume that r is an orthogonal representation of an orthogonal drawing D.

(1) This is trivial since D is an orthogonal drawing and thus G must be planar

(2) Let $r, r' \in R$ with $e_r = e_{r'}$. Then one of them must be described in clockwise order and the other in counter-clockwise order, so to obtain s_r from $s_{r'}$, one must reverse the bits. Furthermore, a 90° bend in one order now become a 270° in the other order so one must apply bitwise negation.



(3) Let F be a face with p angles (and thus p faces) and v vertices then there are v vertex-angles and $p - v$ bend-angles, so:

$$\begin{aligned} \sum_{r \in R(F)} \rho(r) &= \sum_{r \in R(F)} \left(|s_r|_0 - |s_r|_1 + \left(2 - \frac{a_r}{90^\circ}\right)\right) \\ &= \#\{90^\circ \text{ bend-angles in } F\} - \#\{270^\circ \text{ bend-angles in } F\} \\ &\quad + 2v - \frac{\sum\{\text{vertex-angles in } F\}}{90^\circ} \end{aligned}$$

But remember that following from lemma 4.1.2 we have:

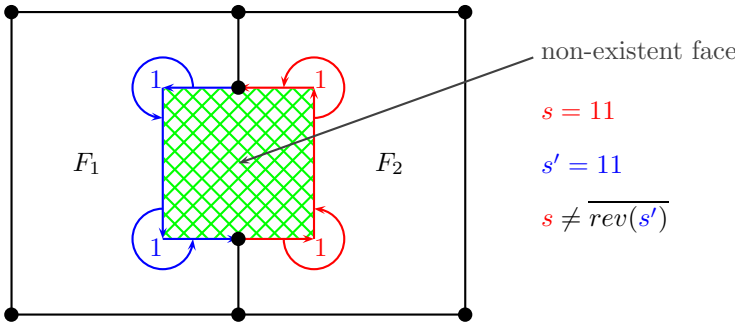
$$\begin{aligned} 2p + 4\delta &= \frac{\sum\{\text{angles in } F\}}{90^\circ} \\ &= \frac{\sum\{\text{vertex-angles in } F\} + 90^\circ \#\{90^\circ \text{ bend-angles in } F\} + 270^\circ \#\{270^\circ \text{ bend-angles in } F\}}{90^\circ} \\ &= \frac{\sum\{\text{vertex-angles in } F\}}{90^\circ} + \#\{90^\circ \text{ bend-angles in } F\} + 3\#\{270^\circ \text{ bend-angles in } F\} \\ &= 2v + 2(p - v) + 4\delta \end{aligned}$$

where $\delta = +1$ if F is the outer face and $\delta = -1$ otherwise. Thus:

$$\begin{aligned}\sum_{r \in R(F)} \rho(r) &= -2(p-v) - 4\delta + 2\#\{90^\circ \text{ bend-angles in } F\} + 2\#\{270^\circ \text{ bend-angles in } F\} \\ &= -2(p-v) - 4\delta + 2\underbrace{\#\{\text{bend-angles in } F\}}_{=p-v} \\ &= -4\delta\end{aligned}$$

(4) This is a direct consequence of lemma 4.1.1

$\Leftarrow$ Assume that properties (1) to (4) are satisfied by R. Then (1) implies that the faces described in R are the faces of the planar graph G. Furthermore, (3) implies that each described face is rectilinear, so putting things together, it means that each face of G is rectilinear. Property (2) ensure that the faces are consistent, i.e. there is not "hole". Finally, property 4 ensures the faces fill the entire space and are non-overlapping.



□

To find an orthogonal drawing, a method which is used is to reduce this problem into a flow network problem. That is why the problem of flow networks is shortly described in section 4.2.1 and in section 4.2.2, we explain how to actually reduce an orthogonal drawing into a network flow problem.

4.2.1 Flow network

As it will be presented in more details in an other report, a flow network is a directed graph $N = (V, E)$ such that N has to disjoint non-empty sets of distinguished nodes called its *sources* and *sinks*. Each arc e of N is labeled with three nonnegative integers:

- a lower bound $\lambda(e)$
- a capacity $\mu(e)$ and
- a cost $c(e)$

For each node u which neither a source nor a sink, the sum of all the flows of the incoming edges must be equal to the sum of the flows of all the outgoing edges (see figure 7).

Definition 4.2.1 (Production and Consumption)
Each node has a *production* $\sigma(v) \geq 0$ and a *consumption* $-\sigma(v) \geq$ associated to a flow. Only the sources and the sink have respectively a consumption or a production equal to 0.

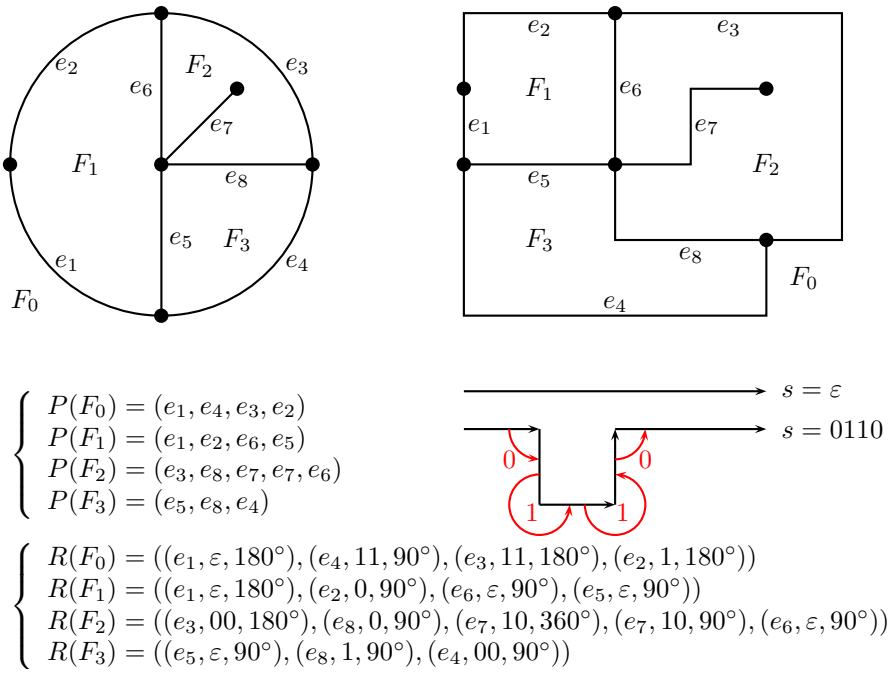


Figure 6: Example of an orthogonal representation

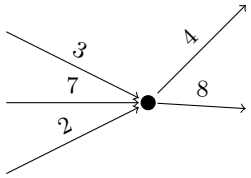


Figure 7: Incoming and outgoing flows of a vertex v: 3+7+2= 4+8= 12

Definition 4.2.2 (Flow)
A **flow** ϕ in $\mathcal{N}$ associates a nonnegative integer $\phi(e)$ with each arc e, which is the flow of arc e. The flow of each arc e must satisfy the following inequality: $\lambda(e) \leq \phi(e) \leq \mu(e)$.

Definition 4.2.3 (Cost)
We denote the **cost** of a flow ϕ: $COST(\phi)$.
$COST(\phi) = \sum_{e \in E} c(e)\phi(e)$

The flow problem states as follows. Given a network N, find a feasible flow ϕ, such that all the conditions described above are satisfied.

4.2.2 Finding an Orthogonal Drawing

In this section, we explain how to transform a drawing orthogonal problem, described in section 4.2, into a flow problem. Each feasible flow is a solution for an orthogonal drawing.

We first build another graph G_f which gives the informations concerning the nodes of G and their place in the faces of the drawing.

We introduce two types of vertices.

- vertices in U_V which represent the set of vertices of G. The production of each nodes in U_V represents the number of wires which can be linked to these nodes. Thus their production is $\mu = 4$. This type of vertices are *sources*.
- vertices in $U_{\mathcal{F}}$ represent the faces of the drawing. The consumption of vertices in $U_{\mathcal{F}}$ represents the number of bends they can admit for their boundary, thus $\mu = +\infty$. This kind of vertices are called *sinks*.

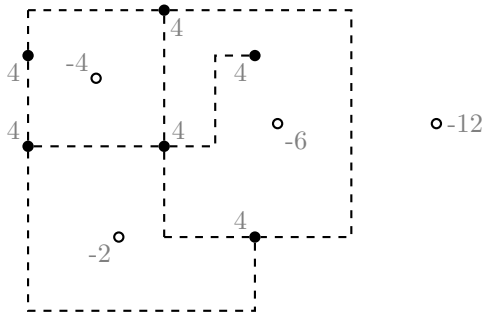


Figure 8: Graph of nodes and faces

We also introduce two kind of edges.

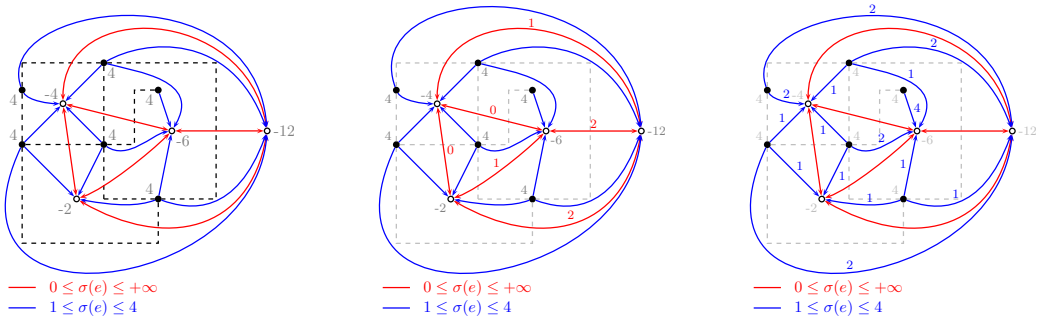
- the edges of type (u_v, u_F) which link a vertex in U_V and a vertex in $U_{\mathcal{F}}$: we denote this set of edges E_V. These edges mean that vertex v belongs to face F. The flows of this kind of edges corresponds to the sum of all the angles formed by the incident edges of v inside face f.
- the edges of type $(u_F, u_{F'})$ which link to vertices in $U_{\mathcal{F}}$. We denote this set $E_{\mathcal{F}}$. The flow of these edges represents the number of bends separating face F and face F'.

Because of lemma 4.1.1, each vertex $v \in U_V$ has a production $\sigma(v) = 4$. Because of lemma 4.1.2, each vertex $v \in U_{\mathcal{F}}$ has a consumption

$$-\sigma(v) = \begin{cases} 2p(F) - 4 & \text{if } F \text{ is an inner face} \\ 2p(F) + 4 & \text{if } F \text{ is the outer face } F_0 \end{cases}$$

This definition satisfies the flow conditions. Clearly, the total production is $4n$, with n the total number of vertices in G to be orthogonalized. The total consumption can be expressed as follows:

$$\begin{aligned} \sum_{F \neq F_0} (2p(F) - 4) + 2p(F_0) + 4 &= \underbrace{2p(F_0) + \sum_{F \neq F_0} (2p(F))}_{4e} - \sum_{F \neq F_0} 4 + 4 \\ &= 4e - (f-1) * 4 + 4 \\ &= 4e - 4f - 8 \\ &= 4n \qquad \text{according to Euler's formula} \end{aligned}$$



(a) Production and consumption of nodes and faces, with their associated edges

(b) Flows corresponding to the number of bends between faces

(c) Flows corresponding to edges in E_V

Figure 9: Construction of the flow network

Thus, the sum of consumption is equal to the sum of production.

Theorem 4.2.1
Let G be a plane graph and $\mathcal{N}$ the network constructed from G. For each integer flow σ in network $\mathcal{N}$, there is an orthogonal representation R that represents an orthogonal drawing D of G and whose number of bends is equal to the cost of the flow σ. In particular, the minimum cost flow can be used to construct a bend-optimal orthogonal drawing of G. (see [TS04] for more explanation)

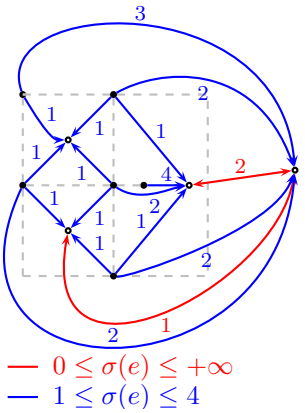


Figure 10: An optimal orthogonal representation

5 Conclusion

As already suggested in introduction, graph drawing is an important field which has a lot of application in distinct areas of research. Planar graph drawing was first motivated by the field

involving circuits building. We saw that a straight line drawing is always possible for a planar graph. Indeed, crossing wires are not advised to get good results. Specific planar graph drawing such as orthogonal drawing are easy to use in practice even if they are not always possible.

References

[AXE] KOHNERT AXEL. Algorithm of demoucron, malgrange, pertuiset.

[Ede] Herbert Edelsbrunner. 23 planar graphs. Lecture from Duke University of Computer Science.

[MK] Puneet Maheswari and Nagendra Kamath. Planarity testing. `www.utdallas.edu/~nkk071000/combinatorics/planarity.ppt`.

[TS04] Nishizeki Takao and Rahman Saidur. *Planar Graph Drawing*, volume 12. New Jersey ; London ; Singapore, [etc.] : World Scientific, 2004.

[Wes96] D. B. West. *Introduction to Graph Theory*. Prentice-Hall, 1996.