3. Triangulations

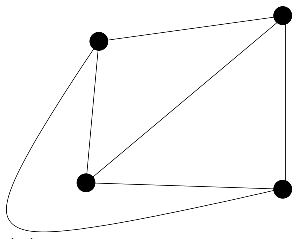
Lecture on Monday 28th September, 2009 by Bernd Gärtner <gaertner@inf.ethz.ch>

3.1 Planar and Plane Graphs

A graph is a pair $G = (V, E)$ where V is a finite set of *vertices* and E is the set of *edges*, $E \subseteq \binom{V}{2} := \{\{v, v'\} \mid v, v' \in V, v \neq v'\}$.

A *drawing* of a graph G is obtained by identifying vertices with (distinct) points in $\mathbb{R}^2$ and edges with simple Jordan arcs that connect their two vertices.

A graph is *planar* if there is a drawing of it such that no two edges cross in their interior. Such a drawing is also called an *embedding* of the graph. For example, K_4 (the complete graph on 4 vertices) is planar, see Figure 3.1 (left).


(a) crossing-free drawing (embedding) of K_4

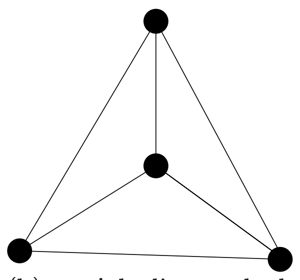

(b) straight-line embedding of K_4

Figure 3.1: *Planar graphs*

It can be shown that the graph K_5 is not planar. If a graph is planar, then there also exists a drawing in which all arcs are line segments.We call this a *straight-line embedding*. In order to get such a drawing for K_4, we have to put one vertex into the convex hull of the other three, see Figure 3.1 (right).

A *plane graph* is an embedding of a planar graph. Both graphs in Figure 3.1 are plane graphs. Sometimes, we will encounter straight-line embeddings with half-infite edges (having a vertex only on one side). On the graph side, this can be handled with an "infinite vertex" that is incident to all half-infinite edges. In the geometric setting, this is often not even necessary, if we generalize the concept of an edge to allow half-infinite rays.

3.2 The Euler Formula

A plane graph has vertices and edges (the arcs) but also *faces* (the connected components of the complement of the drawing). Both plane graphs in Figure 3.1 have 4 vertices, 6 edges and 4 faces. In general, if $|V|$ is the number of vertices of a *connected* plane graph,

$|E|$ its number of edges and $|F|$ the number of faces, then the Euler Formula states that

$$|V| - |E| + |F| = 2.$$

In the example, we get $4 - 6 + 4 = 2$.

If you don't insist on being too formal, the proof is simple and works by induction over the number of edges. If we fix the number of vertices, the base case occurs for $|V| - 1$ edges where the plane graph is a tree. Then we have $|F| = 1$ and the formula holds. A graph with more edges always contains a cycle and therefore at least one bounded face. Choose one edge from a bounded face and remove it. The resulting graph is still connected and has one edge less but also one face less since the edge removal merges the bounded face with another one. Consequently, since the Euler Formula holds for the smaller graph by induction, it also holds for the larger graph.

The Euler Formula can be used to prove the following important fact about planar graphs (exercise).

Lemma 3.1 *A planar graph with* n *vertices has at most* $3n - 6$ *edges (and* $2n - 4$ *faces).*

3.3 The Doubly-Connected Edge List

Many algorithms in computational geometry work with plane graphs, and in particular with straight-line embeddings. The *doubly-connected edge list* (DCEL) is a data structure for representing a straight-line embedding of a graph in such a way that it can easily be traversed and manipulated. Let's only discuss connected graphs here, this will suffice for the time being.

The main building block of a DCEL is a list of *halfedges*. Every actual edge is represented by two halfedges going in opposite directions, and these are called *twins*, see Figure 3.2.

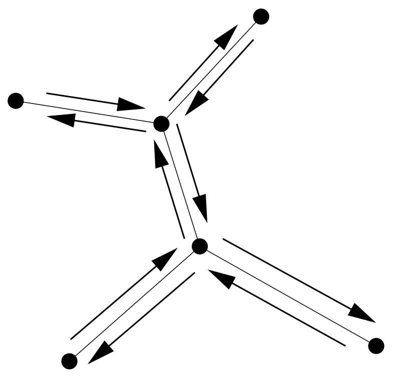


Figure 3.2: *Halfedges in a DCEL*

Within each face, the direction of halfedges is counterclockwise around the face.

The DCEL stores the list of halfedges, the list of vertices (with their coordinates), and the list of faces. These lists are interconnected by various pointers. A vertex v stores a pointer to an arbitrary halfedge originating from v, and a face f stores a pointer to an arbitrary halfedge within the face. A halfedge e stores *five* pointers: one to its twin, one to its predecessor in its face, one to its successor in its face, one to the vertex it originates from, and one to the face that it is contained in.

This information is sufficient for most tasks. For example, traversing all edges around a face can be done by going from the face to the halfedge it points to, and then following the successor and predecessor pointers. In case of a face that contains half-infinite edges, we may actually have to go in both directions in order to explore the whole face.

Traversing all edges incident to a vertex can be done by going from the vertex to the halfedge it points to, and then repeatedly going to the successor of the twin of the current halfedge.

The whole DCEL needs storage proportional to $|V|+|E|+|F|$ which is $O(n)$ for a plane graph with n vertices by Lemma 3.1.

3.4 Triangulations

Definition 3.2 *A* triangulation *is a straight-line embedding of a connected graph (the graph* underlying *the triangulation) with the property that every bounded face is a triangle.*

Figure 3.3 shows three triangulations. The second one has the specific property that all vertices are incident to the unbounded face. The third one has the even more specific property that all vertices are in convex position (meaning that they all appear on the convex hull of the set of vertices).

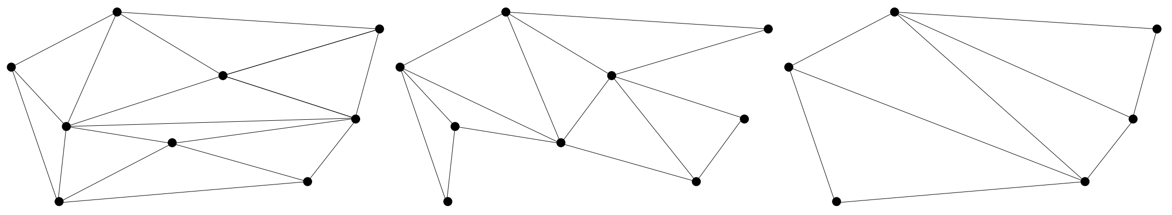

(a) point set triangulation (b) simple polygon triangulation (c) convex polygon triangulation

Figure 3.3: *Three triangulations*

In all three cases, there are many other triangulations with the same set of vertices and the same outer face. But as it turns out, every triangulation with a fixed set of vertices and a fixed outer face has the same number of triangles. This is another consequence of the Euler Formula.

3.5 Convex Polygon Triangulations

Let's take a closer look at case (c) in Figure 3.3. A triangulation whose set of vertices P is in convex position is called a *convex polygon triangulation*. We also say that we have a triangulation *of* the convex polygon formed by the convex hull of P.

In this case, the triangle counting is really easy: if there are n vertices, there will be $n-2$ triangles. But there is more we can do: it is possible to give an explicit formula for the number of triangulations of a convex n-gon, and it is possible to quickly compute a triangulation that is optimal w.r.t. a given measure from a large family of measures.

3.5.1 Number of convex polygon triangulations

Counting the number of point set triangulations that a set of n points can have in the worst case is an open problem. There are reasonable upper and lower bounds, but there is a substantial gap. For conex polygon triangulations, this is easy, as we show next.

Let's do some small cases first. If $n = 3$, there is exactly one triangulation, and for $n = 4$, we have two. This is due to the fact that a convex quadrilateral has two diagonals, and exactly one of them has to be chosen to get a triangulation, see Figure 3.4.

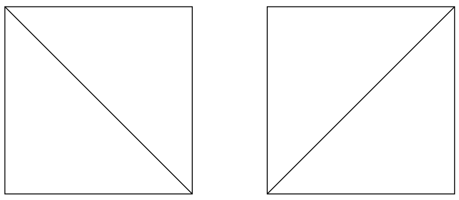


Figure 3.4: *The two triangulations of a convex 4-gon*

The fact that we have drawn a regular 4-gon is meant to indicate that the exact shape of the n-gon doesn't matter. Any convex n-gon can be transformed into a regular n-gon without changing the number of triangulations.

Given this, let's define p_n to be the number of triangulations of a convex n-gon. Somewhat arbitrarily, we set $p_2 = 1$ and let p_i remain undefined for $i < 2$.

To determine p_n, we do the following: take the convex n-gon and fix one edge of it (the *base edge*). In every triangulation, this edge must be part of exactly one inner triangle, where the third vertex of this triangle can be any of the other $n-2$ vertices, see Figure 3.5.

In how many ways can we complete each of these $n-2$ pictures to a full triangulation? Adding up the resulting $n-2$ numbers gives us the total number p_n of triangulations that we are looking for. Let's consider the k-th picture. Removing the triangle that contains the base edge leaves a $(k+1)$-gon to the left and an $(n-k)$-gon to the right. In Figure 3.5, we have a 2-gon (just an edge) and a 5-gon in the first picture, a 3-gon and a 4-gon in the second picture, a 4-gon and a 3-gon in the third picture, and a 5-gon and a 2-gon in the fourth picture.

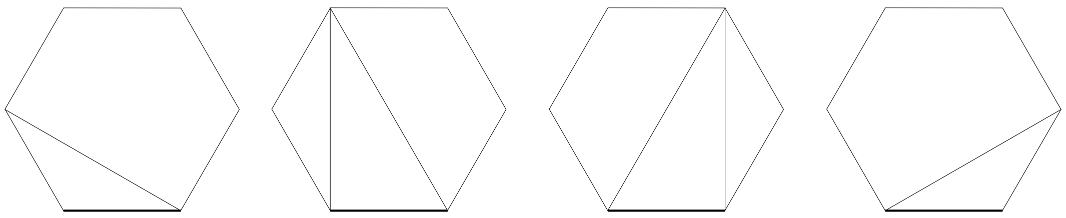


Figure 3.5: *The $n-2$ possible triangles containing the base edge*

Consequently, the k-th picture can be completed in $p_{k+1}p_{n-k}$ ways, since we can combine every triangulation of the $(k+1)$-gon with every triangulation of the $(n-k)$-gon (here it pays off that we defined $p_2 = 1$). This gives us the following result.

Theorem 3.3 *The number p_n of triangulations of a convex n-gon is given by the recurrence*

$$p_n = \sum_{k=1}^{n-2} p_{k+1}p_{n-k}, \quad n \geq 3,$$

with $p_2 = 1$.

Let's check some small values: we get

$$\begin{aligned} p_3 &= p_2p_2 = 1, \\ p_4 &= p_2p_3 + p_3p_2 = 2, \\ p_5 &= p_2p_4 + p_3p_3 + p_4p_2 = 5, \\ p_6 &= p_2p_5 + p_3p_4 + p_4p_3 + p_5p_2 = 14. \end{aligned}$$

Earlier we claimed that there is an explicit formula for p_n, so what is it? We will come back to this question later.

3.5.2 Optimal convex polygon triangulations

You might argue that counting the number of triangulations is not that interesting in itself, but as it turns out, we can use exactly the same approach to actually compute good triangulations. Here is the setup: let us consider a function μ that assigns a quality $\mu(\Delta)$ to each of the possible $\binom{n}{3}$ triangles formed by the n points of our n-gon. $\mu(\Delta)$ could for example be the sum of side lengths, or the largest angle.

Note that *now* the shape of the n-gon does matter, since the quality of a triangle may depend on its geometry.

Minimizing the total measure. Here is one optimization problem that we can consider (see the exercises for a different one that is also solvable by the same means). We want to compute a triangulation of a given n-gon that minimizes the sum of μ-values of all its triangles. If $\mu(\Delta)$ is the sum of side lengths, for example, it is easy to see that the optimum is achieved by a triangulation that minimizes the total edge length (a so-called *min-weight triangulation.* Finding an optimal *point set triangulation*, for a given set of n points, is a hard problem in general (in particular under the min-weight criterion). But if the points are in convex position, it's again easy.

To solve this optimization problem, we could go through all triangulations in a brute-force manner, for each one compute the sum of μ-values, and then output the best triangulation that we have seen. But we will see later that the numbers p_n from Theorem 3.3 grow exponentially in n, meaning that this is not a practical approach.

Instead, we employ *dynamic programming*, based on the fact that taking out the triangle over the base edge decomposes the problem into two *independent* smaller problems. Let's go back to Figure 3.5, but this time we label the vertices from 1 to n, starting and ending at the base edge, see Figure 3.6.

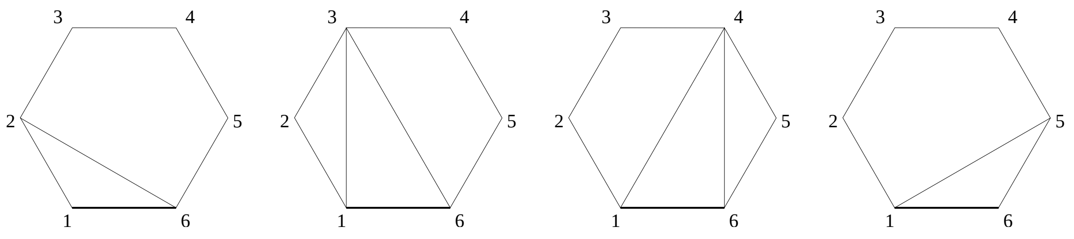


Figure 3.6: *The* $n-2$ *possible triangles containing the base edge, with vertices labeled*

Now fix $1 \leq i < j \leq n$ and define M_{ij} as the total measure of the best triangulation of the convex $(j-i+1)$-gon induced by the vertices with labels $i, i+1, \ldots, j$. "Best" is defined with respect to the sum of measures of the triangles.

Then M_{1n} is the quantity we are interested in: the smallest total measure that we can achieve in some triangulation of the original n-gon. We also know that the best triangulation must contain one of the $n-2$ possible triangles Δ over the base edge. To get the best triangulation for a fixed Δ, we add up $\mu(\Delta)$ and the total measures of the best triangulations to the left and to the right of Δ. Here we use the fact that these two subproblems are independent.

Therefore,

$$M_{1n} = \min_{k=2}^{n-1} (\mu_{1kn} + M_{1k} + M_{kn}),$$

where μ_{1kn} is the measure of the triangle spanned by the points with labels $1, k, n$. Replacing $1, 2, \ldots, n$ with $i, i+1, \ldots, j$, we obtain the general formula:

$$M_{ij} = \min_{k=i+1}^{j-1} (\mu_{ikj} + M_{ik} + M_{kj}). \tag{3.4}$$

If $j-i = m$, then $k-i, j-k < m$, so we reduce a problem of size m to two smaller problems. This allows the following dynamic programming approach: for $m =$

$1, 2, \ldots, n-1$, compute (and store) all values M_{ij} for which $j - i = m$. For $m = 1$, this is easy (the value is always 0, since we have just a 2-gon), and for every larger value of m, compute M_{ij} in time $O(m)$ through (3.4), just looking up the already known values M_{ik}, M_{kj}.

Since there are $O(n^2)$ pairs i, j that need to be considered, and we need time $O(n)$ to compute M_{ij} for each of them, we get the following result.

Theorem 3.5 *Given a convex n-gon and an arbitrary measure μ on triangles. We can compute a triangulation of the n-gon with smallest (or largest) total measure (sum of measures of all triangles) in time $O(n^3)$ and space $O(n^2)$.*

It is not too hard to extend this result to also deliver the best triangualation. During computation, we can for example always store the k that led us to the minimum. Then we can start with M_{1n}, find the correct splitting triangle through the corresponding value of k and recurse to the left and right to complete the triangulation to the optimal one.

Questions

6. *What are planar / plane graphs and straight-line embeddings?* Give the definitions and explain how straight-line embeddings can be represented on the computer.

7. *What is a triangulation?* Provide the definition and prove a basic property: every triangulation with the same set of vertices and the same outer face has the same number of triangles.

8. *How many convex polygon triangulations are there?* Derive a formula for the number of triangulations of a convex n-gon!

9. *How can you compute an optimal polygon triangulation?* Give an example of a reasonable quality measure, and describe an efficient (polynomial-time) algorithm for computing a triangulation with best the quality according to this measure. What are the key properties of your quality measure that make this algorithm work? Can you come up with a quality measure for which you can argue that your approach fails?