Planarity

Four stories
1. Connectivity
2. Planarity
3. Rectilinearity
4. Symmetry

But first: some administrative matters

Survey
Assignment 1 results
Assignment 3 hand out

Some administrative matters

The mid-semester survey
• PLEASE participate in our mid-semester survey in WebCT.
• PLEASE give feedback on the lectures, assignments and infrastructure
• PLEASE raise any Occupational Health & Safety issues.
• The survey is available on the WebCT page until next Tuesday
• PLEASE fill in this survey this week or early next week.

Assignment 1 results
• Available (on the web?)
• Feedback: either
  ➢ make an appointment to see me, or
  ➢ chat after class

Assignment 3
• Details at the end of this lecture
Hand drawn diagrams in textbooks are good visualizations of networks. Often they display:
2. Planarity
3. Rectilinearity
4. Symmetry

Fundamental scientific question: *is there any algorithm that produces good pictures of network data?*

Connectivity notions are fundamental in any study of networks:

- A graph is *connected* if for every pair of vertices, there is a path between them.
- A graph is *k-connected* if there is no set of (k-1) vertices whose deletion disconnects the graph.
The classical approach to network visualization uses connectivity concepts.

Understanding the classical approach helps in understanding many network visualization algorithms.

Q: How can we draw a graph?
A: Break it up into connected components.

One-connected
- A graph is connected (one-connected) if there is a path between any pair of vertices.
- A maximal connected subgraph is a connected component, sometime just called a component.

This graph is one-connected
This graph is not connected

Every graph consists of one or more connected graphs.

The decomposition into connected components gives:
- A kind of abstraction of the graph;
- A “divide and conquer” approach to network visualization.

Classical method for drawing graphs:
1. Assign a region for each connected component, according to the relative sizes of the connected components.
2. Draw each connected component inside its drawing space.

We need a good bin-packing algorithm.

We need a good algorithm for drawing one-connected graphs.
Q: How can we draw a one-connected graph?

A: Break it up into biconnected components.

**Biconnected**
- A cutvertex is a vertex whose removal would disconnect the graph.
- A graph without cutvertices is biconnected.
- A biconnected component (block) is a maximal biconnected subgraph.

**Classical method for drawing connected graphs**

1. Draw the shape of the BC-tree
   - regions assigned for each biconnected component
   - noting the relative sizes of the biconnected components
2. Draw each biconnected component inside its region, constrained by the positions of the cutvertices

Cutvertices and biconnected components can be found in linear time with a straightforward algorithm.

The cutvertices and biconnected components form a tree structure called the BC-tree, or Block-Cutpoint tree.

The BC-tree gives
a) An abstraction of the connected graph;
b) A divide and conquer method

1. Draw the shape of the BC-tree
Q: How can we draw a biconnected graph?
A: Break it up into triconnected components

**Triconnected**
- A separation pair is a pair of vertices whose removal would disconnect the graph.
- A graph without separation pairs is **triconnected**.

Classical method for drawing **connected** graphs

1. Draw the shape of the BC-tree
   - regions assigned for each biconnected component
   - noting the relative sizes of the biconnected components

2. Draw each biconnected component inside its region, constrained by the positions of the cutvertices

We need a good tree drawing algorithm
We need a good algorithm for drawing biconnected graphs

Classical method for drawing **biconnected** graphs

1. Choose one of the triconnected components, and draw it according to the size and structure of the other triconnected components.
2. Draw each other triconnected component inside its drawing space, gluing them together according to the structure.

WARNING: the next 5 minutes contains many oversimplifications.
Triconnected components can be found in linear time, with a very complex algorithm (J. Unimp. Alg., 1970s).

SPQR tree (3-block tree)
- Nodes are the triconnected components
- Edges are separation pairs

The SPQR tree gives:
1) An abstraction of a biconnected graph
2) A divide and conquer approach

Classical method for drawing biconnected graphs
1. Find the SPQR tree, choose a root r.
2. Draw the triconnected component corresponding to the root r, using cigar-shaped polygons for the virtual edges.
3. For each child c of r, draw the graph corresponding to the tree rooted at c, in the cigar shape, recursively.

We need good algorithms for drawing triconnected graphs

Q: How can we draw a triconnected graph?
A: It's fairly easy
- Triconnected graphs are very well behaved and easy to deal with
- In many practical cases, triconnected components are small
Summary: The classical approach

Abstraction / Decomposition

Graph

Component


Remarks on the classical approach

Where does it come from?
- The classical connectivity approach has been used for a long time by mathematicians to prove theorems that relate geometry and graph theory
- Enunciated by Tutte in the late 1950s

Remarks on the classical approach

Has it been implemented?
- There are many possible variations, many have been implemented
- First implemented in the GRAX system by Ron Read
  - late 1970s
  - PDP11/20 and PDP11/40
  - Tek4001
- Many other implementations over the years since 1980

Remarks on the classical approach

Does it work?
- Works well as long as:
  a) Almost all biconnected components are small
  b) Most triconnected components are very very very small

2. Planarity

A graph is planar if it can be drawn without edge crossings.
A graph is **planar** if it can be drawn without edge crossings.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Kuratowski’s Theorem (1930)**

A graph is planar if and only if it does not contain a subgraph that is a subdivision of $K_5$ or $K_{3,3}$.

**$K_{3,3}$** (complete bipartite graph on 6 vertices)  
**$K_5$** (the complete graph on 5 vertices)

Kuratowski’s Theorem is mathematically elegant
- Connects the geometric notion of planarity with the discrete notion of subgraph

**BUT**
- Kuratowski’s Theorem does not lead immediately to an efficient algorithm

A planar drawing divides the plane into **faces**.

<table>
<thead>
<tr>
<th>$F_0$</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
<th>$F_4$</th>
<th>$F_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_0$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$F_1$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$F_2$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$F_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$F_4$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$F_5$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The incidence structure of the faces defines the **topological embedding** of the graph.

How to make a planar picture of a planar graph:
1. Get the topology right
2. Place the nodes and route the edges
1. Get the topology right

Graph

Vertex-edge incidence structure

Topological embedding

Face-vertex-edge incidence structure

Theorem: A graph is planar if and only if each of its triconnected components is planar.

Theorem: There is only one topological embedding of a triconnected planar graph (on the sphere).

Use the classical connectivity approach

Hopcroft-Tarjan Planarity Algorithm (1974)
- Efficiently tests whether a biconnected graph is planar or not
- “Can be adjusted” to give a topological embedding
- Very complicated algorithm; implementation difficult
  - First version incorrect; corrected by Deo (1976)
  - Most implementations incorrect
  - Topological embedding not correctly implemented until the 1990s (?)
    - Implemented by Mehlhorn in mid-1990s
    - Implementation corrected by Mutzel

Many other Planarity Algorithms
- Lempel-Even-Cederbaum 1966
- Booth-Lueker 1976
- Rosensthiel-de Frayssieux 1990
- Hsu/Boyer-Myvold 2000

All are efficient and effective, but none of them are elegant.

Open problem (Read, 1970s):
Develop an elegant planarity algorithm (teachable to 2nd year undergraduates).

Planarity compromise

How do we deal with a nonplanar graph \( G=(V,E) \)?

1. Find a maximum planar subgraph \( G'=(V,F) \) with \( F \subseteq E \); find a topological embedding of \( G' \)

2. Route the edges of \( E-F \) using a shortest path through the faces, replacing crossing points with dummy nodes, making a planar graph \( G'' \)

3. Find a topological embedding of \( G'' \)
1. Find a maximum planar subgraph
2. Route the edges of E-F
3. Find a topological embedding

For a nonplanar graph \( G = (V, E) \)
1. Find a maximum planar subgraph \( G' = (V, F) \), with \( F \subseteq E \); find a topological embedding of \( G' \)
2. Route the edges of \( E - F \) using a shortest path through the faces, replacing crossing points with dummy nodes, making a planar graph \( G'' \)
3. Find a topological embedding of \( G'' \)

We need solutions for a difficult problem:

**Maximum planar subgraph (MPS)**
- Given a graph \( G \)
- Find a planar subgraph of \( G \) with as many edges as possible

Note
- MPS is NP-complete
- Many heuristic approaches in the last 30 years

*The most successful approach to MPS so far is to encode MPS as an Integer Linear Program (Mutzel, 1994).*

**Variables**
- \( x_{e} \) edge \( e \)
- \( x_{e} \in \{0, 1\} \)

**Objective**
- \( \text{maximize} \sum_{e} x_{e} \)
- Defines the set of planar subgraphs as a polytope

**Constraints**
- \( 0 \leq x_{e} \leq 1 \) \( \quad \forall \text{ edge } e \in E \)
- \( \sum_{e} x_{e} = |K| - 1 \) \( \quad \forall \text{ Kuratowski subgraph } K \text{ in } G \)
- \( \sum_{e} x_{e} \leq 3 |V| - 6 \)
- \( x_{e} \in [0, 1] \)

3. Rectilinearity

Graph \( \rightarrow \) Topological embedding \( \rightarrow \) Picture

Face-vertex-edge incidence structure

route the edges and place the nodes
Rectilinear: each edge is a straight line.

- Eyes can follow straight lines better than bendy lines.

Fary’s Theorem (Wagner, 1936) Every planar graph has a rectilinear planar drawing.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wagner, 1936</td>
<td>Algorithm?</td>
</tr>
<tr>
<td>Fary, 1948</td>
<td>Algorithm?</td>
</tr>
<tr>
<td>Stein, 1951</td>
<td>Algorithm?</td>
</tr>
<tr>
<td>Tutte, 1960</td>
<td>(O(n^{1.5}))</td>
</tr>
<tr>
<td>Read, 1975</td>
<td>(O(n))</td>
</tr>
<tr>
<td>Chiba et al., 1985</td>
<td>(O(n))</td>
</tr>
<tr>
<td>de Frayssinet et al., 1989</td>
<td>(O(n))</td>
</tr>
<tr>
<td>Schnyder, 1990</td>
<td>(O(n))</td>
</tr>
<tr>
<td>Kant, 1993</td>
<td>(O(n))</td>
</tr>
</tbody>
</table>

Rectilinearity for planar hierarchical networks

Variations on planarity

A number of constraints are needed for good network visualization.

For example:

- In directed graphs, edges should follow a given direction (usually downward, or left-right).
- Some sets of vertices may to be placed together in “clusters”.
- Vertices may be constrained to “layers”, that is, horizontal lines.

Such constraints have motivated a number of variations of the notion on planarity.

Layered digraphs

In a **layered digraph**, each vertex is assigned to a layer.

- Layers are horizontal lines.
- A vertex must stay on its layer
- Edges drawn monotonically downward

Hierarchical planarity: A layered digraph is \(h\)-planar if it can be drawn without edge crossings.

\(h\)-planar

not \(h\)-planar
**Question:** does every h-planar graph have a rectilinear h-planar drawing?  

**Answer:** yes

**Theorem** (Feng et al., 1996)  
Every h-planar layered digraph has a rectilinear h-planar drawing.

The algorithm has two phases:

1. Increase connectivity, along the lines of the classical approach
2. Use a divide-and-conquer algorithm.

This phase gives a network with high connectivity:

- One source and one sink
- All the inner faces are triangles.

2. Use a divide-and-conquer algorithm.

Input

- a convex polygon \( P \)
- a triangulated layered digraph \( G \) with one source and one sink,
- a correspondence between vertices of \( P \) and the outside face of \( G \).

Output

- a straight-line drawing of \( G \) with outside face \( P \).

The general divide & conquer idea

\[ P \]

\[ G \]

\[ P_1 \]

\[ P_2 \]
Recursively draw the subgraphs $G_1$ and $G_2$ in the polygons $P_1$ and $P_2$.

The divide and conquer approach basically works, but:

- We must be careful with chords.
- Sometimes we must divide into three parts.
- We must divide $G$ with monotonic paths.

Any chord on a straight-line section of $P$ will prevent a straight-line drawing.

We must ensure that there are no such chords.

But chords can also be helpful ...

We can divide the graph into two parts along a chord of the outside face.

1. Find a chord of the outside face.
2. Divide $G$ along this chord.
3. Cut $P$ along a straight line between the vertices of $P$ corresponding to the endpoints of the chord.

The "glue" part of the algorithm is easy as well.

But … the chord may not exist …
If the outside face has no chord, then we need to divide $G$ into three parts.

For convexity, we need the three cutting lines to be straight. This means that they must be chordless.

We must find

- A “pivot” $v$
- Three chordless paths $\pi_1$, $\pi_2$, and $\pi_3$, between the outside cycle and the pivot.

Also, the paths must be “monotonic”...

A path in $G$ is monotonic if all the edges point in the same direction (upward or downward).

Straight-line cuts are impossible with non-monotonic paths.

Given

- A triangulated h-planar layered digraph with one source, one sink, and no chord on the outside face.
- We must find
  - a pivot $v$
  - 3 chordless paths $\pi_1$, $\pi_2$, and $\pi_3$, between the outside cycle and the pivot
  - such that each path is monotonic.
  - and a location for the pivot $v$

We choose the pivot $v$ such that

- $v$ is not on the outside face
- $v$ is adjacent to the outside face.

In this way, $\pi_3$ is a single edge.

We then show that there are two more monotonic chordless paths $\pi_1$ and $\pi_2$ from $v$ to the outside face.

How do we find such pivot?
Choose a vertex \( w \) on the left hand outside face.
Consider the rightmost up- and down-neighbors of \( w \).
One of these must be not on the outside face; otherwise \( w \) has degree 2 and the outside face has a chord.

Thus we can assume the rightmost up-neighbor of \( w \), is not on the outside face.
This will be the pivot \( v \).

We find a chordless downward path as a "steepest descent" path to the outside face.
An upward path is found in the same way.

We have the three monotonic paths \( \pi_1, \pi_2, \pi_3 \).

We use a barycenter method to locate the pivot
- Ensures convexity

The rest is easy:
- Cut the graph \( G \) using the paths.
- Cut the polygon \( P \) using the location of \( v \).
- Then apply recursion and obtain a rectilinear drawing.

**Theorem** (Feng et al., 1996)
Every \( h \)-planar layered digraph has a \( h \)-planar straight-line drawing; the drawing can be obtained in linear time.

I have omitted
- Phase 1: a variation on the classical connectivity approach
- Many details.
- Some pathological cases.
- Achieving linear time.
Rectilinearity compromise

In some domains, orthogonal drawings are used. In such cases, we need to minimize the bendiness of the edges.

Bends Theorem (Tamassia 1987)
There is an E^3 (Efficient/Effective/Elegant) algorithm to make a drawing with a minimum number of bends, given a topological embedding.

Symmetry

Textbooks draw graphs with symmetry.

Symmetry
An isometry $\sigma$ is a symmetry of a figure $F$ if $\sigma(F) = F$.

Rotation by $90^\circ$

The set of symmetries of a figure forms a group under composition.

Symmetries in $\mathbb{R}^2$
- Rotations
- Reflections

Symmetry groups in $\mathbb{R}^2$
- Cyclic
- Size 2
- Dihedral

The symmetry group of a polygon with $n$ sides has size at most $2n$. 

Textbooks draw graphs with symmetry.

Symmetry
An isometry $\sigma$ is a symmetry of a figure $F$ if $\sigma(F) = F$.

Rotation by $90^\circ$

The set of symmetries of a figure forms a group under composition.

Symmetries in $\mathbb{R}^2$
- Rotations
- Reflections

Symmetry groups in $\mathbb{R}^2$
- Cyclic
- Size 2
- Dihedral

The symmetry group of a polygon with $n$ sides has size at most $2n$. 

**Symmetry problem**

*Input:* a figure $F$

*Output:* the symmetry group of $F$

**Algorithms for the symmetry problem:**

- Well studied
- Not difficult
- $E^3$ algorithms are available

**Automorphism**

Say $G=(V,E)$ is a graph.

A permutation $\beta: V \to V$ is an automorphism of $G$ if $(u,v) \in E$ if and only if $(\beta(u), \beta(v)) \in E$.

The set of automorphisms of a graph forms a group under composition.

**Automorphism groups**

- May be exponentially large
- Group theoretic structure can be anything

**Automorphism problem**

*Input:* a graph $G=(V,E)$

*Output:* the automorphism group of $G$

**Algorithmics of the automorphism problem**

- "Isomorphism complete"
  - No polynomial time algorithm known
  - Not known to be NP-hard
  - Somewhere in between P and NP
- Classic unsolved problem in Computer Science

**Discrete**

Automorphism
- Applies to graphs, combinatorial structures
- Detection problem is mostly unsolved

**Geometric**

Symmetry
- Applies to geometric figures
- Detection problem is relatively easy

**A planar automorphism** of a graph $G$ is an automorphism which can be displayed as a symmetry of a planar drawing of $G$. 

**Planar automorphism**

**Nonplanar automorphisms**
All automorphisms of G

Planar automorphisms of G

Planar Automorphism Problem
Input: A planar graph G
Output: A maximum size planar automorphism group

Symmetric Drawing Problem
Input: A planar graph G
Output: A maximally symmetric drawing of G.

The planar automorphism problem can be solved in linear time.

Theorem (S. Hong et al., 1999 - 2006):
The planar automorphism problem can be solved in linear time.

We can make a maximally symmetric planar drawing of planar graph in linear time.

Triconnected Lemma
A maximally symmetric planar drawing of a triconnected planar graph can be found in linear time.

Algorithm
• Use the orbit-stabilizer theorem
• Use Fontet’s algorithm
• Use Mani’s theorem
• Use a projection

Suppose that A is a group of automorphisms of a graph G=(V,E), and v∈V.

The orbit Orbitₐ(v) of v is {u∈V : a(v)=u for some a∈A}.

Fontet’s algorithm (1976): efficiently computes all orbits of a planar graph

The stabilizer Stabₐ(v) of v in A is {a ∈ A : a(x)=x} (that is, the subgroup of A that fixes x).

Orbit -stabilizer Theorem:
For every v, |A| = |Stabₐ⁻¹(v)||Orbitₐ⁻¹(v)|
Some administrative matters

Assignment 3 is available on the web

- You will give a 15 – 20 minute talk about a research paper.
- Your talk should be a clear summary of the paper. You will be marked on the clarity, accuracy, and completeness of your presentation.
- Talks will be in class on October 7, 14, and 21.
- Details at the end of this lecture

BUT FIRST!

Masters students: List A
Honours students: List B

How to select a paper?
- Look at the list, and decide which papers you like.
- Send me email, with 3 – 4 papers that you prefer, in order of preference.
- Allocation of papers will be on a first-come-first-served basis
- If you have not selected a paper by next Wednesday, then I will select one for you.

Talks will be in class on October 7, 14, and 21.

Scheduled by me, after I get the allocation of people to papers done.

Steinitz Theorem (1930s): Every triconnected planar graph can be drawn as a three dimensional polyhedron

Mani’s Theorem (1971): Every triconnected planar graph can be drawn as a three dimensional polyhedron such that the automorphism group of the graph is isomorphic to the symmetry group of the polyhedron

All automorphisms can be displayed in a 3D polyhedral drawing

Mani's Theorem All automorphisms can be displayed in a 3D polyhedral drawing

Face Lemma: a symmetry of a drawing stabilizes a face (the outside face)

Lemma: The maximum size planar automorphism group is the maximum stabilizer of a face, over all faces.

Algorithm: Use Fontet to find a minimum size orbit, this gives a maximum size stabilizer

Assignment will be carried out in class on October 7, 14, and 21.