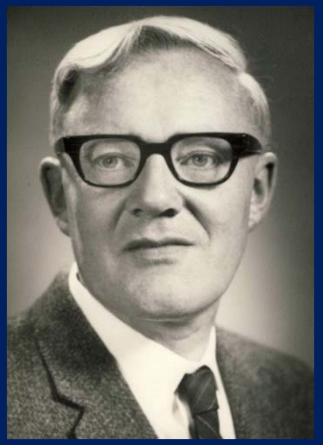
How to Draw a Graph … Revisited

Peter Eades
University of Sydney

W. T. Tutte 1917 - 2002

- Code breaker at Bletchley park
- Pioneer of matroid theory
- Pioneer of graph theory
- Inventor of the first *graph drawing* algorithm
  - His 1963 paper "*How to Draw a Graph*"* has inspired much research

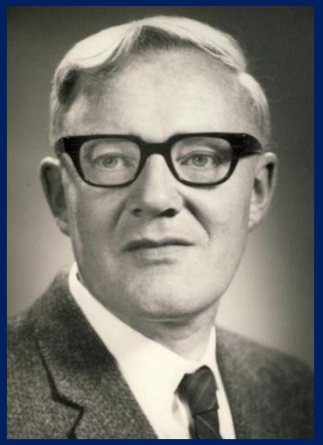


This talk is about two strands of Graph Drawing research that followed Tutte's work

*William T. Tutte. "How to draw a graph." Proc. London Math. Society, 13(52):743 -768, 1963.

This talk

1. Background
   a) Graphs
   b) Graph drawings
   c) Connectivity
   d) Planar graphs
   e) Topological embedding
   f) Good graph drawings



2. How to draw a graph?
   a) Before Tutte: 1920s – 1950s
   b) Tutte: 1960s
   c) After Tutte: 1970s – 1990s
   d) Recent work

(a) Graphs

A *graph* consists of

- Nodes, and
- Binary relationships called “edges” between the nodes

Example: a “Linked-In” style social network

Nodes:

- Alice, Andrea, Annie, Amelia, Bob, Brian, Bernard, Boyle

Edges

- Bob is connected to Alice
- Bob is connected to Andrea
- Bob is connected to Amelia
- Brian is connected to Alice
- Brian is connected to Andrea
- Brian is connected to Amelia
- Boyle is connected to Alice
- Boyle is connected to Andrea
- Boyle is connected to Annie
- Bernard is connected to Alice
- Bernard is connected to Andrea
- Bernard is connected to Annie

(b) Graph drawing

Graph

A graph consists of
- Nodes, and
- Edges

Graph Drawing

A *graph drawing* is a picture of a graph.

That is, a graph drawing is a mapping that assigns
- a location for each node, and
- a curve to each edge.

A social network

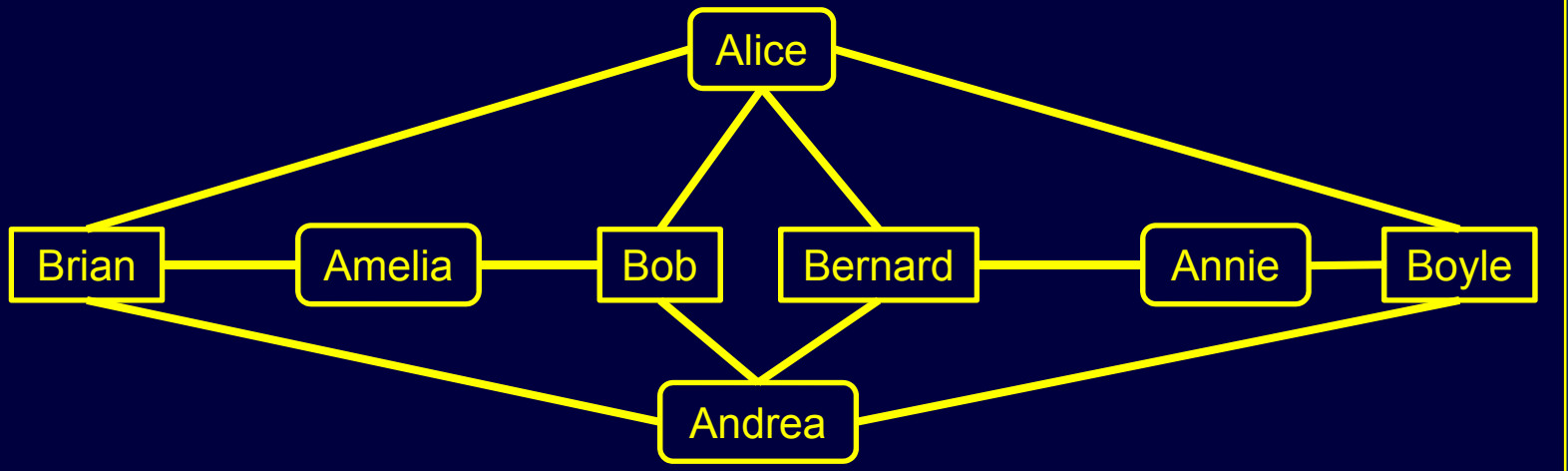
Nodes:

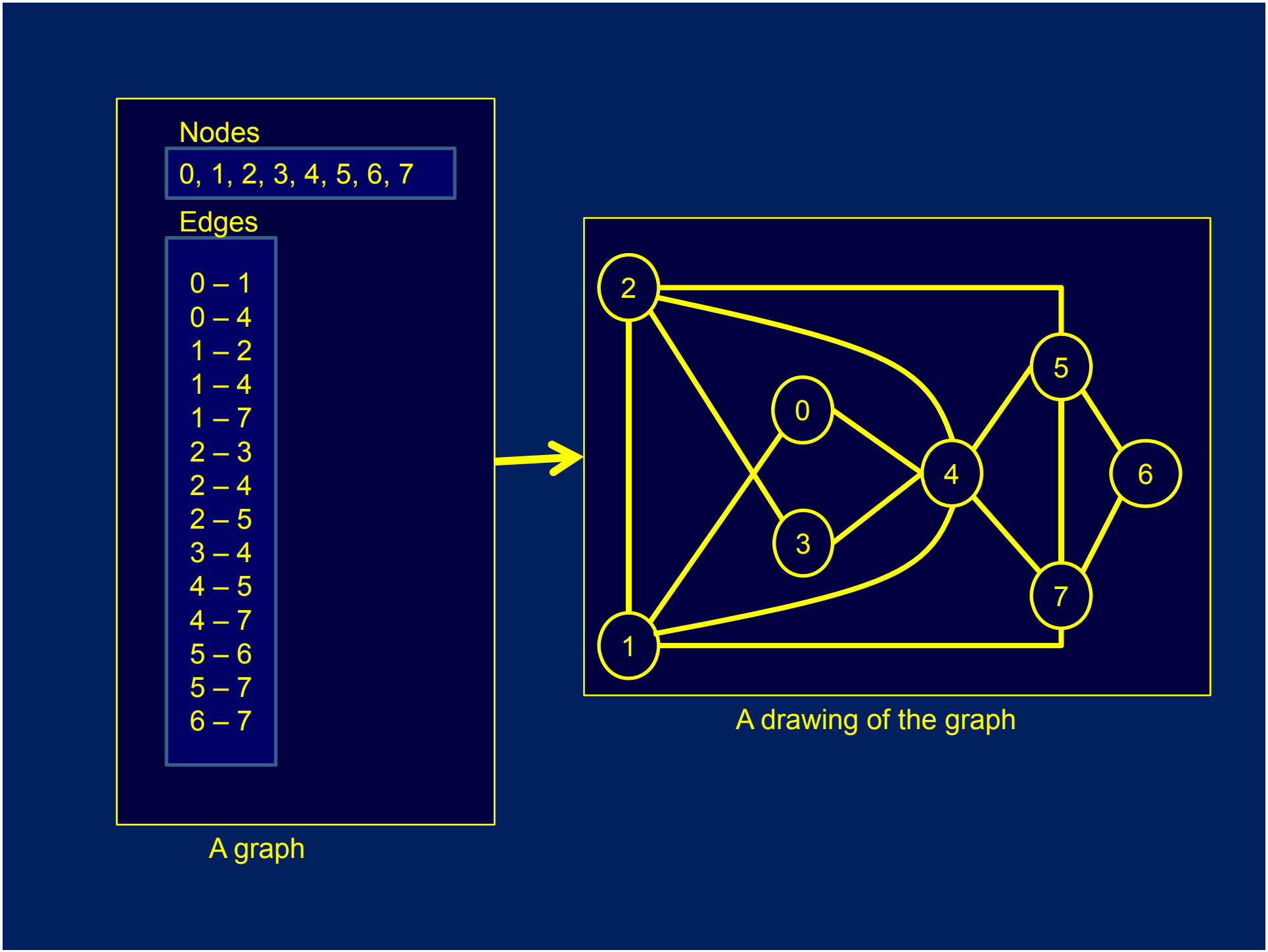
- Bob, Brian, Bernard, Boyle, Alice, Andrea, Annie, Amelia

Edges

- Bob is connected to Alice
- Bob is connected to Andrea
- Bob is connected to Amelia
- Brian is connected to Alice
- Brian is connected to Andrea
- Brian is connected to Amelia
- Boyle is connected to Alice
- Boyle is connected to Andrea
- Boyle is connected to Annie
- Bernard is connected to Alice
- Bernard is connected to Andrea
- Bernard is connected to Annie

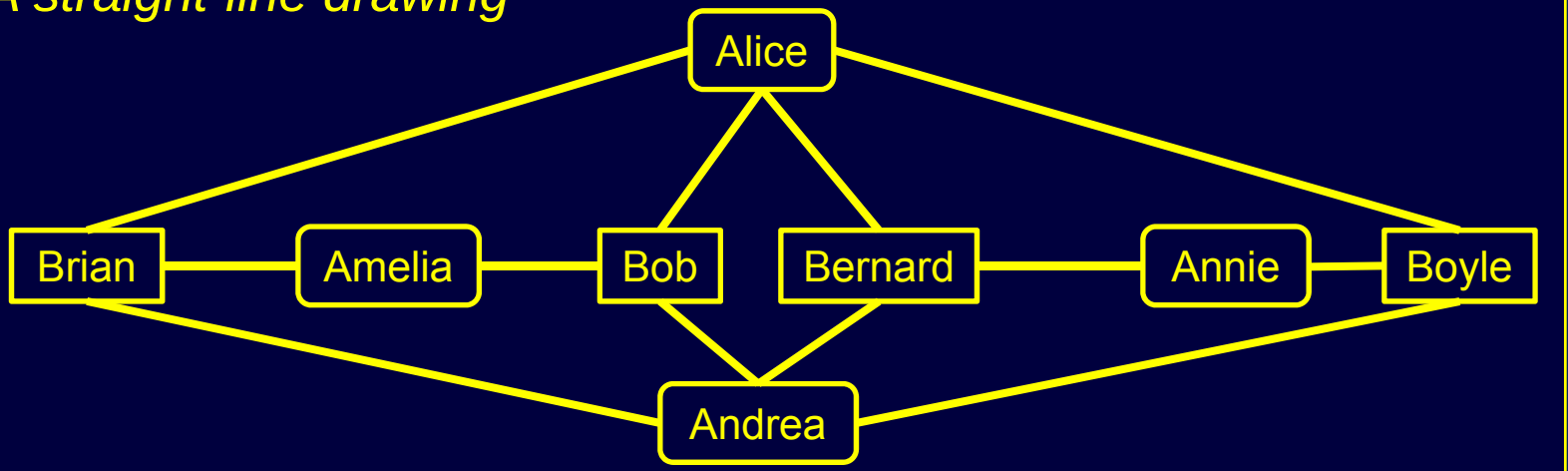
A drawing of the social network



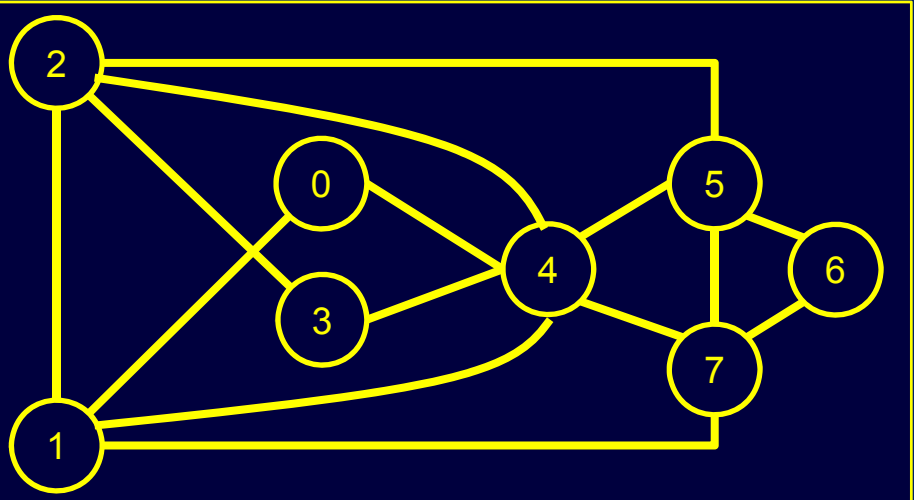

Nodes
0, 1, 2, 3, 4, 5, 6, 7
Edges
0 – 1
0 – 4
1 – 2
1 – 4
1 – 7
2 – 3
2 – 4
2 – 5
3 – 4
4 – 5
4 – 7
5 – 6
5 – 7
6 – 7
A graph
2
0
5
4
6
3
7
1
A drawing of the graph

A graph drawing is a *straight-line drawing* if every edge is a straight line segment.

A straight-line drawing



NOT a straight-line drawing



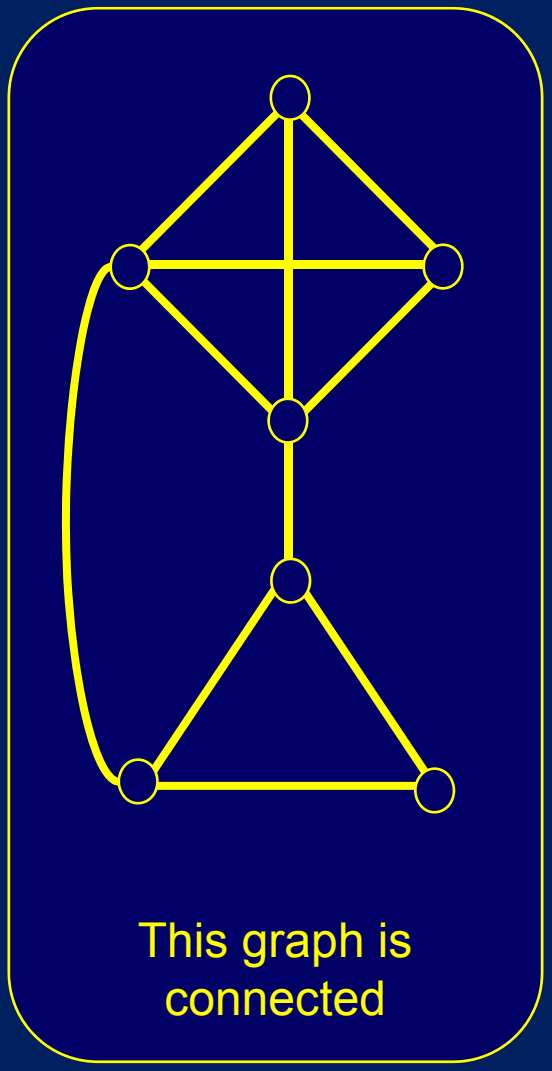
(c) Connectivity

Connectivity notions are fundamental in any study of graphs or networks

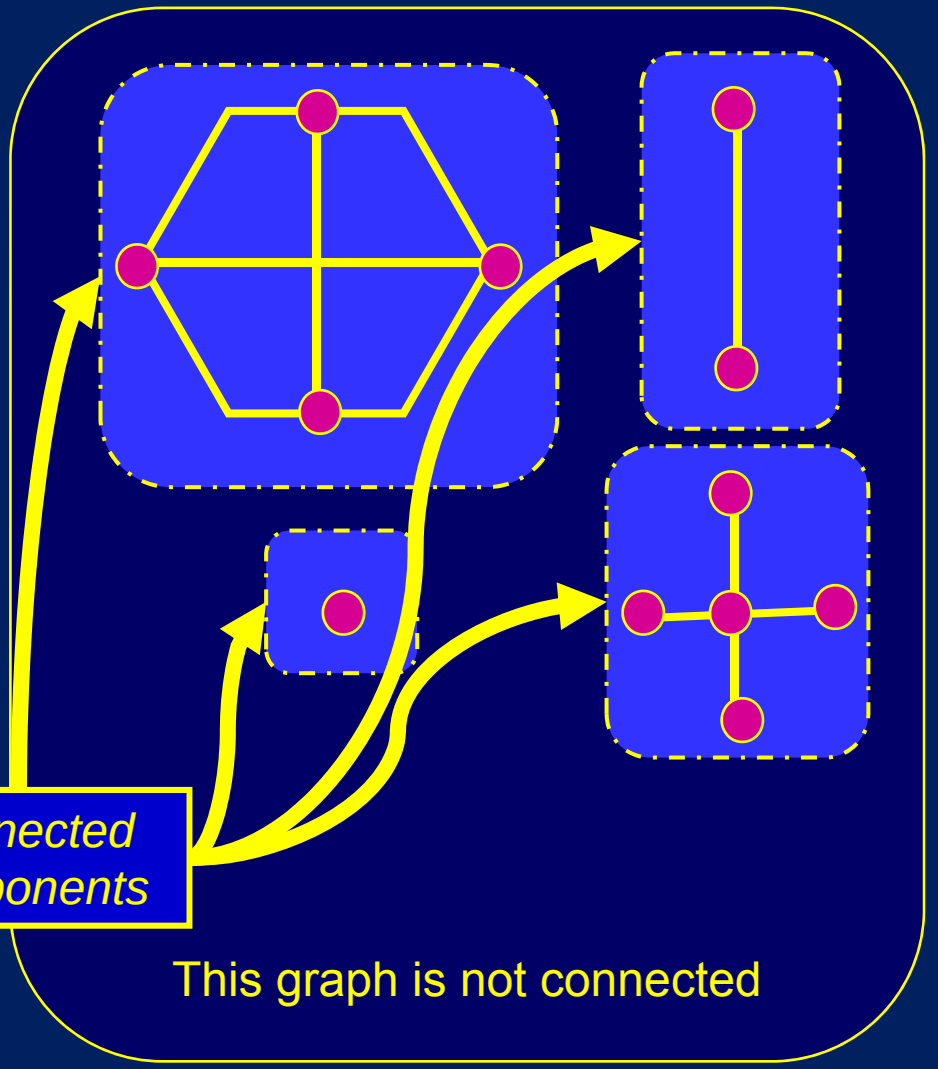
- A graph is *connected* if for every pair u, v of vertices, there is a path between u and v.

- A graph is *k-connected* if there is no set of (k-1) vertices whose deletion disconnects the graph.

  - k = 1: “1-connected” ≡ “connected”
  - k = 2: “2-connected” ≡ “*biconnected*”
  - k = 3: “3-connected” ≡ “*triconnected*”



This graph is connected



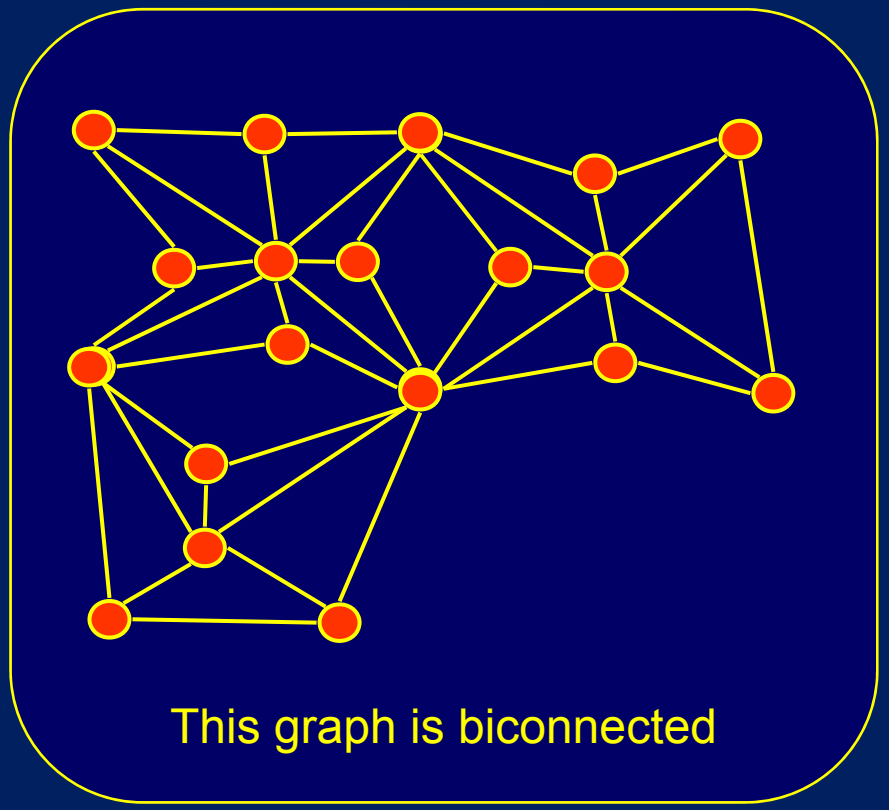
This graph is not connected

Connectivity notions are fundamental in any study of networks

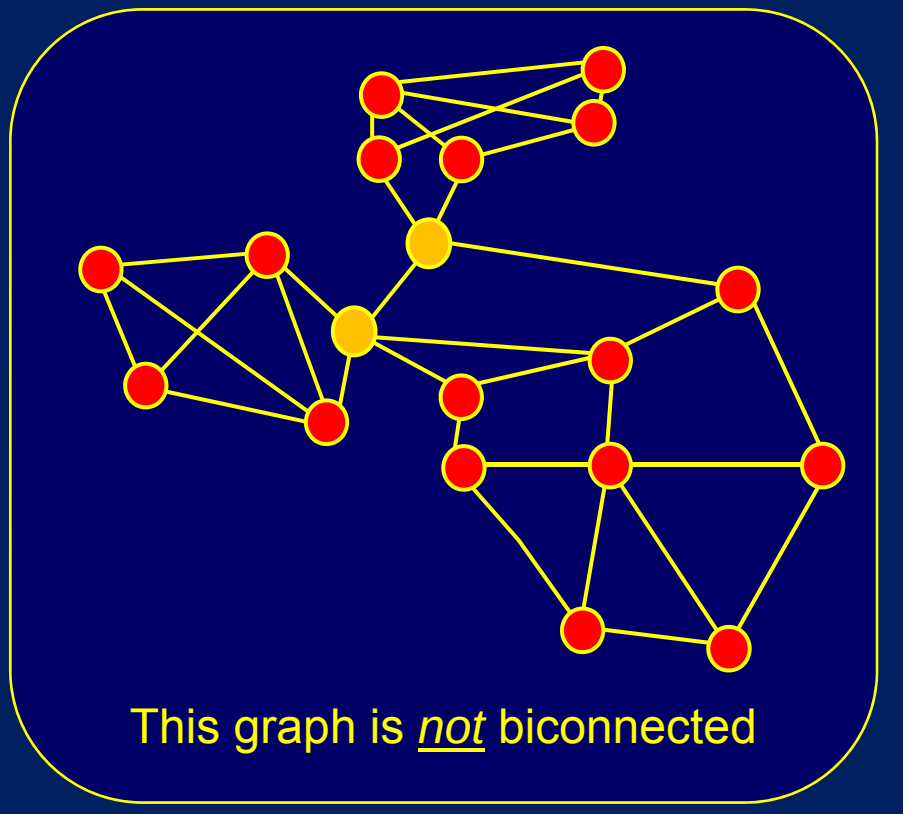
- A graph is *connected* if for every pair u, v of vertices, there is a path between u and v.
- A graph is *k-connected* if there is no set of (k-1) vertices whose deletion disconnects the graph.
  - k = 1: “1-connected” ≡ “connected”
  - k = 2: “2-connected” ≡ “*biconnected*”
  - k = 3: “3-connected” ≡ “*triconnected*”

"2-connected" ≡ "*biconnected*"

- A *cutvertex* is a vertex whose removal would disconnect the graph.
- A graph without cutvertices is *biconnected*.



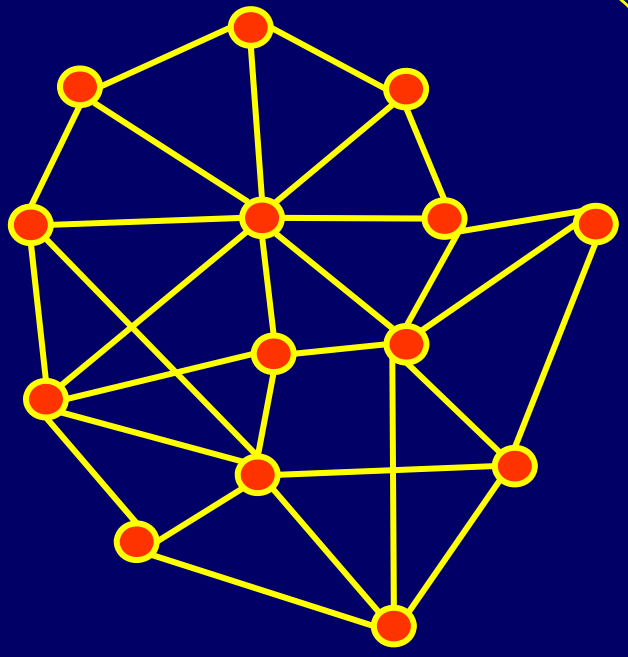
This graph is biconnected



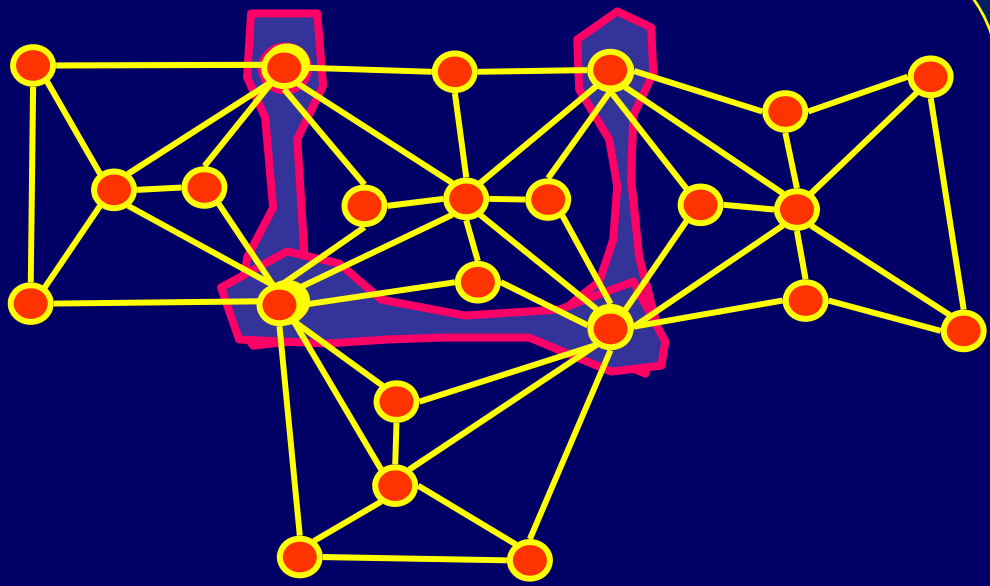
This graph is *not* biconnected

"3-connected" ≡ "*triconnected*"

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



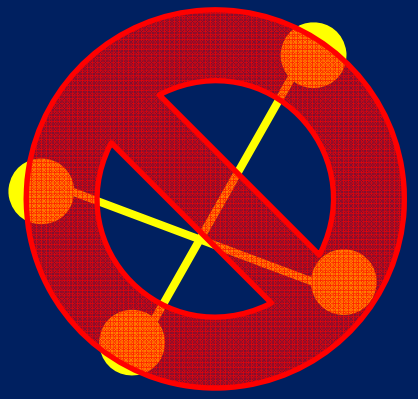
This graph is triconnected



This graph is *not* triconnected

(d) Planar graphs

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



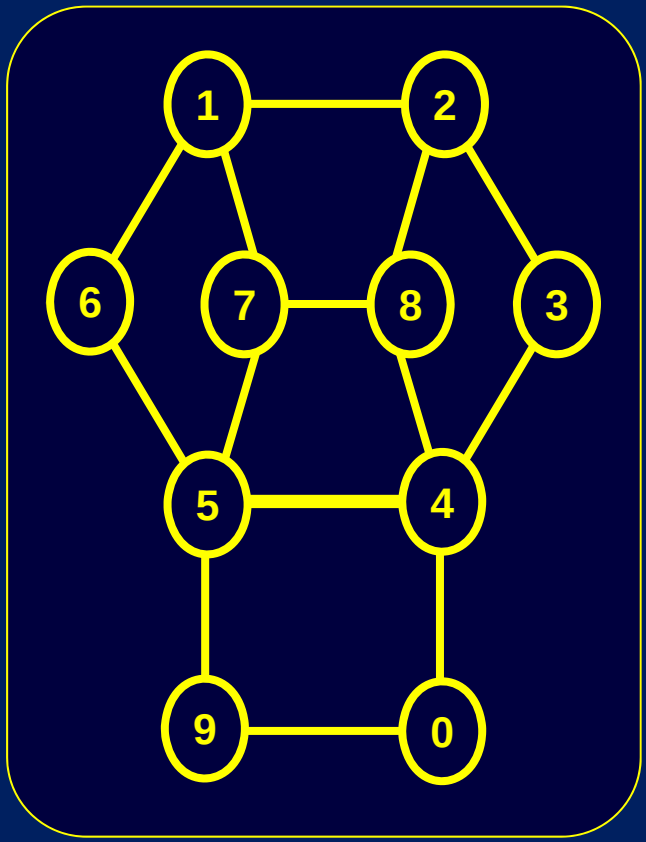
A graph is *planar* if it can be drawn without edge crossings.

Nodes:

- 0,1,2,3,4,5,6,7,8,9

Edges

- 0 – 4
- 0 – 9
- 1 – 2
- 1 – 6
- 1 – 7
- 2 – 3
- 2 – 8
- 3 – 4
- 4 – 5
- 4 – 8
- 5 – 6
- 5 – 7
- 7 – 8



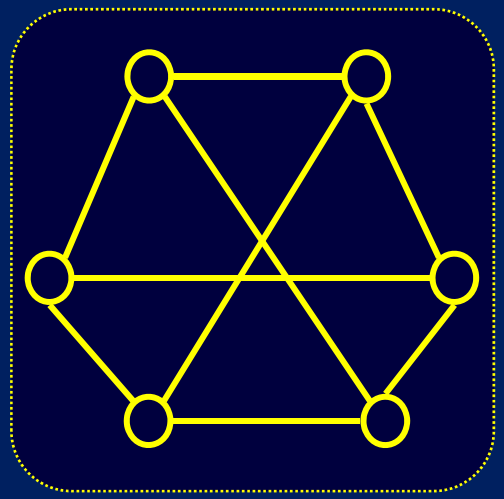
Non-planar

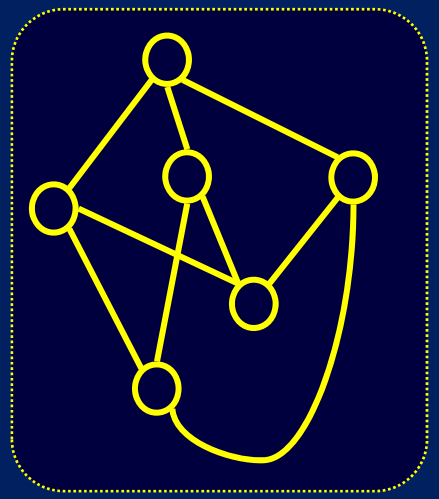
Nodes:

- 0,1,2,3,4,5

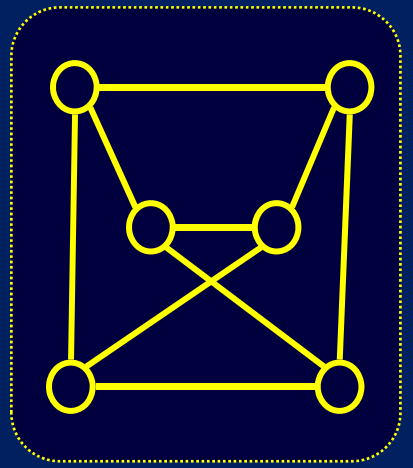
Edges

- 0 – 1
- 0 – 3
- 0 – 5
- 1 – 2
- 1 – 4
- 2 – 3
- 2 – 5
- 3 – 4
- 4 – 5



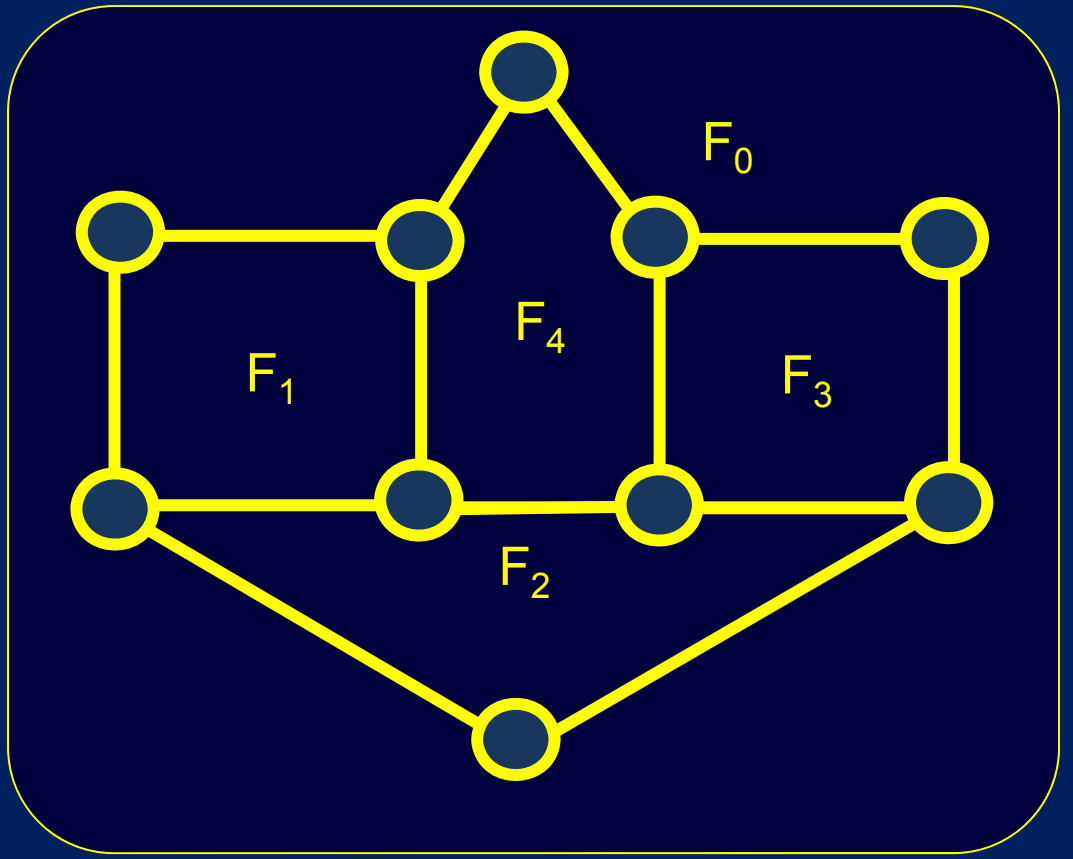


A graph is *non-planar* if *every* drawing has at least one edge crossing.



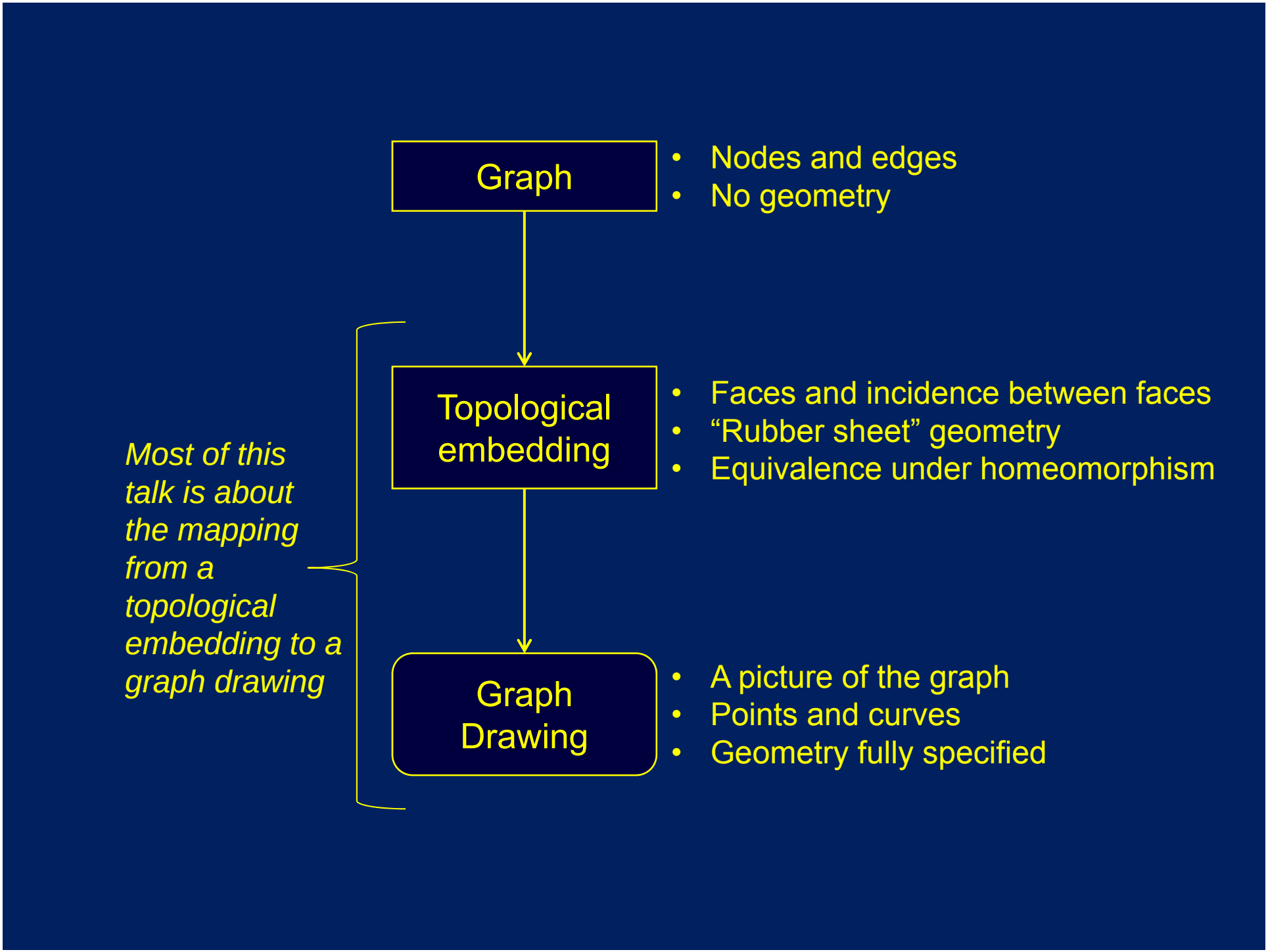
(e) Topological embedding

A planar drawing divides the plane into *faces*.



F_0 shares a boundary with F_1
F_0 shares a boundary with F_2
F_0 shares a boundary with F_3
F_0 shares a boundary with F_4
F_1 shares a boundary with F_2
F_1 shares a boundary with F_4
F_2 shares a boundary with F_1
F_2 shares a boundary with F_3
F_2 shares a boundary with F_4
F_3 shares a boundary with F_4

The boundary-sharing relationships of the faces defines a *topological embedding* of the graph drawing


Graph
Nodes and edges
No geometry
Topological embedding
Faces and incidence between faces
“Rubber sheet” geometry
Equivalence under homeomorphism
Most of this talk is about the mapping from a topological embedding to a graph drawing
Graph Drawing
A picture of the graph
Points and curves
Geometry fully specified

Euler formula

If

n = #vertices

f = #faces

m = #edges

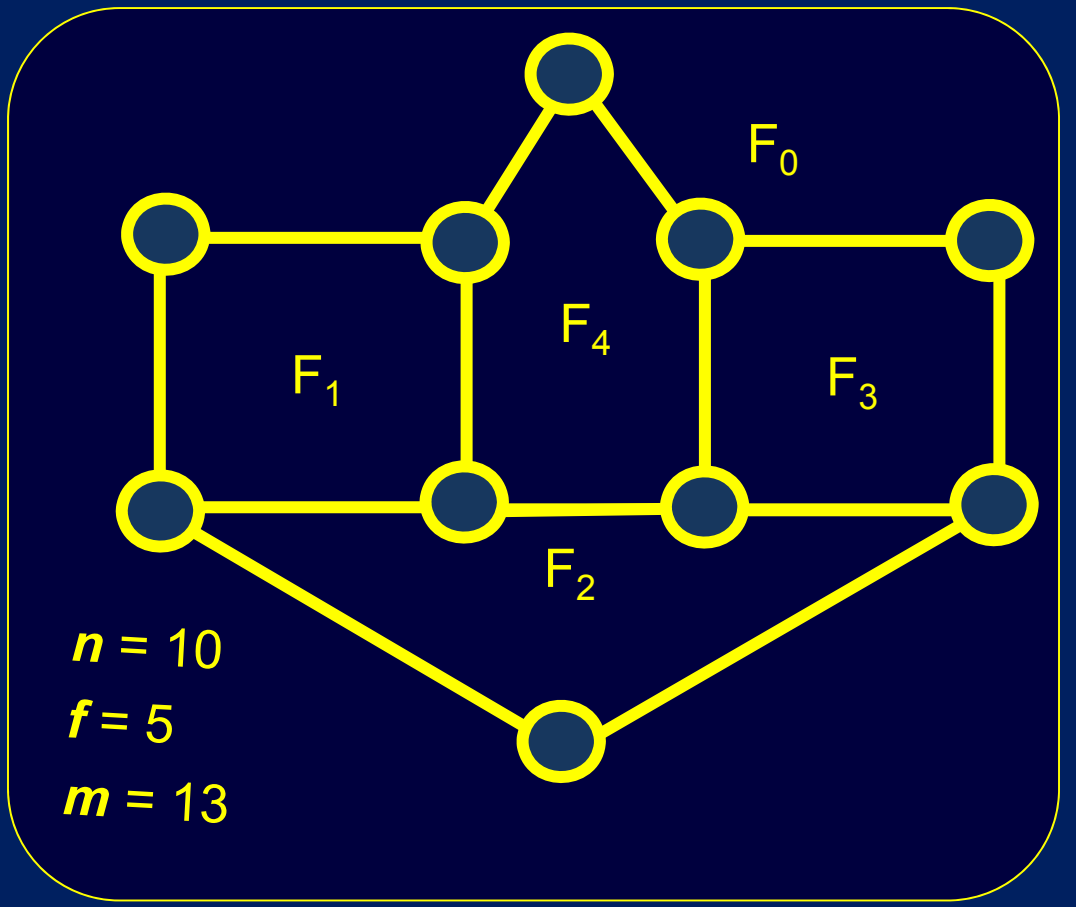
then

n+f = m+2

Corollary ***m ≤ 3n-6***

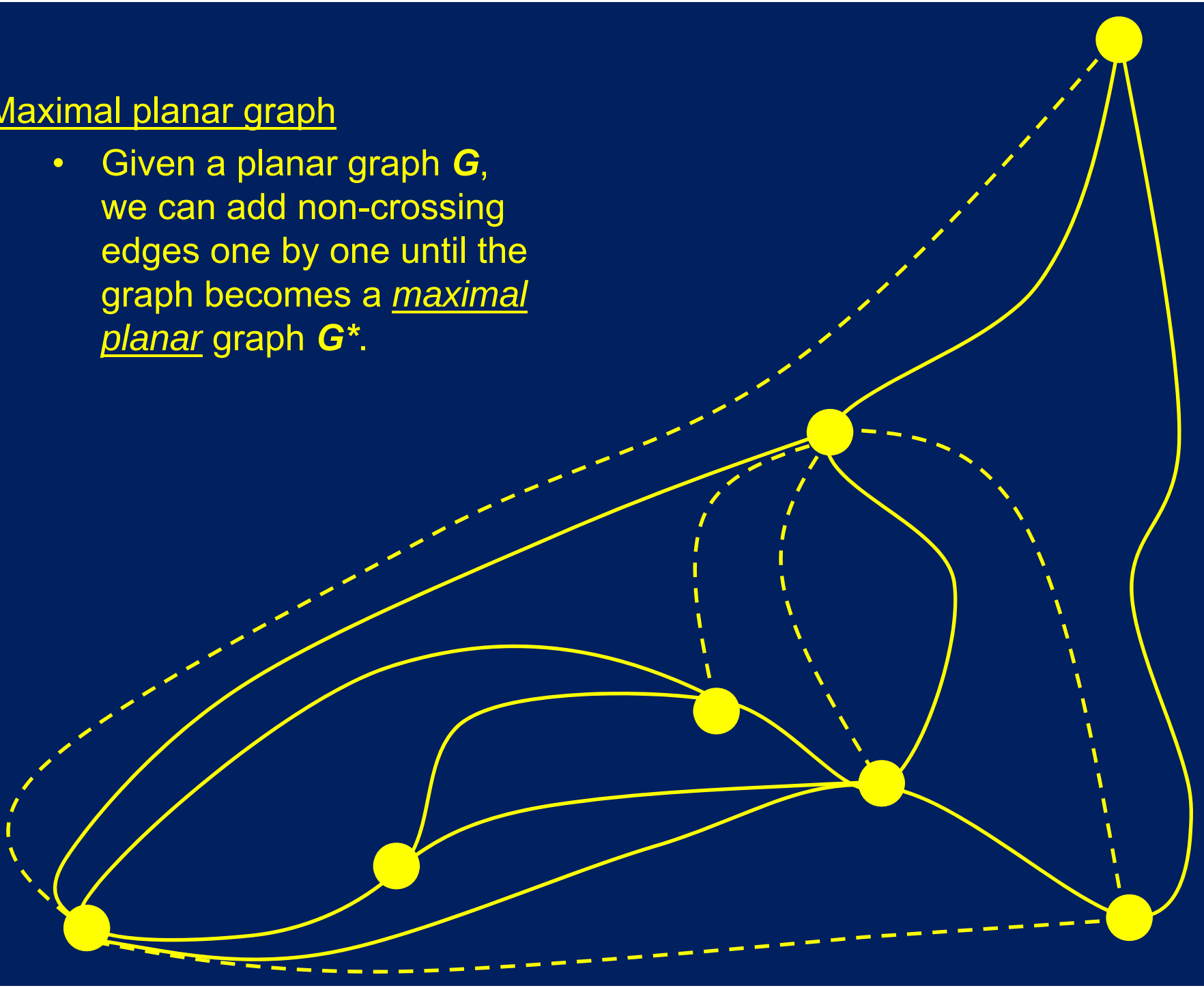
Corollary

If ***m = 3n-6*** then every face is a triangle



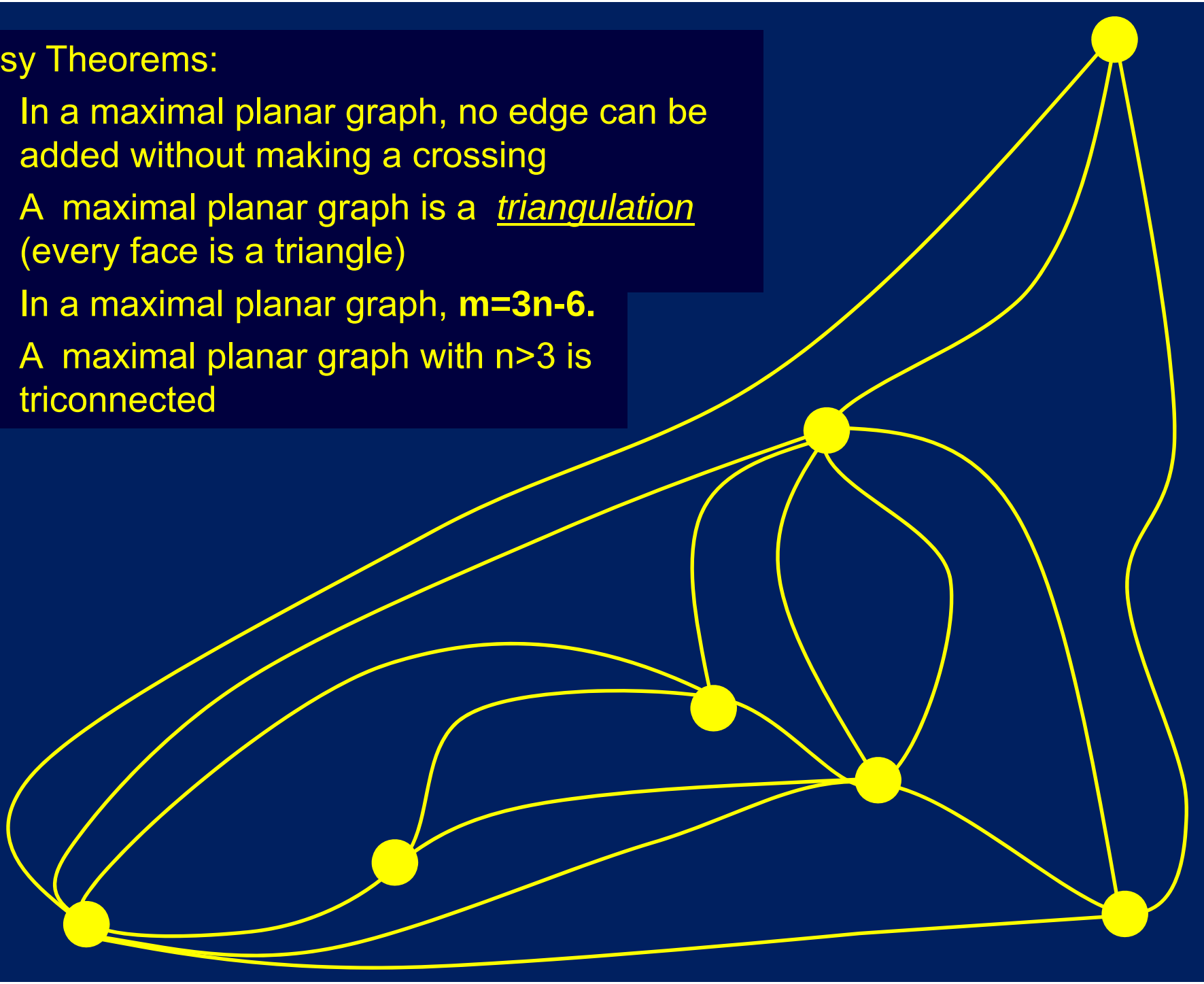
Maximal planar graph

- Given a planar graph ***G***, we can add non-crossing edges one by one until the graph becomes a *maximal planar* graph ***G****.



Easy Theorems:

- In a maximal planar graph, no edge can be added without making a crossing
- A maximal planar graph is a *triangulation* (every face is a triangle)
- In a maximal planar graph, **m=3n-6.**
- A maximal planar graph with n>3 is triconnected



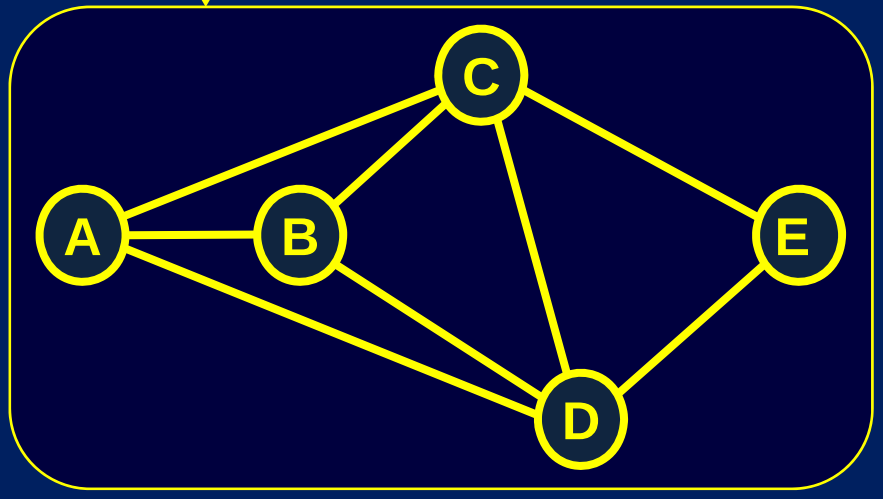
(e) Good graph drawing

Graph

A - B, C, D
B - A, C, D
C - A, B, D, E
D - A, B, C, E
E - C, D

The input is a graph with no geometry

Graph Drawing



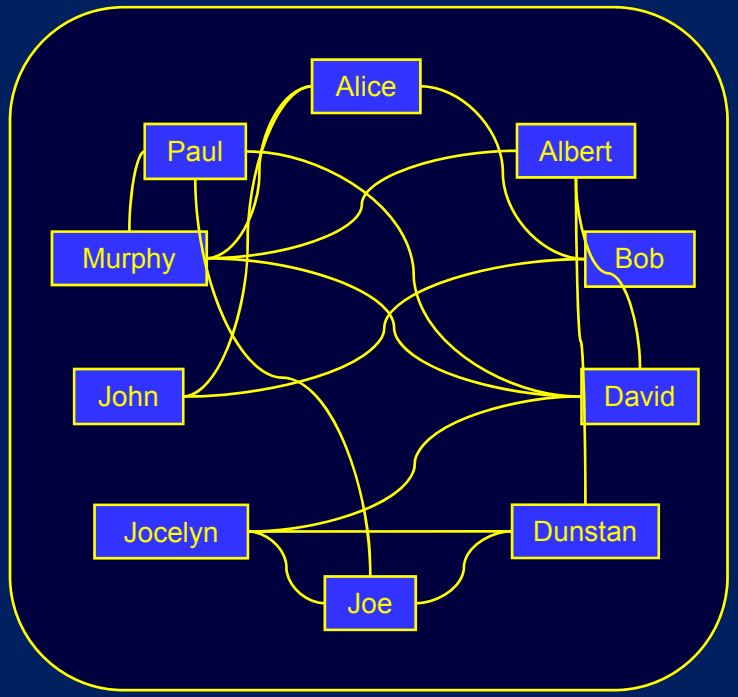
The output should be a <u>good</u> graph drawing:

- *<u>easy to understand</u>,*
- *<u>easy to remember</u>,*
- *<u>beautiful</u>.*

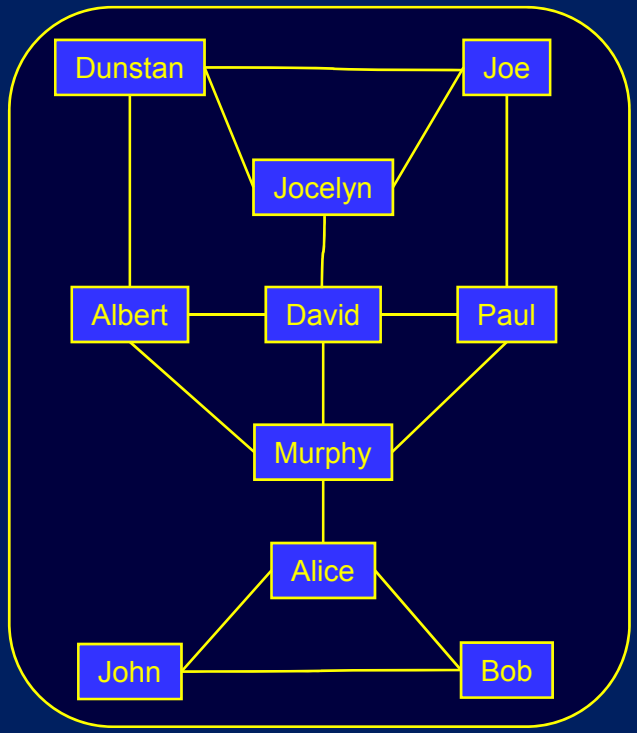
~1979 Intuition (Sugiyama et al. 1979; Batini et al 1982; etc):

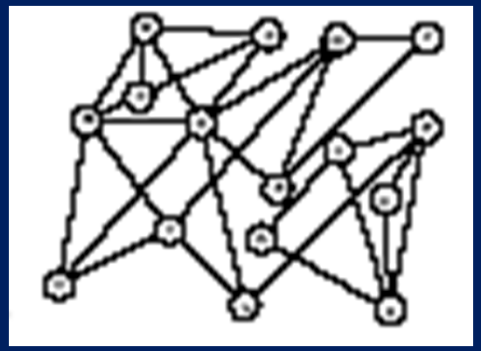
– Planar straight-line drawings make good pictures

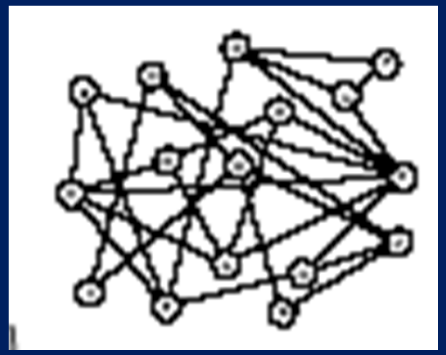
Bad drawing

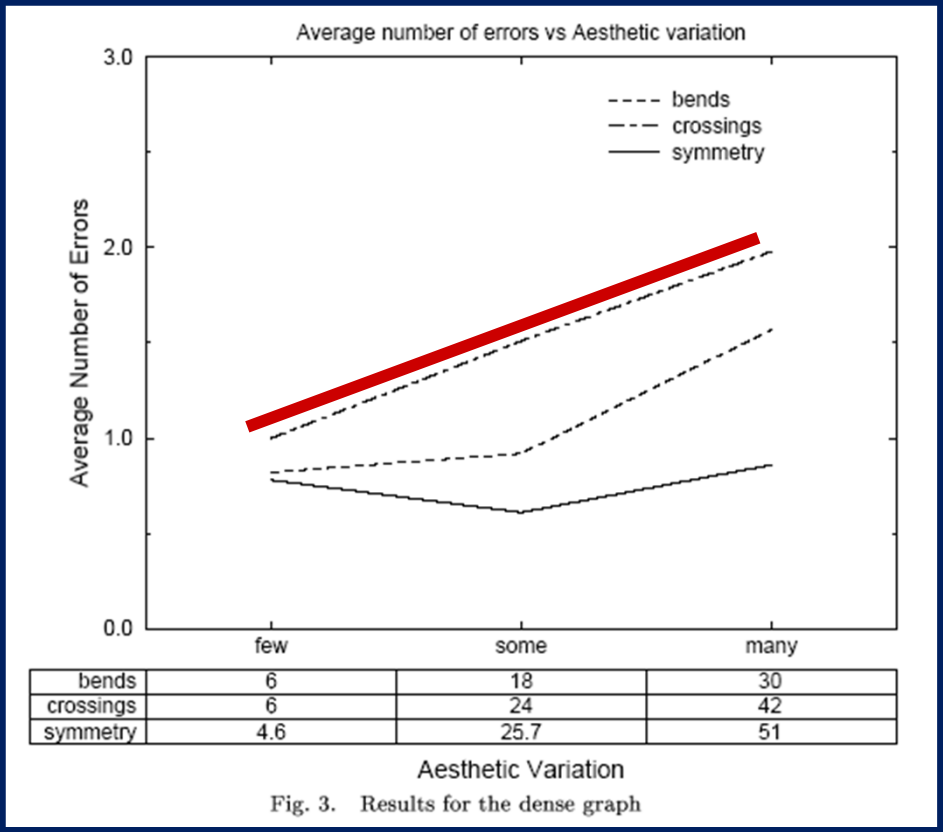


Good drawing









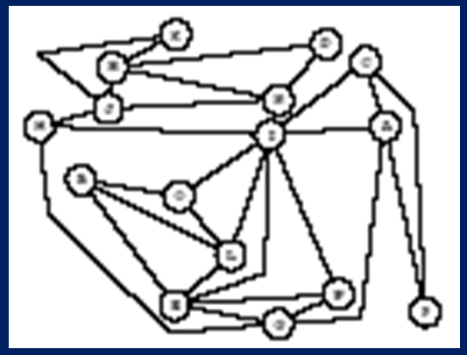
	few	some	many
bends	6	18	30
crossings	6	24	42
symmetry	4.6	25.7	51

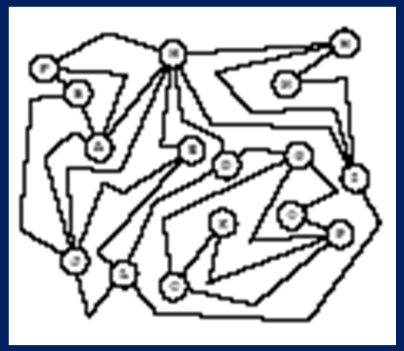
Fig. 3. Results for the dense graph

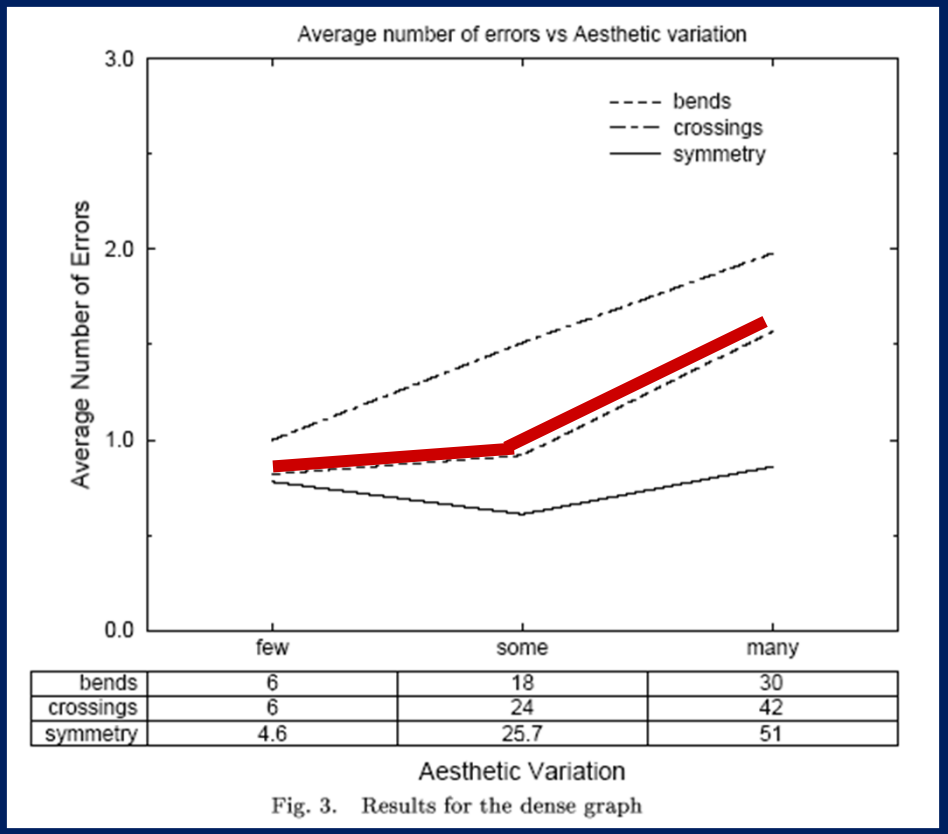
Purchase et al., 1997:

Significant correlation between *edge crossings* and human understanding

- *More edge crossings means more human errors in understanding*







bends	6	18	30
crossings	6	24	42
symmetry	4.6	25.7	51

Fig. 3. Results for the dense graph

Purchase et al., 1997:

Significant correlation between *straightness of edges* and human understanding

- *More bends mean more human errors in understanding*

What makes a *good* drawing of a graph?

- lack of edge crossings (planar drawings are good!)
- straightness of edges (straight-line drawings are good!)

(plus some other things)

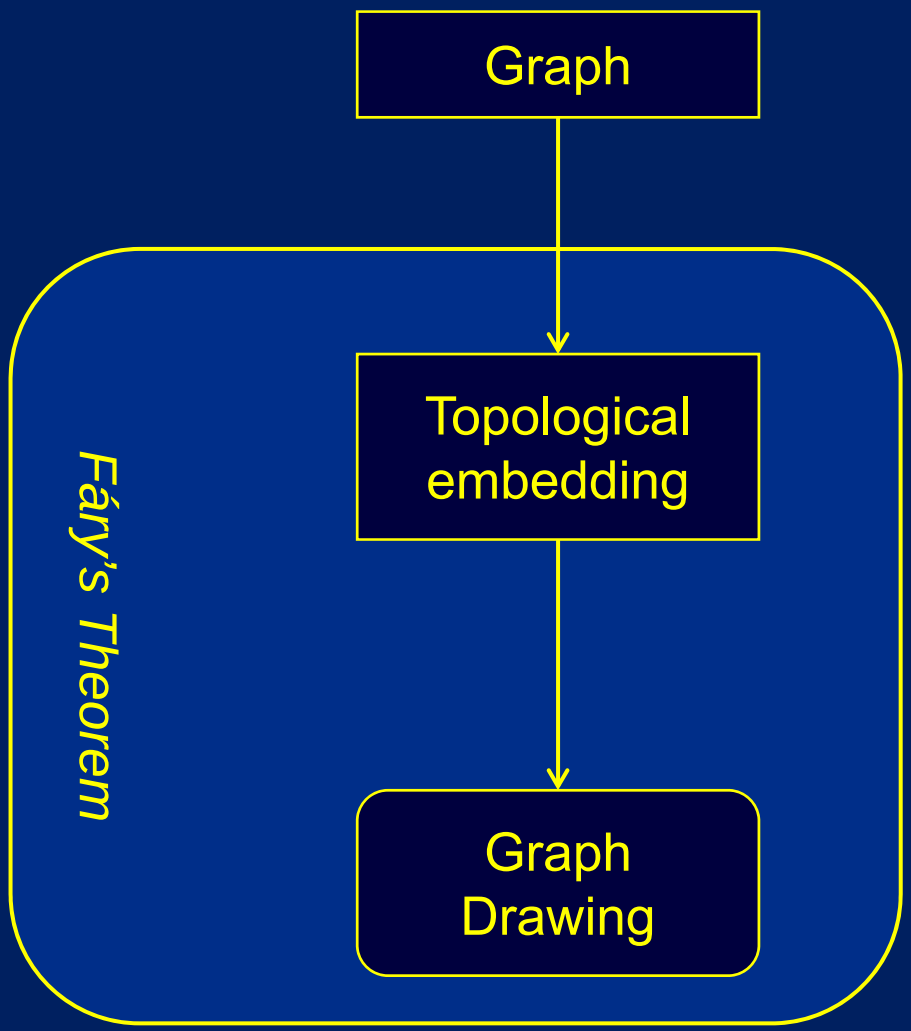
2. How to draw a graph

a) Before Tutte: 1920s – 1950s

Fáry's Theorem

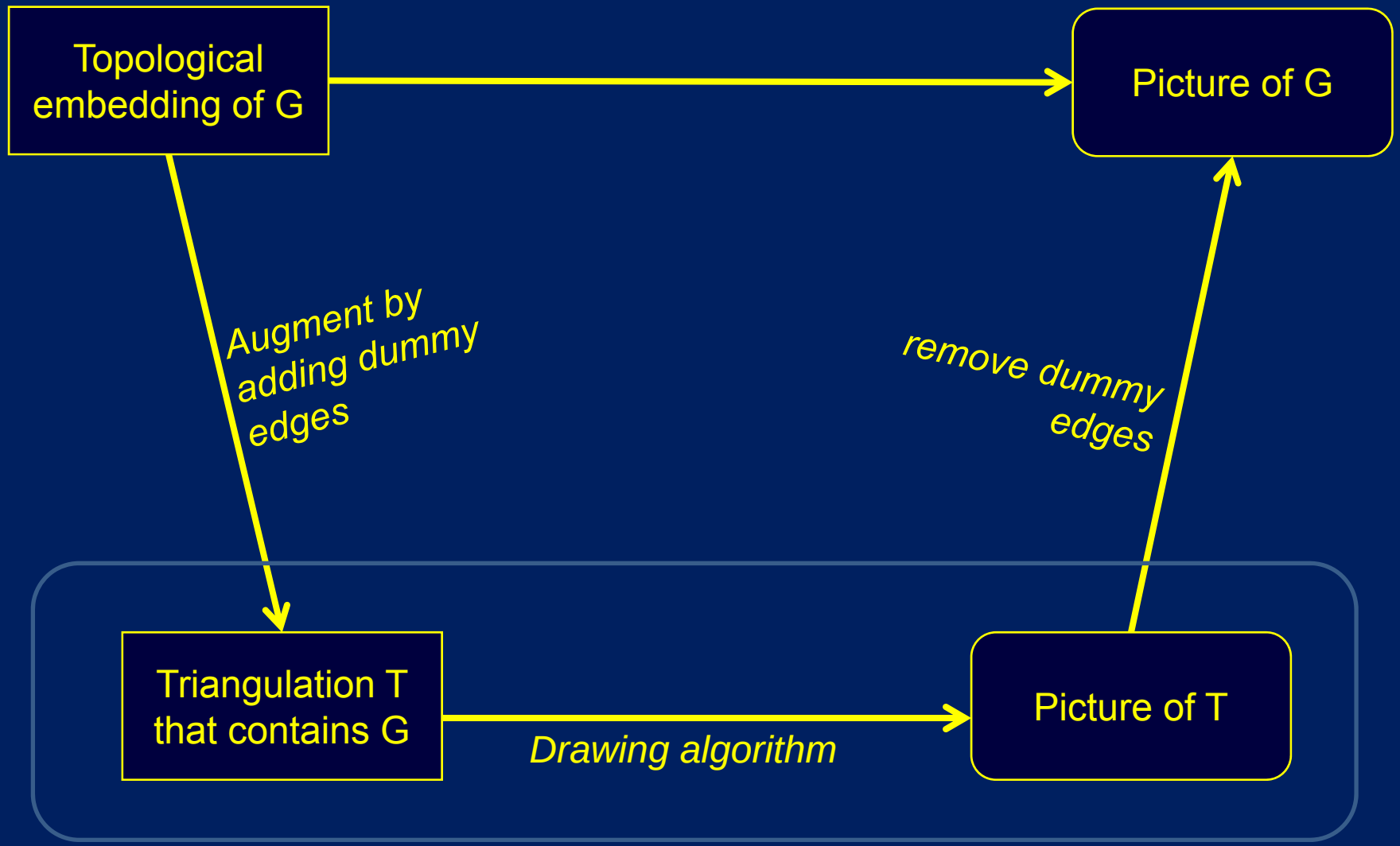
Every topological embedding of a planar graph has a straight-line planar drawing.

Proved independently by Wagner (1936), Fary (1948) and Stein (1951)


Graph
Topological embedding
Graph Drawing
Fáry's Theorem

Wikipedia proof of Fáry's Theorem:

- First note that it is enough to prove it for triangulations.



We prove Fáry's theorem by induction on the number of vertices.

If G has only three vertices, then it is easy.

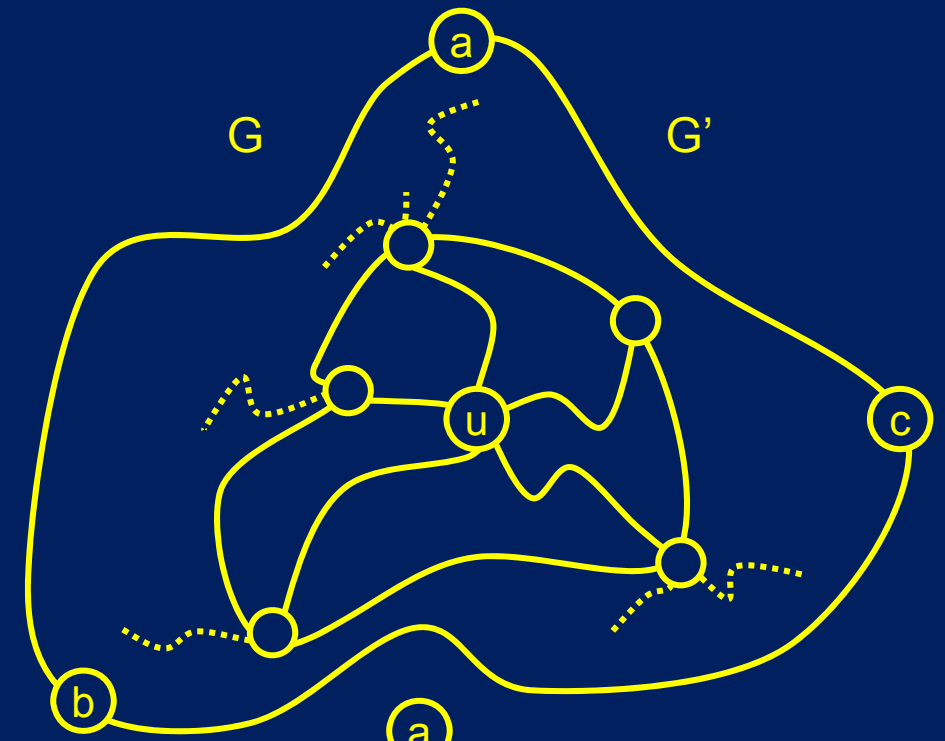
Suppose G has n>3 vertices and 3n-6 edges, and that the outer face of G is the triangle <abc>.

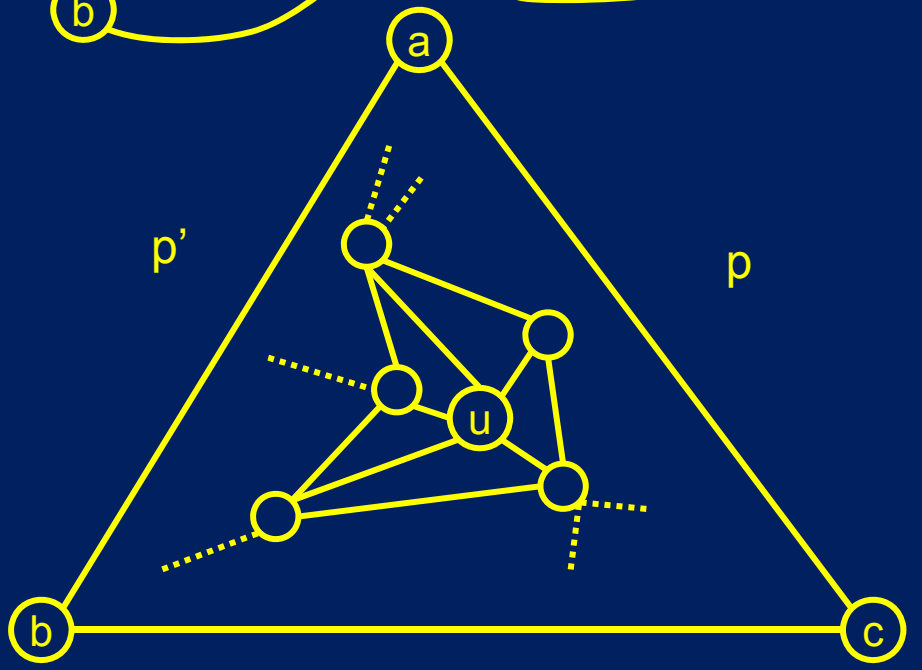
Since every vertex has degree at least 3, Euler implies that there is a vertex ***u*** not on the outside face with degree at most 5.

Delete ***u*** from G to form G'; this gives a face F of G of size at most 5.

Since G' has n-1 vertices, by induction it has a planar straight-line drawing p'.

Since F has at most 5 vertices, it is star-shaped, and we can place the vertex u in the kernel of F to give a planar straight-line drawing p of G





2. How to draw a graph?
   a) Before Tutte: 1920s – 1950s
   b) Tutte

W. T. Tutte,
"How to Draw a Graph",
Proceedings of the London Mathematical Society
13, pp743 – 767, 1960

Tutte's barycentre algorithm

Input:

- A graph G = (V,E)

Output

- A straight-line drawing p

Step 1. Choose a subset A of V

Step 2. Choose a location p(a) = (x_a, y_a) for each vertex a ∈ A

Step 3. For all u ∈ V-A, choose p(u) by

$$p(u) = \frac{1}{\deg(u)} \sum_{v \in N(u)} p(v)$$

where the sum is over all neighbors v of u

Vertex u is placed at the "*barycenter*" of its neighbors

Step 1. Choose a subset A of V

Step 2. Choose a location p(a) = (x_a, y_a) for each vertex a ∈ A

Step 3. For all u ∈ V-A,

$$p(u) = \frac{1}{\deg(u)} \sum_{v \in N(u)} p(v)$$

where the sum is over all neighbors v of u

$$x(u) = \frac{1}{\deg(u)} \sum_{v \in N(u)} x(v)$$

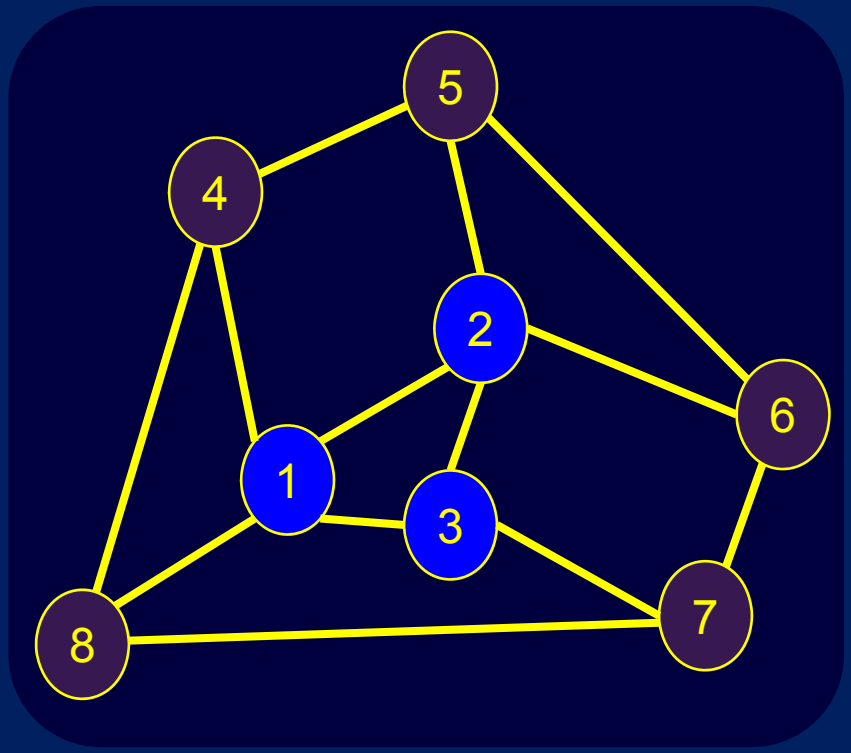
and

$$y(u) = \frac{1}{\deg(u)} \sum_{v \in N(u)} y(v)$$

2n equations

Tutte's barycenter algorithm

1. Choose a set A of vertices.
2. Choose a location p(a) for each a∈A
3. For each vertex ***u*** ∈V-A, place ***u*** at the barycentre of its graph-theoretic neighbors.



Example

Step 1. A = {4, 5, 6, 7, 8}

Step 2. For all i = 4, 5, 6, 7, 8,
choose x_i and y_i in some way.

Step 3. Find x_1, y_1, x_2, y_2, x_3, and y_3 such that:

$$x_1 = \frac{1}{4}(x_2 + x_3 + x'_4 + x'_8)$$

$$x_2 = \frac{1}{4}(x_1 + x_3 + x'_5 + x'_6)$$

$$x_2 = \frac{1}{3}(x_1 + x_2 + x'_7)$$

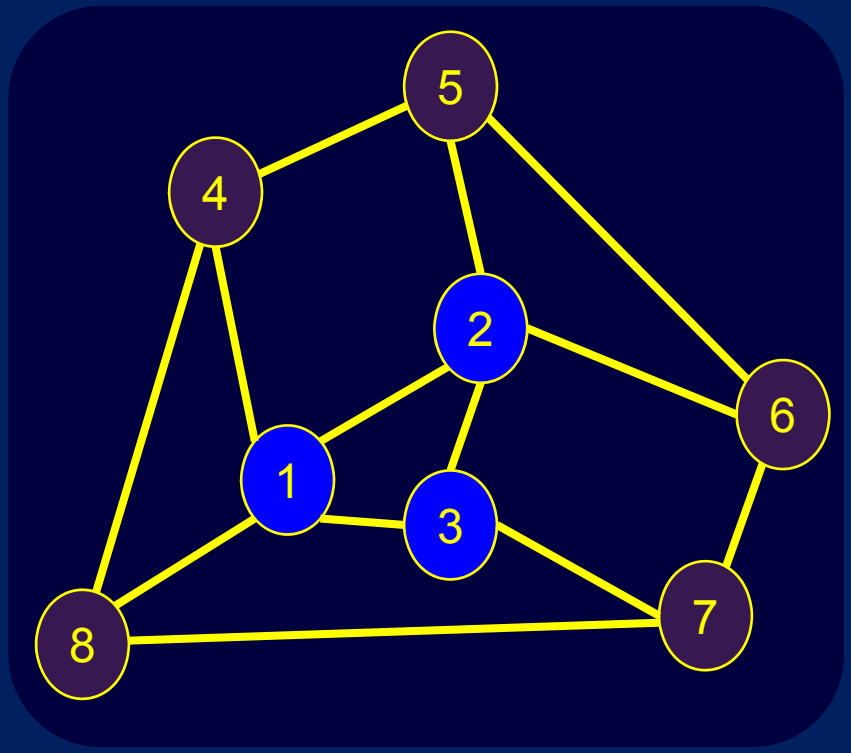
and

$$y_1 = \frac{1}{4}(y_2 + y_3 + y'_4 + y'_8)$$

$$y_2 = \frac{1}{4}(y_1 + y_3 + y'_5 + y'_6)$$

$$y_2 = \frac{1}{3}(y_1 + y_2 + y'_7)$$

where x'_i and y'_i are the values chosen at step 2.



Step 1. A = {4, 5, 6, 7, 8}

Step 2. For all i = 4, 5, 6, 7, 8, choose x_i and y_i in some way.

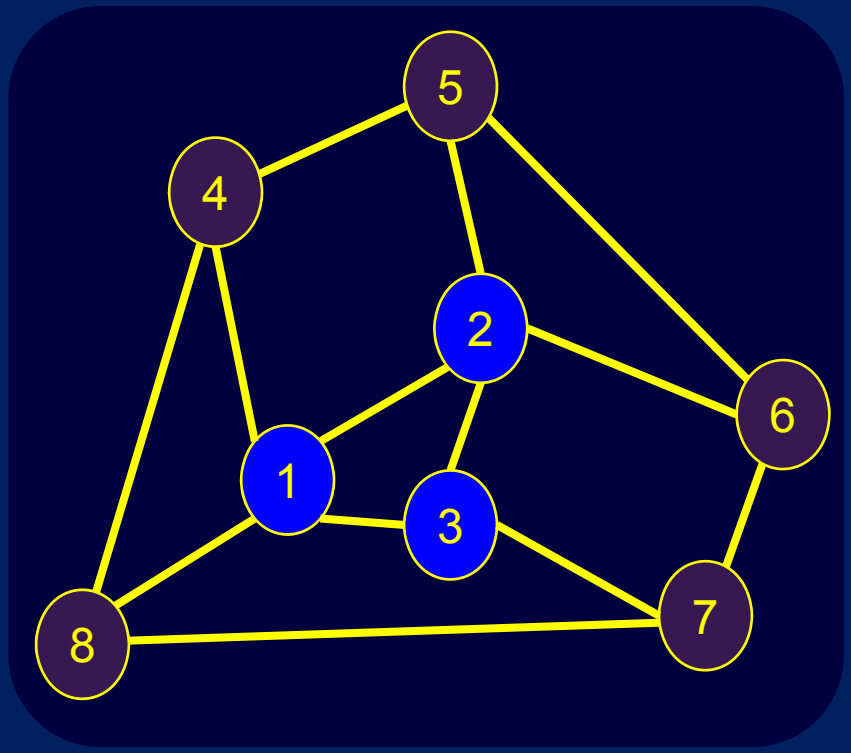
Step 3. Find x_1, y_1, x_2, y_2, x_3, and y_3 such that:

$$4x_1 - x_2 - x_3 = x'_4 + x'_8 = c_1$$
$$-x_1 + 4x_2 - x_3 = x'_5 + x'_6 = c_2$$
$$-x_1 - x_2 + 3x_3 = x'_5 = c_3$$

and

$$4y_1 - y_2 - y_3 = y'_4 + y'_8 = d_1$$
$$-y_1 + 4y_2 - y_3 = y'_5 + y'_6 = d_2$$
$$-y_1 - y_2 + 3y_3 = y'_5 = d_3$$

where c_1, c_2, c_3, d_1, d_2, d_3 are constants



Step 1. A = {4, 5, 6, 7, 8}

Step 2. For all i = 4, 5, 6, 7, 8, choose x_i and y_i in some way.

Step 3. Find vectors $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ and $y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$ such that

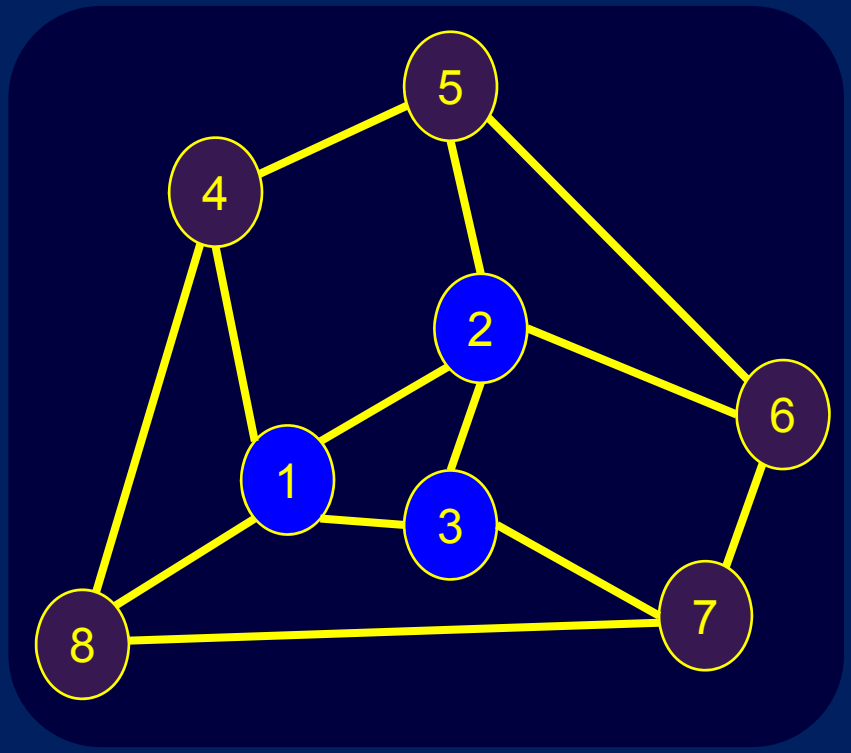
$$Mx = c$$

and

$$My = d$$

where

$$M = \begin{bmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 3 \end{bmatrix}$$



The essence of Tutte's barycentre algorithm is inverting a matrix

Tutte's barycentre algorithm

Input: A graph G = (V,E)

Output: A straight-line drawing p

Step 1. Choose a subset A of V

Step 2. Choose a location p(a) = (x_a, y_a) for each vertex a ∈ A

Step 3. Let M be the matrix, indexed by V-A, defined by

$$M_{uv} = \begin{cases} \deg(u) \text{ if } u = v \\ -1 \text{ if } (u,v) \epsilon E \\ 0 \text{ otherwise} \end{cases}$$

Let c and d be the vectors, indexed by V-A, de

$$c_u = \sum_{w \in A} x_w, \quad \text{and} \quad d_u = \sum_{w \in A} y_w.$$

Then choose x and y to be the vectors, indexed over V-A, defined by

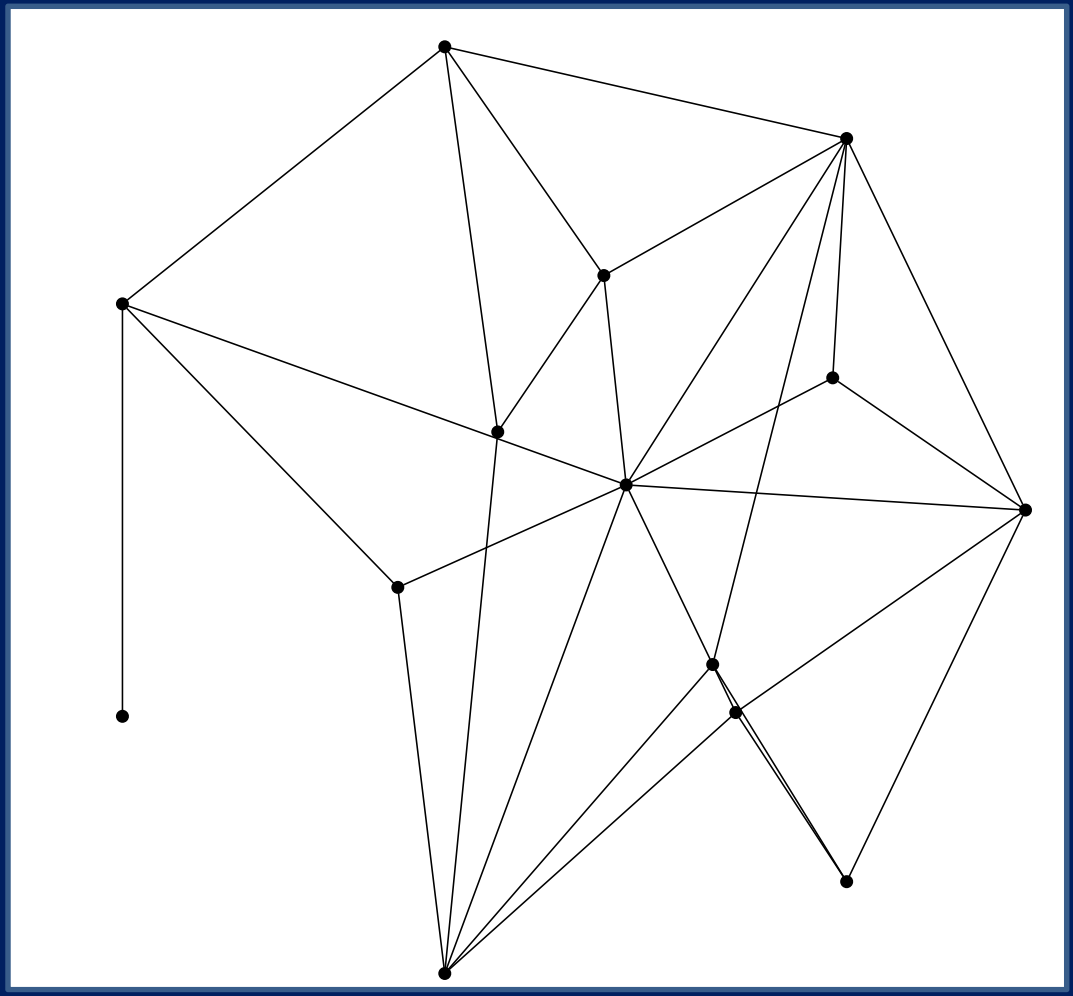
$$x = M^{-1}c \quad \text{and} \quad y = M^{-1}d.$$

Then choose $p(u) = (x_y, y_u)$ for all $u \in V - A$.

- This is a special matrix: *Laplacian submatrix*.
- Many software packages can invert such a matrix efficiently

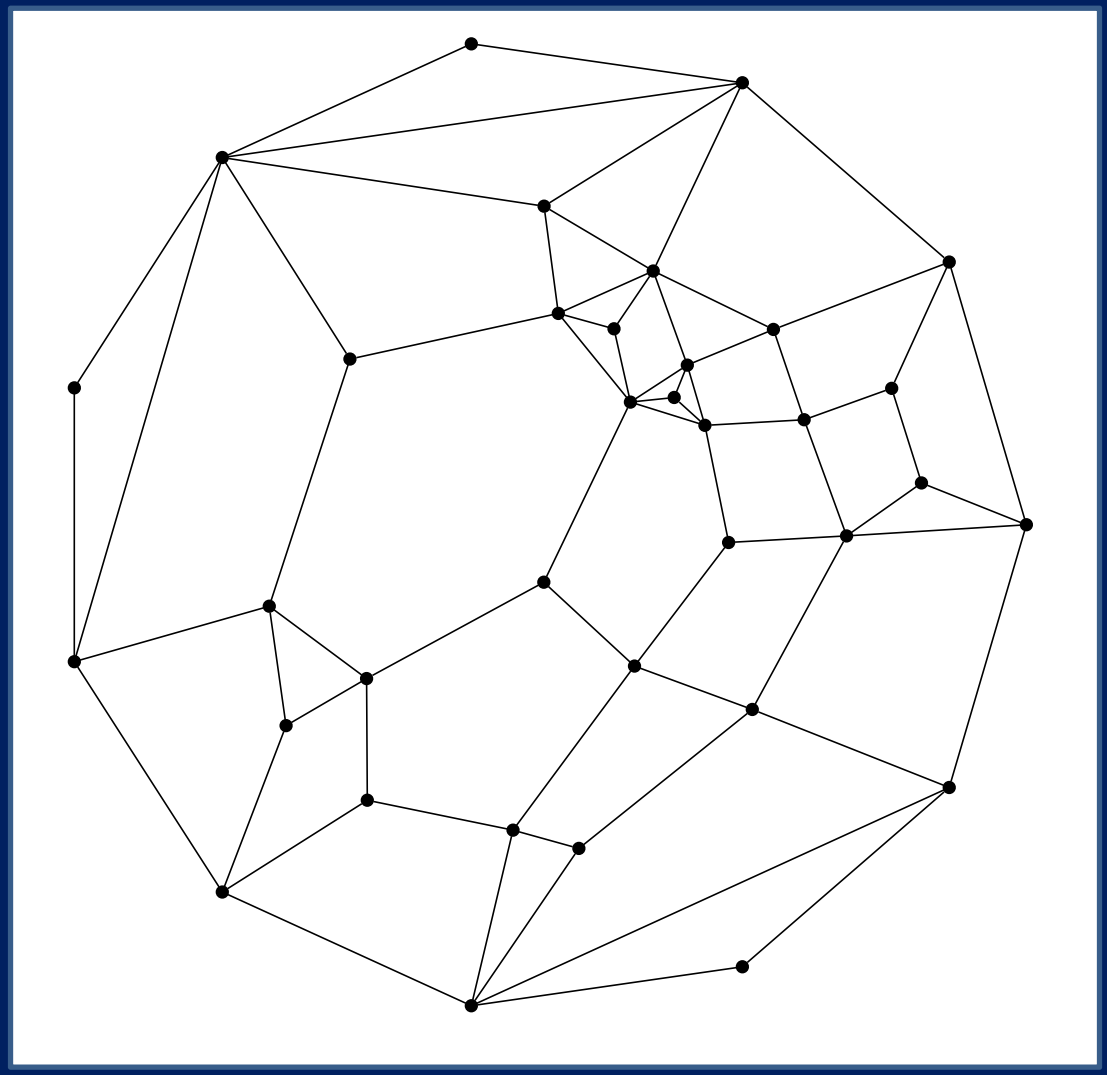
Tutte's barycentre algorithm

- Example output on a non-planar graph



Tutte's barycentre algorithm

- Example output on a *planar* graph



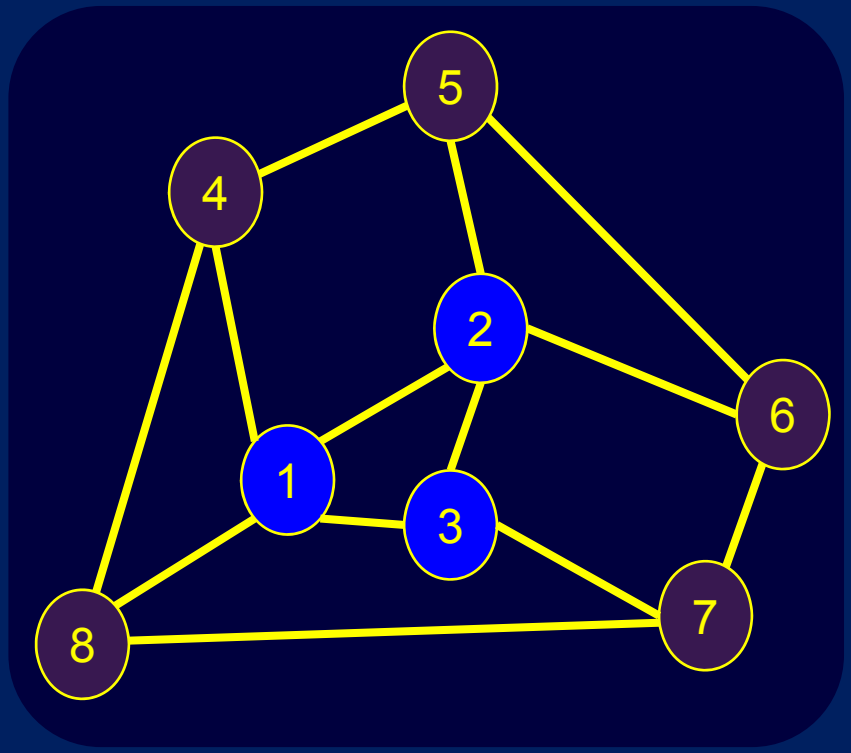
Tutte's barycenter algorithm for
triconnected planar graphs **…..**

Tutte's barycenter algorithm **for triconnected planar graphs**

1. Choose A to be *the outside face of the topological embedding*.

2. Choose the location p(a) for each a∈A to be *at the vertices of a convex polygon*.

3. For each vertex ***u*** ∈V-A, place ***u*** at the barycentre of its graph-theoretic neighbors.



Note: For planar graphs, the Laplacian matrix is sparse, and can be inverted fast.

Tutte's amazing theorem (1960)

If the input graph is planar and triconnected, then

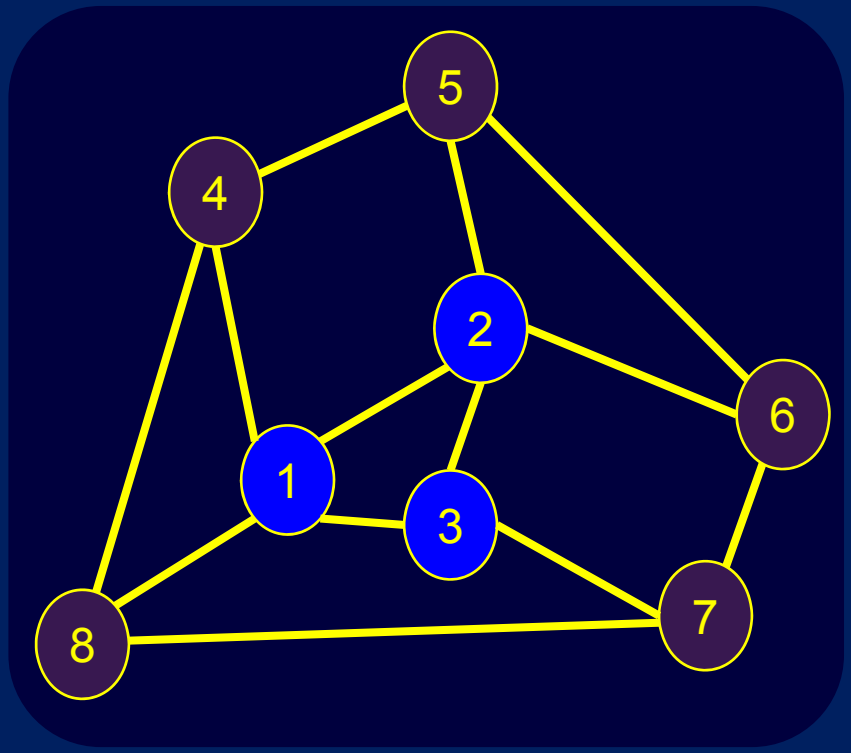
- the drawing output by the barycentre algorithm is planar, and
- every face is convex.

The *energy* view of Tutte's barycentre algorithm

Tutte's barycenter algorithm:

The energy view

1. Choose a set A of vertices.
2. Choose a location p(a) for each a∈A
3. *Place all the other vertices to minimize energy.*



What is the *energy* of a drawing p?

- The Euclidean distance between u and v in the drawing p is:
$$d(u,v) = \sqrt{(x_u - x_v) + (y_u - y_v)^2}$$
- The *energy in the edge (u,v)* is $d(u,v)^2 = (x_u - x_v)^2 + (y_u - y_v)^2$
- The *energy in the drawing p* is the sum of the energy in its edges, ie,

$$energy(p) = \sum_{(u,v)\in E} d(u,v)^2 = \sum_{(u,v)\in E} (x_u - x_v)^2 + (y_u - y_v)^2$$

Tutte's barycenter algorithm:

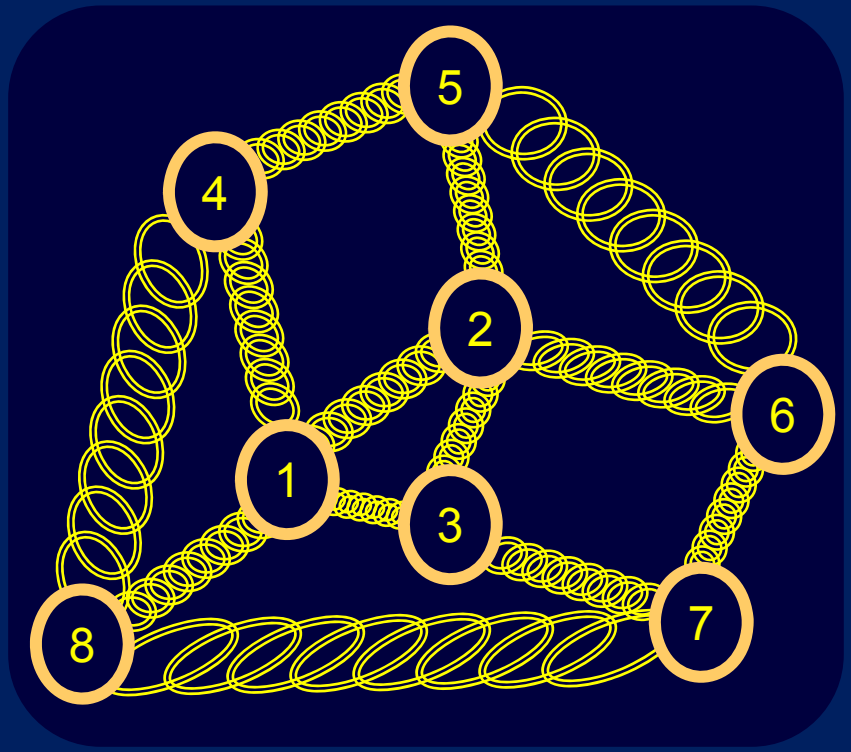
The energy view

We represent each vertex by a steel ring, and represent each edge by a spring of natural length zero connecting the rings at its endpoints.

1. Choose a set A of vertices.

2. For each a∈A, nail the ring representing a to the floor at some position.

3. The vertices in V-A will move around a bit; when the movement stops, take a photo of the layout; this is the drawing.



How to minimize energy:

- Choose a location $p(u) = (x_u, y_u)$ for each $u \in V - A$ to minimize

$$energy(p) = \sum_{(u,v)\in E} d(u,v)^2 \quad = \sum_{(u,v)\in E} (x_u - x_v)^2 + (y_u - y_v)^2$$

- Note that the minimum is unique, and occurs when

$$\frac{\partial\,(energy(p))}{\partial x_u} = 0 \quad \text{and} \quad \frac{\partial\,(energy(p))}{\partial y_u} = 0$$

for each $u \in V - A$.

For x_u, the minimum occurs when:

$$\frac{\partial\,(energy(p))}{\partial x_u} = 0$$

$$\frac{\partial}{\partial x_u}\left(\sum_{(u,v)\in E} (x_u - x_v)^2 + (y_u - y_v)^2\right) = 0$$

$$\sum_{v\in V\ s.t.(u,v)\in E} 2(x_u - x_v) = 0$$

$$x_u = \frac{1}{\deg(u)} \sum_{v\in V\ s.t.(u,v)\in E} x_v$$

Barycentre equations

How good is Tutte's barycentre algorithm?

Efficiency:

- In theory it is not bad: $O(n^{1.5})$ for planar graphs
- In practice it is fast, using numerical methods

Elegance:

- Very simple algorithm
- Easy to implement
- Numerical software available for the hard parts

Effectiveness:

- Planar graphs drawn planar
- Convex faces
- Straight-line edges

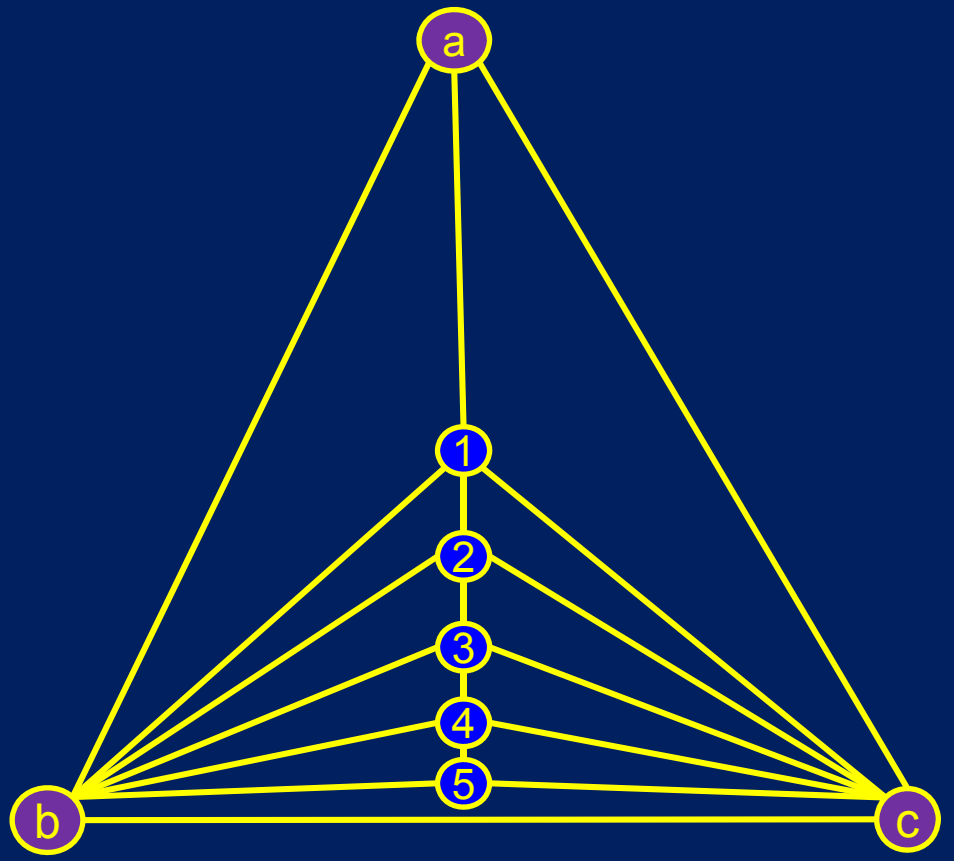
But unfortunately →

But unfortunately:

- Tutte's algorithm gives *poor vertex resolution* in many cases

Example:

- Vertex a is at (0.5, 0)
- b is at (0,0)
- c is at (1,0).
- Suppose that vertex j is at (x_j, y_j) for $0<j<n$.



From the barycentre equations:

$$\begin{cases} x_j = 0.5 \\ y_j = \frac{1}{4}(y_{j-1} + y_{j+1}) \end{cases}$$

One can deduce that

$$|y_{j-1} - y_j| < \frac{1}{2^j}$$

That is, some vertices are exponentially close together.

Aside:

- Commercial graph drawing software needs good resolution.

How good is Tutte's barycentre algorithm?

Efficiency:

- OK

Elegance:

- Excellent

Effectiveness:

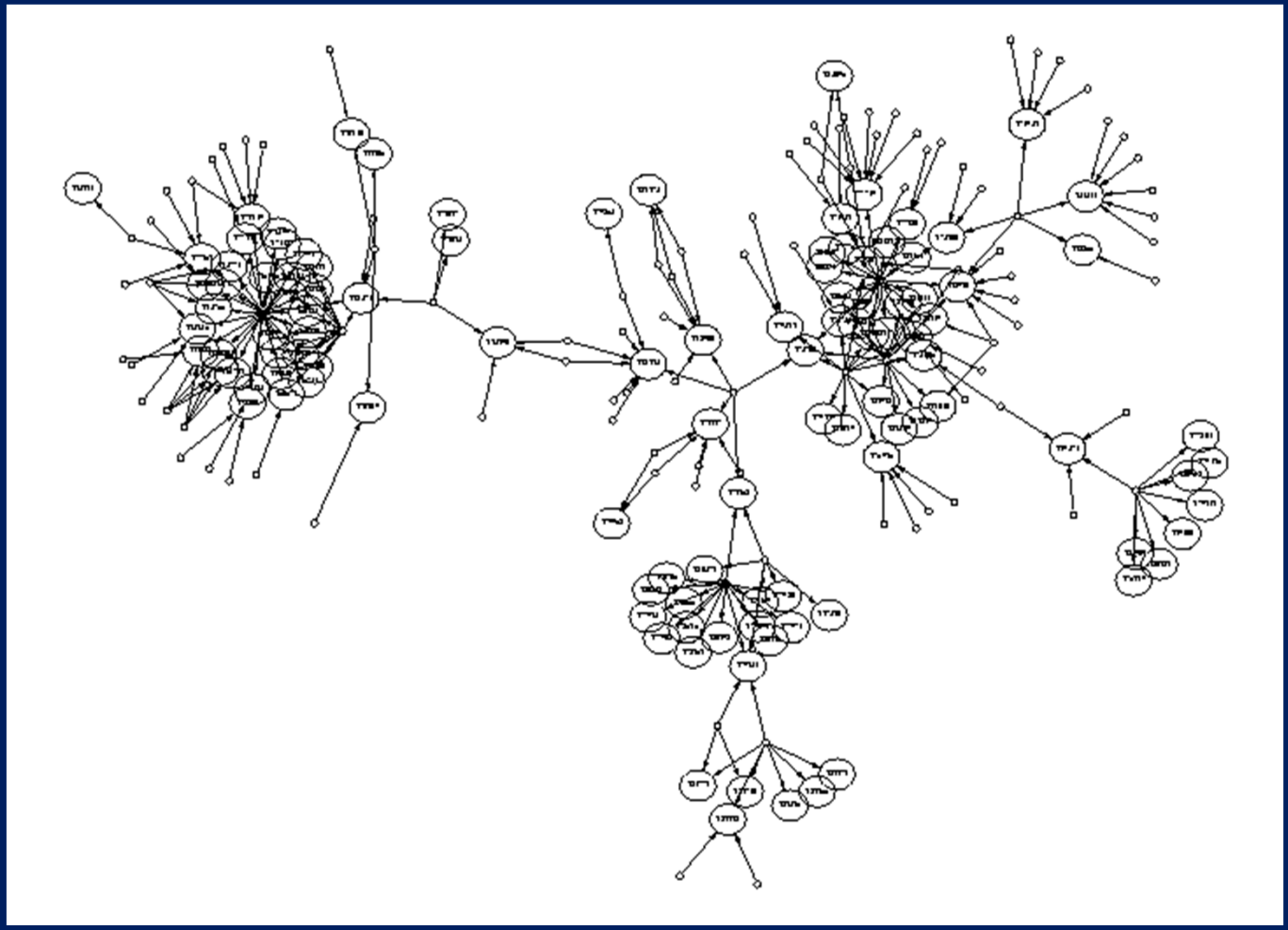
- So-so

2. How to draw a planar graph?
   a) Before Tutte: 1920s – 1950s
   b) Tutte: 1960s
   c) After Tutte: 1970s – 1990s

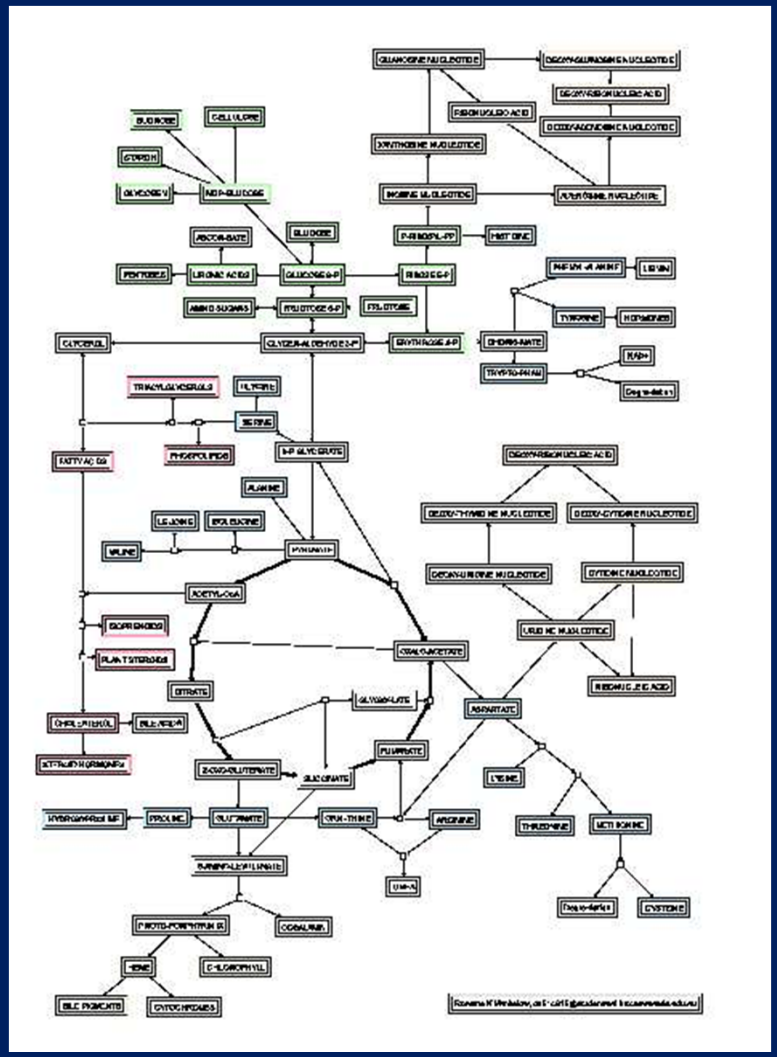
After Tutte: 1970s – 1990s

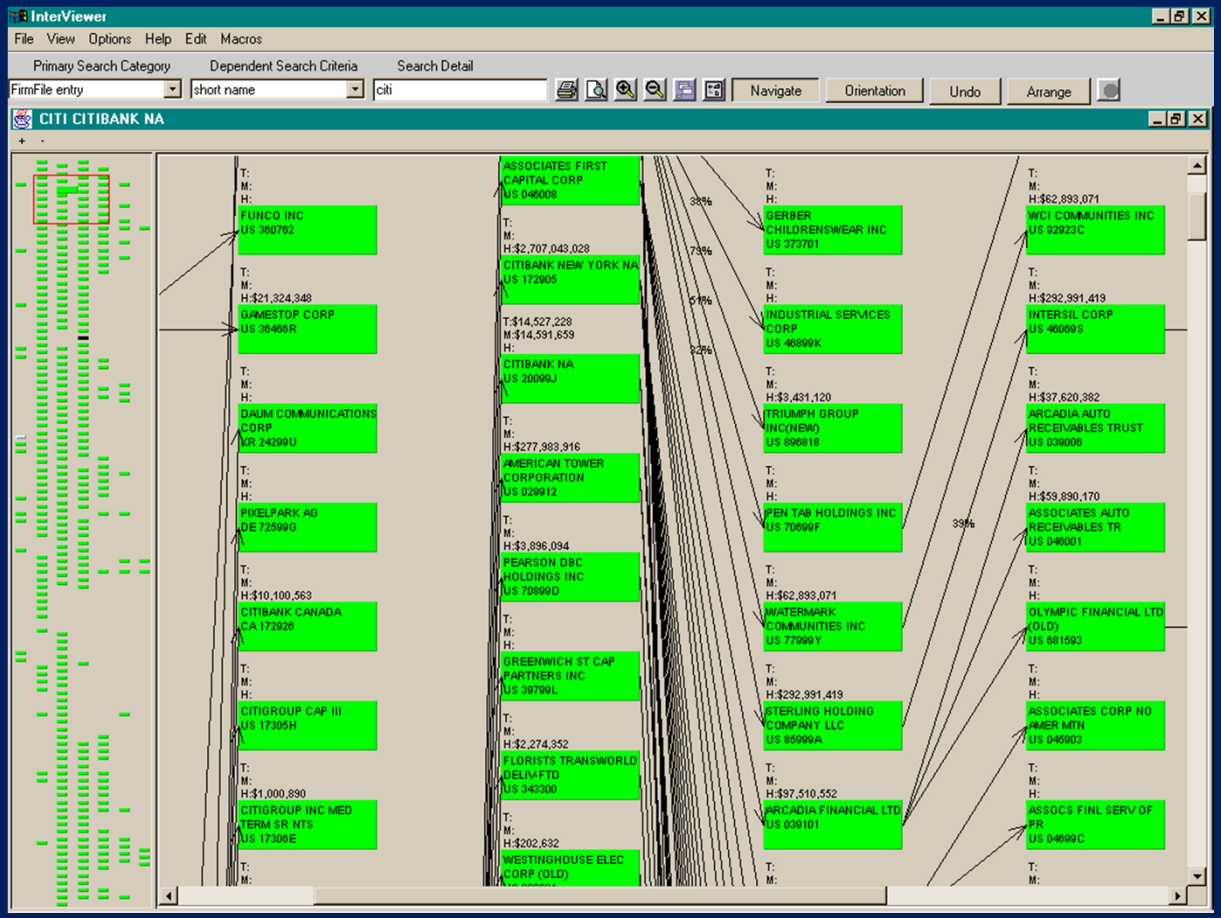
Sometime in the 1980s, the motivation for graph drawing changed from *Mathematical curiosity* to *visual data mining*.

Software



Biology





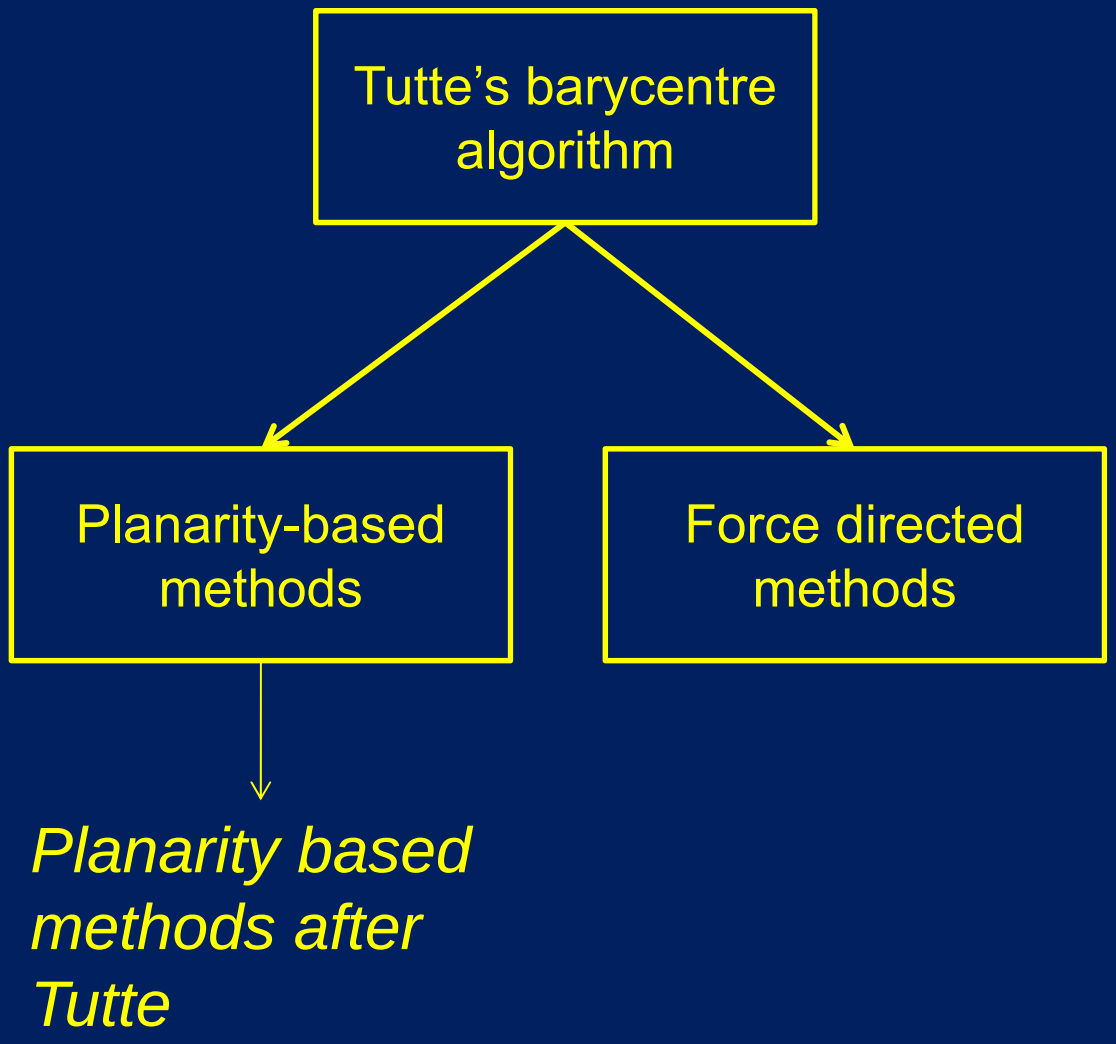
Risk Exposure

From the 1980s, industrial demand for graph drawing algorithms has grown

- Software engineering: CASE systems, reverse engineering
- Biology: PPI networks, gene regulatory networks
- Physical networks: network management tools
- Security: risk management, money movements
- Social network analysis
- Customer relationship management: value identification

Many companies buy graph drawing algorithms, many code them.

Currently the international market for graph drawing algorithms is in the hundreds of millions of dollars per year.


Tutte's barycentre algorithm
Planarity-based methods
Force directed methods
Planarity based methods after Tutte

Planarity based methods after Tutte

R.C. Read (1979, 1980)

1. Efficient?
   - Yes, linear time algorithm

2. Elegant?
   - Yes: follows proof of Fáry's theorem, with some tricks to make it into an efficient algorithm

3. Effective?
   - Maybe ...
     - Straight-line planar drawings of planar graphs
     - But unfortunately, output has poor vertex resolution

Planarity based methods after Tutte

Chiba-Nishizeki-Yamanouchi (1984)

1. Efficienct?
   - Yes, linear time algorithm

2. Elegant?
   - Yes, a simple divide&conquer approach

3. Effective?
   - Maybe ...
     - Straight-line planar drawings of planar graphs
     - Convex faces for well connected input

     - But unfortunately, output has poor vertex resolution

Planarity based methods after Tutte

Breakthrough in 1989:

de Fraysseix-Pach-Pollack Theorem (1989)
Every planar graph has a planar straight-line drawing with vertices located on a *2n X 4n* integer grid.

Notes:

- Good resolution: the minimum distance between vertices is at least *screensize/4n*.
- Linear time (Chrobak, 1990)

The deFraysseix-Pach-Pollack Theorem gave much hope for planarity-based methods, and many refinements appeared 1990 – 2000.

de Fraysseix-Pach-Pollack-Chrobak Algorithm

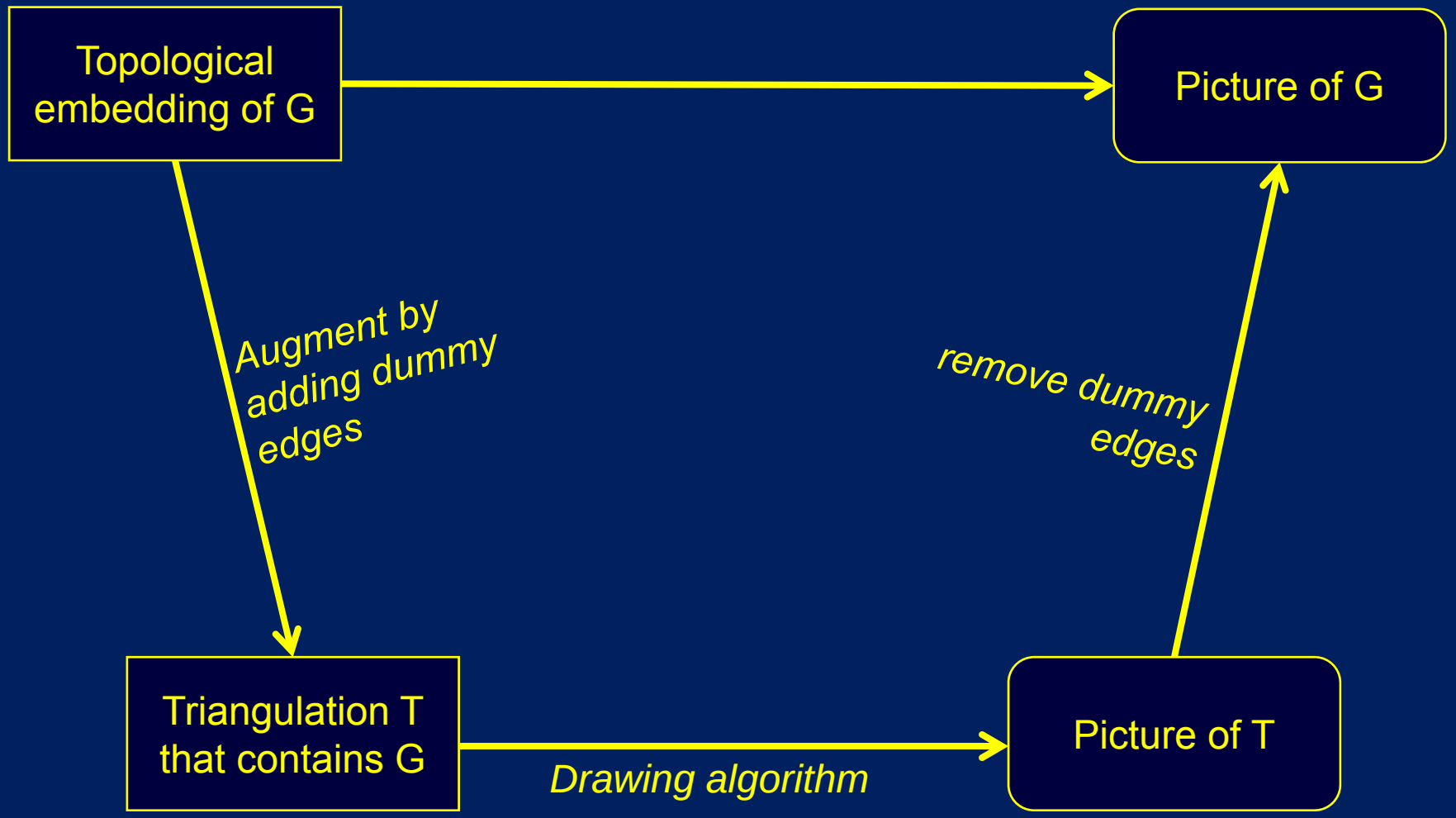
1. Add dummy edges to make the graph into a triangulation

2. Construct an ordering $u_1, u_2, \ldots, u_n$ of the vertices, called the *canonical ordering*.

3. Draw the graph, adding one vertex at a time, in order $u_1, u_2, \ldots, u_n$

Wikipedia proof of Fáry's Theorem

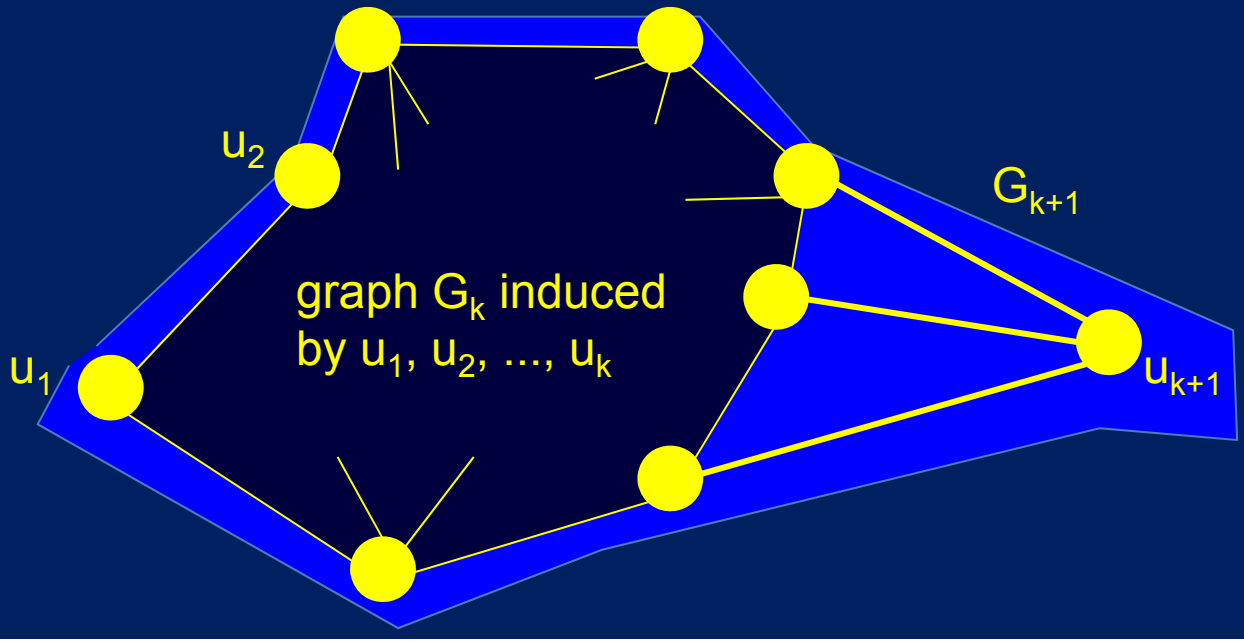
Step 1: Add dummy edges to make the graph into a triangulation



Step 2: Construct an ordering $u_1, u_2, \ldots, u_n$ of the vertices , called the "canonical ordering".

A *canonical ordering* is an ordering $u_1, u_2, ..., u_n$ of the vertices of a triangulation having the property that, for each $3 \leq k < n$, the graph G_k induced by $u_1, u_2, ..., u_k$ has the following properties

- G_k is biconnected
- G_k contains the edge (u_1, u_2) on its outer face,
- Any vertices in G_k adjacent to u_{k+1} are on the outer face of G_k
- The vertices in G_k adjacent to u_{k+1} form a path along the outer face of G_k

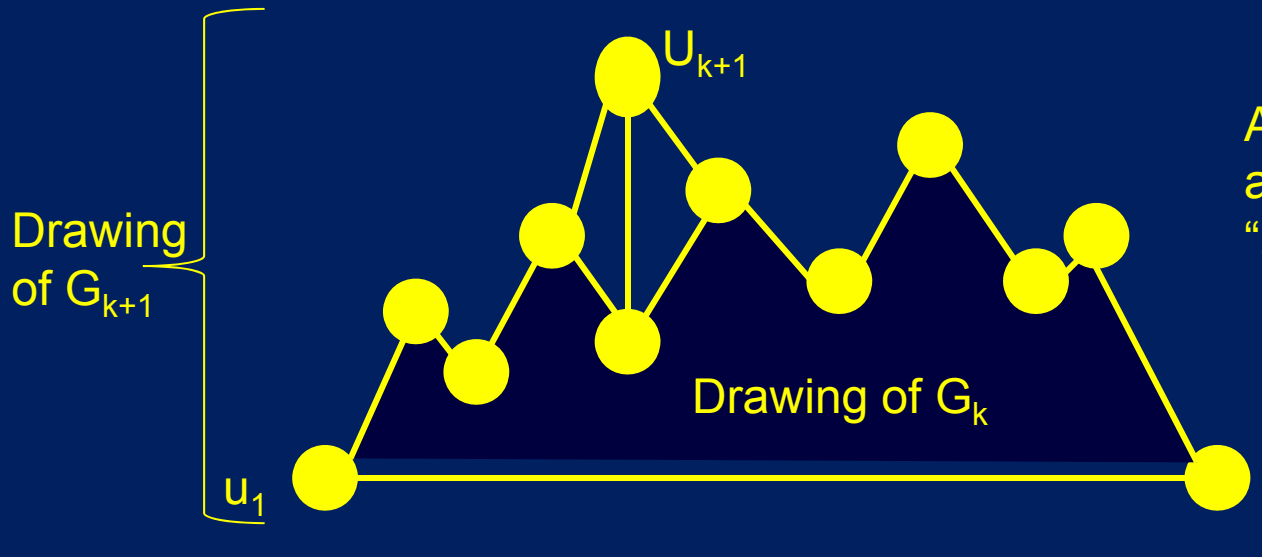


Step 3: Draw the graph, adding one vertex at a time in order $u_1, u_2, \ldots, u_n$

a) Start with the edge (u_1, u_2) at y=0

b) For each k>1:

- add u_{k+1} on y=k
- Choose x coordinate of u_{k+1} so that there are no edge crossings.



At each stage, there is a drawing of G_k as a *"terrain"*.

Some details of deFraysseix-Pach-Pollack-Chrobak algorithm are needed to show

- It runs in linear time
- It is possible to avoid edge crossings
- Each vertex lies on an integer grid of size at most 4nX2n

Also:

For restricted classes of planar graphs, there are algorithms with better resolution, for example:

- Outerplanar graphs: algorithms with output with area $\min(k_1 n^{1.48}, k_2 dn \log n)$ (Frati 2009)

And some lower bounds:

- Planar graphs: area $\Omega(n^2)$ is necessary (Leighton(?), about 1970)
- Series-parallel graphs:$\Omega\left(n2^{\sqrt{\log n}}\right)$ is necessary (Frati, 2010)

The deFraysseix-Pach-Pollack-Chrobak algorithm

Efficiency:

- Yes, linear time

Elegance:

- Not bad; can be coded by a student in a week or so.

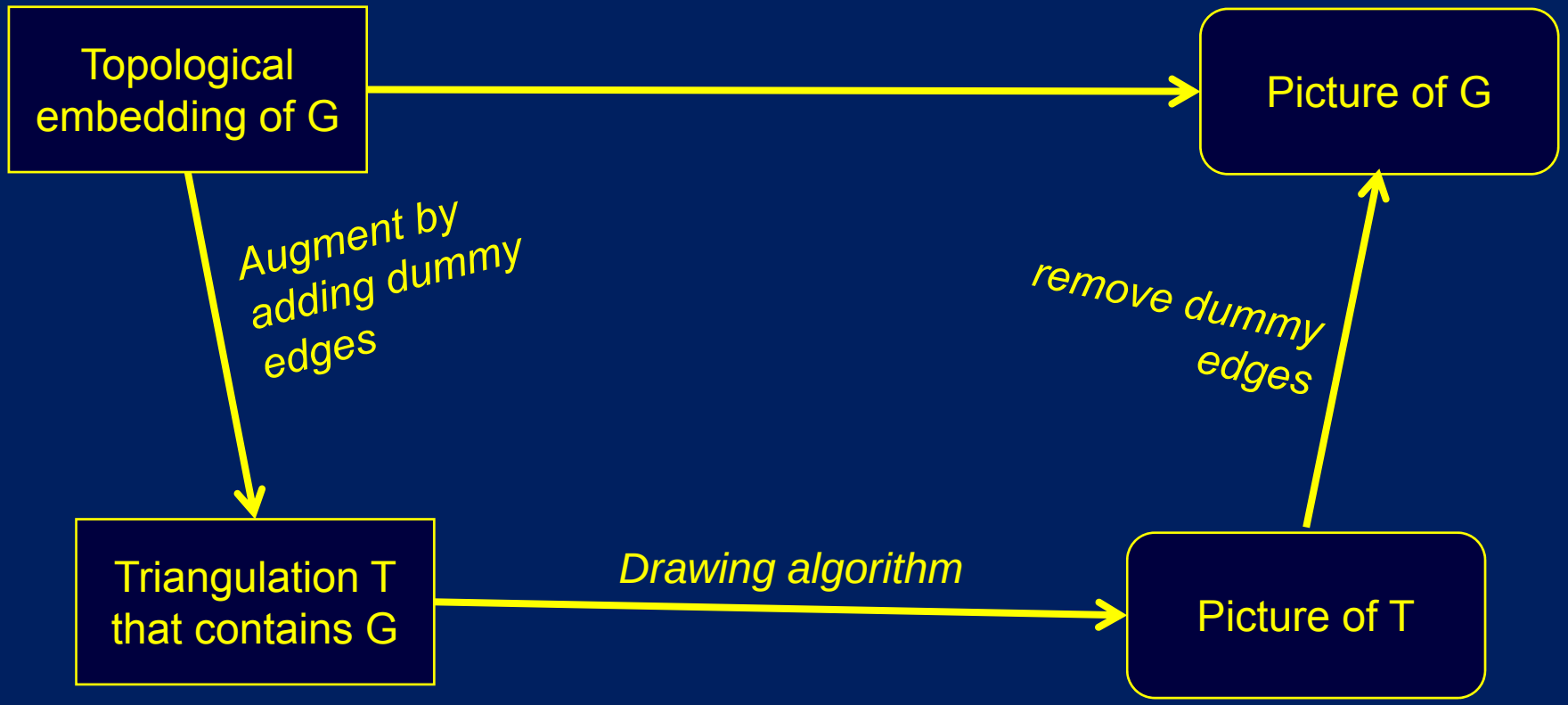
Effectiveness:

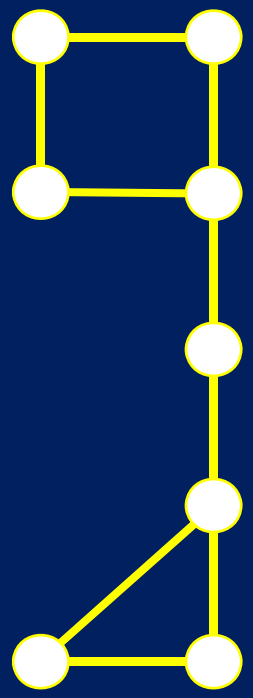
- Maybe pretty good:
  - Straight-line edges
  - No edge crossings
  - Good vertex resolution

  - But unfortunately -->

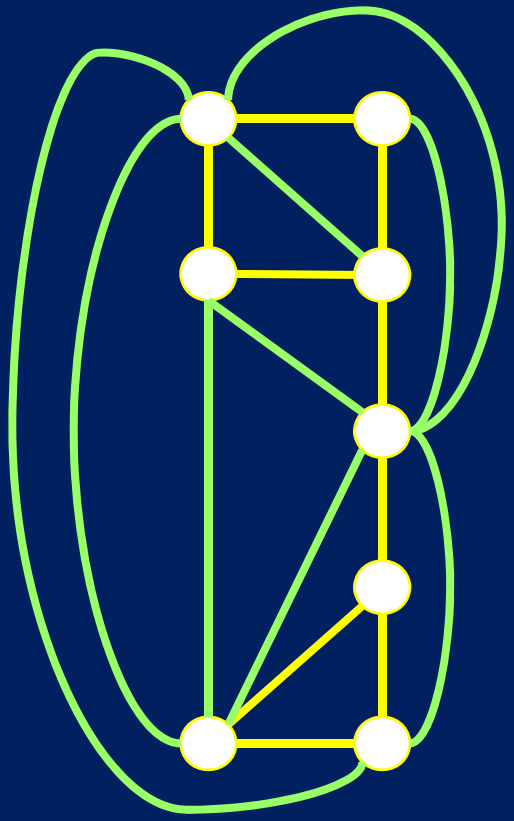
The deFraysseix-Pach-Pollack-Chrobak algorithm gave much hope for planarity-based methods, and many refinements appeared 1990 – 2000.

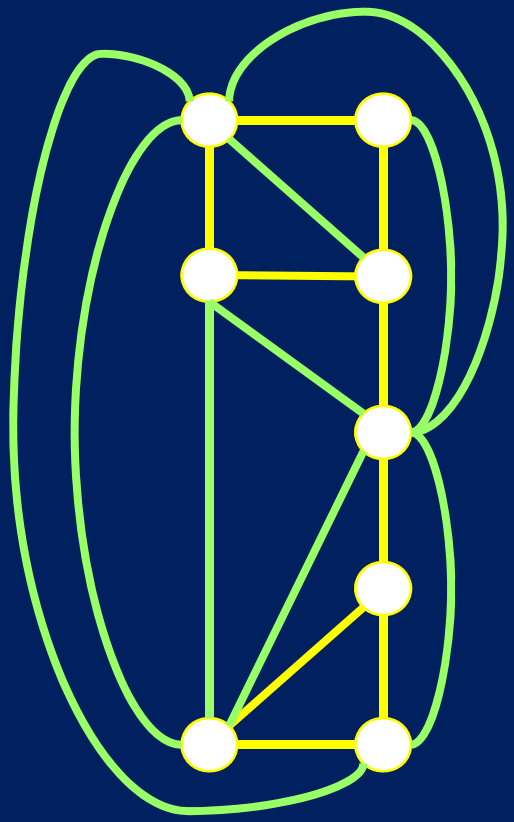
But unfortunately, we found that the first step (increasing connectivity by triangulation) gives some problems.



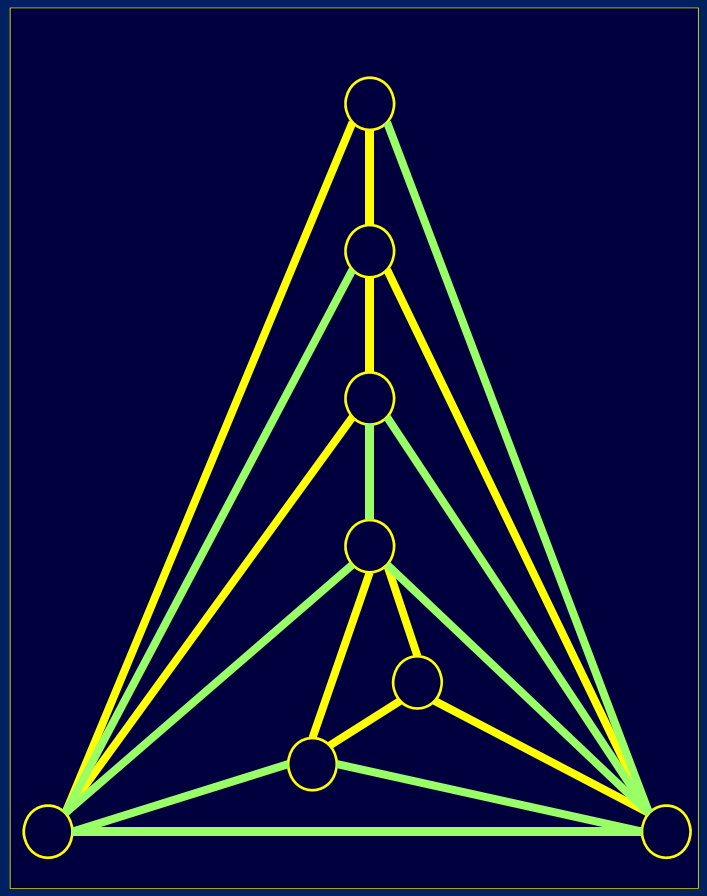


1. Add dummy edges to triangulate

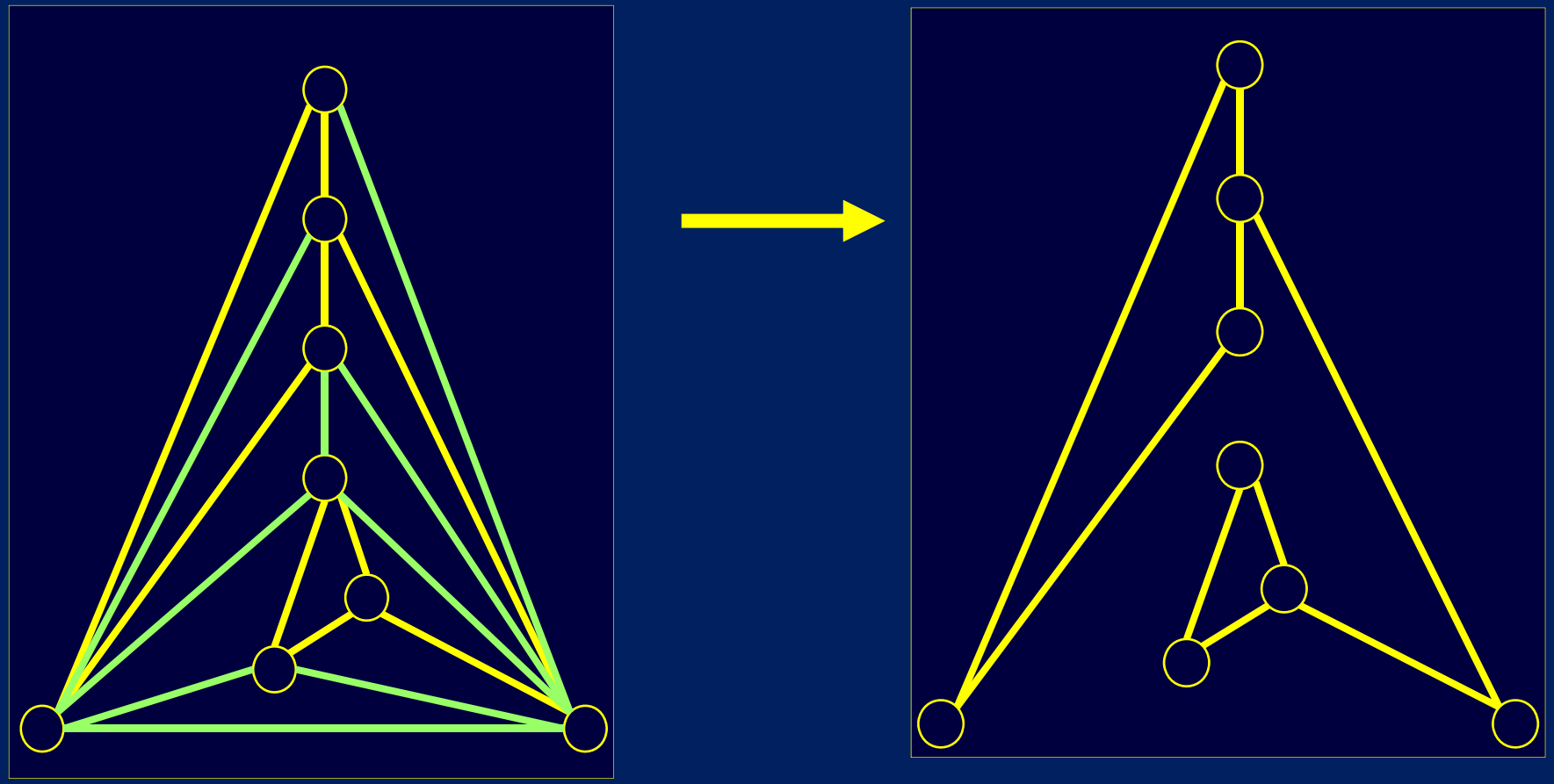




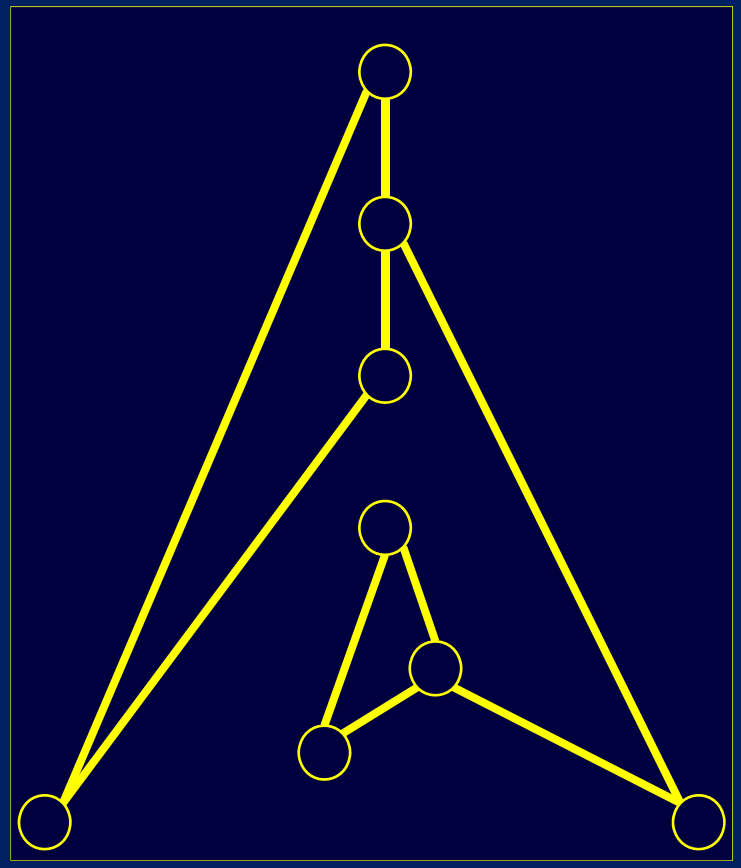
2.Draw the augmented graph.



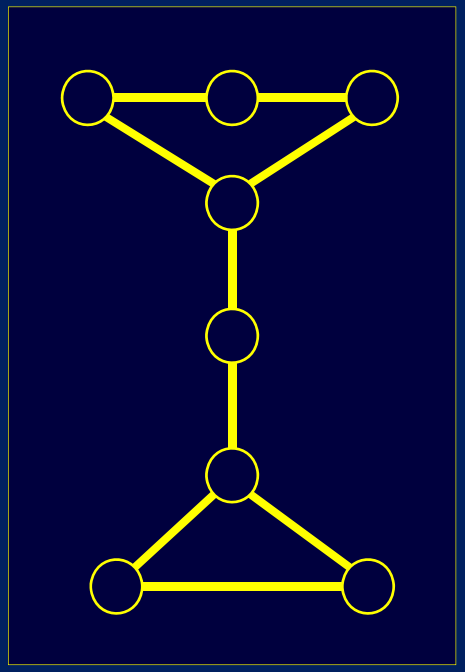
3. Delete the dummy edges



Note: the resulting drawing is ugly.



A better drawing



This kind of ugly drawing is a typical output from methods that use augmentation to increase connectivity

The deFraysseix-Pach-Pollack-Chrobak algorithm

Efficiency:

- Yes, linear time

Elegance:

- Not bad; can be coded by a student in a week or so.

Effectiveness:

- So-so
  - Straight-line edges
  - No edge crossings
  - Good vertex resolution
  - Augmentation step gives poor angular resolution and weird shapes

Aside

What if the input graph is non-planar?

Classical approach: *planarization*

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

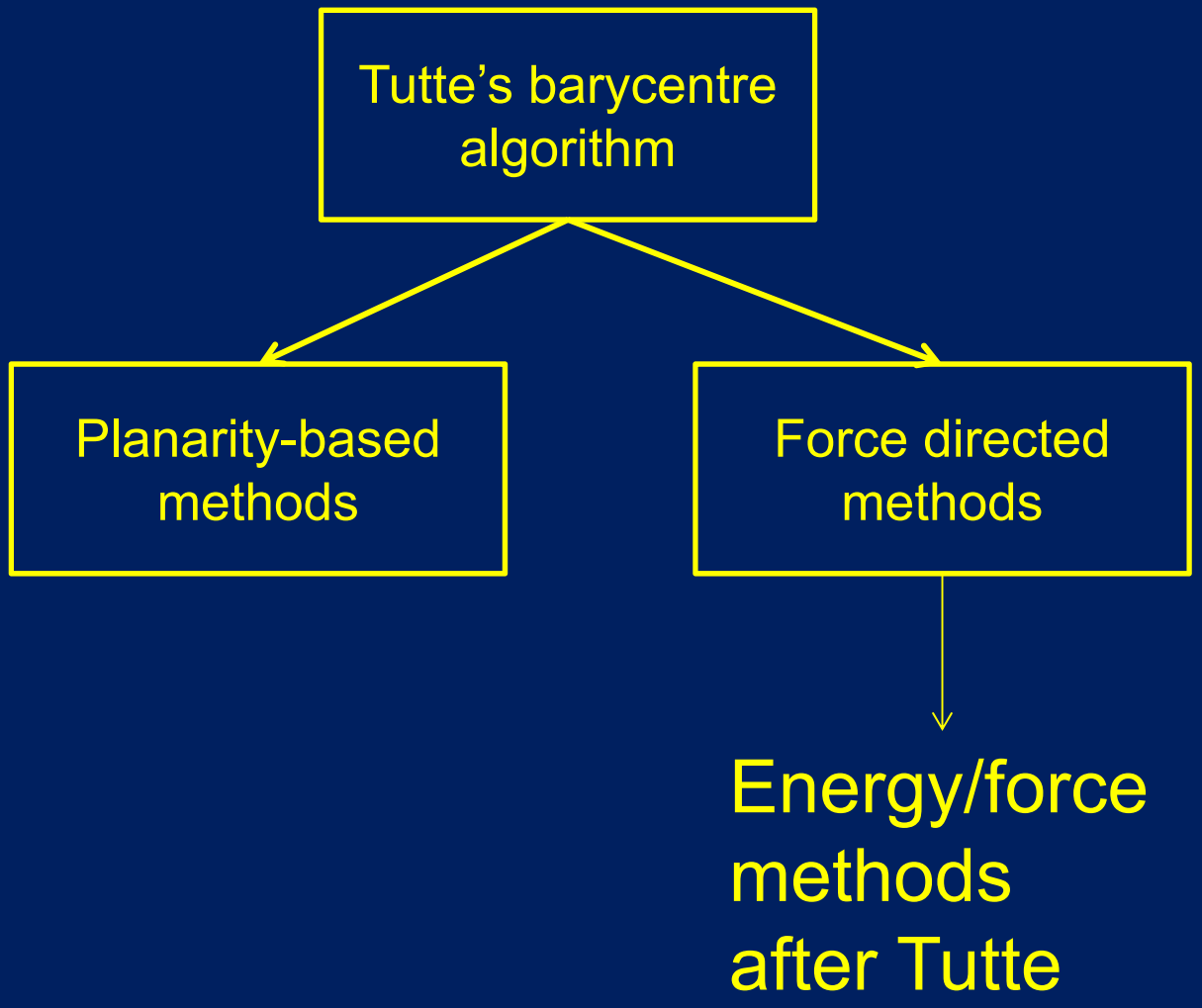
1. Find an approximately maximal planar subgraph $G' = (V, E')$ with $E' \subseteq E$.
2. Find a topological embedding of G'.
3. Route each edge $e \in E - E'$ in such a way to locally minimise crossings.
4. Put dummy vertices at each crossing point, to give a vertex set V' with $V \subseteq V'$.
5. The graph $G'' = (V', E'')$ is then planar. Use a planarity-based drawing algorithm to draw G''.

Current state-of-the-art for planarity based methods:

- There are many small improvements to the deFraysseix-Pach-Pollack-Chrobak algorithm.
- But none have overcome all the connectivity augmentation problem.

Current state-of-the-art for planarity based methods:

- *Almost no planarity based methods* have been adopted in commercial software … despite the fact that planarity is the single most important aesthetic criterion.


Tutte's barycentre algorithm
Planarity-based methods
Force directed methods
Energy/force methods after Tutte

Tutte's algorithm gives poor vertex resolution: vertices can be exponentially close to each other.

To solve this problem, we need to prevent vertices from becoming very close together.

This can be done with forces:-

1. Use springs of nonzero natural length

2. Use an inverse square law repulsive force between nonadjacent vertices.

Force exerted by a vertex v on a vertex u:

If u and v are adjacent:

$$f_{spring}(u,v) = k_{uv}(d(u,v) - q_{uv})i_{uv}$$

where

- k_{uv} is the *strength* of the spring between u and v
- $d(u,v)$ is the Euclidean distance between u and v
- q_u is the *natural length* of the u-v spring
- i_{uv} is a unit vector in the direction from u to v.

If u and v are not adjacent:

$$f_{nonadjac}(u,v) = \frac{r_{uv}}{d(u,v)^2} i_{uv}$$

where

- r_{uv} is the *strength* of the repulsive force

Total force on a vertex u:

$$F(u) = \sum_{(u,v)\in E} f_{spring}(u,v) + \sum_{(u,w)\notin E} f_{nonadjac}(u,w).$$

A (locally) minimum energy configuration satisfies

$$F(u) = 0$$

for each vertex u.

This is a system of *nonlinear* equations.

Note

1. In general, the solution to this system of equations is *not unique*, that is, there are local minima that may not be global.
2. Many methods to solve systems of equations like this are available. Some methods are fast, some are slow, depending on the specific equations.

Force-based techniques can be constrained in various ways.

The constants in the force definitions

$$f_{spring}(u,v) = k_{uv}(d(u,v) - q_{uv})i_{uv}$$

$$f_{nonadjac}(u,v) = \frac{r_{uv}}{d(u,v)^2} i_{uv}$$

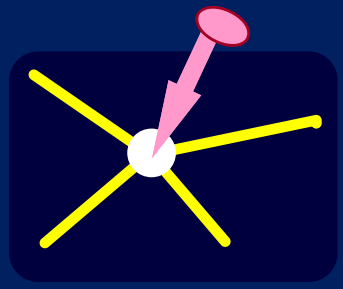
can be chosen to reflect the relationships in the domain.

For example

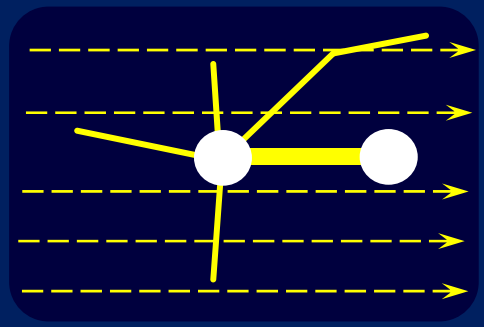
- If the edge between *u* and *v* is important in the domain, then we can choose k_{uv} to be large and q_{uv} to be small.

Force-based techniques can be constrained in various ways.

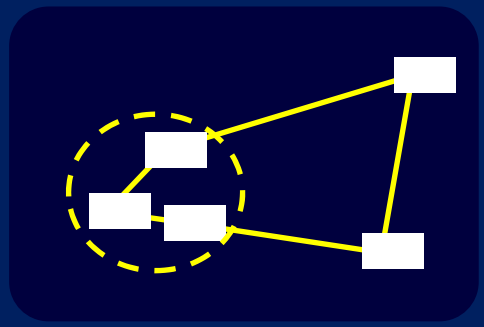
Nails can be used to hold a node in place.



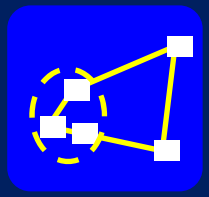
Magnetic fields and *magnetized springs* can be used to align nodes in various ways.

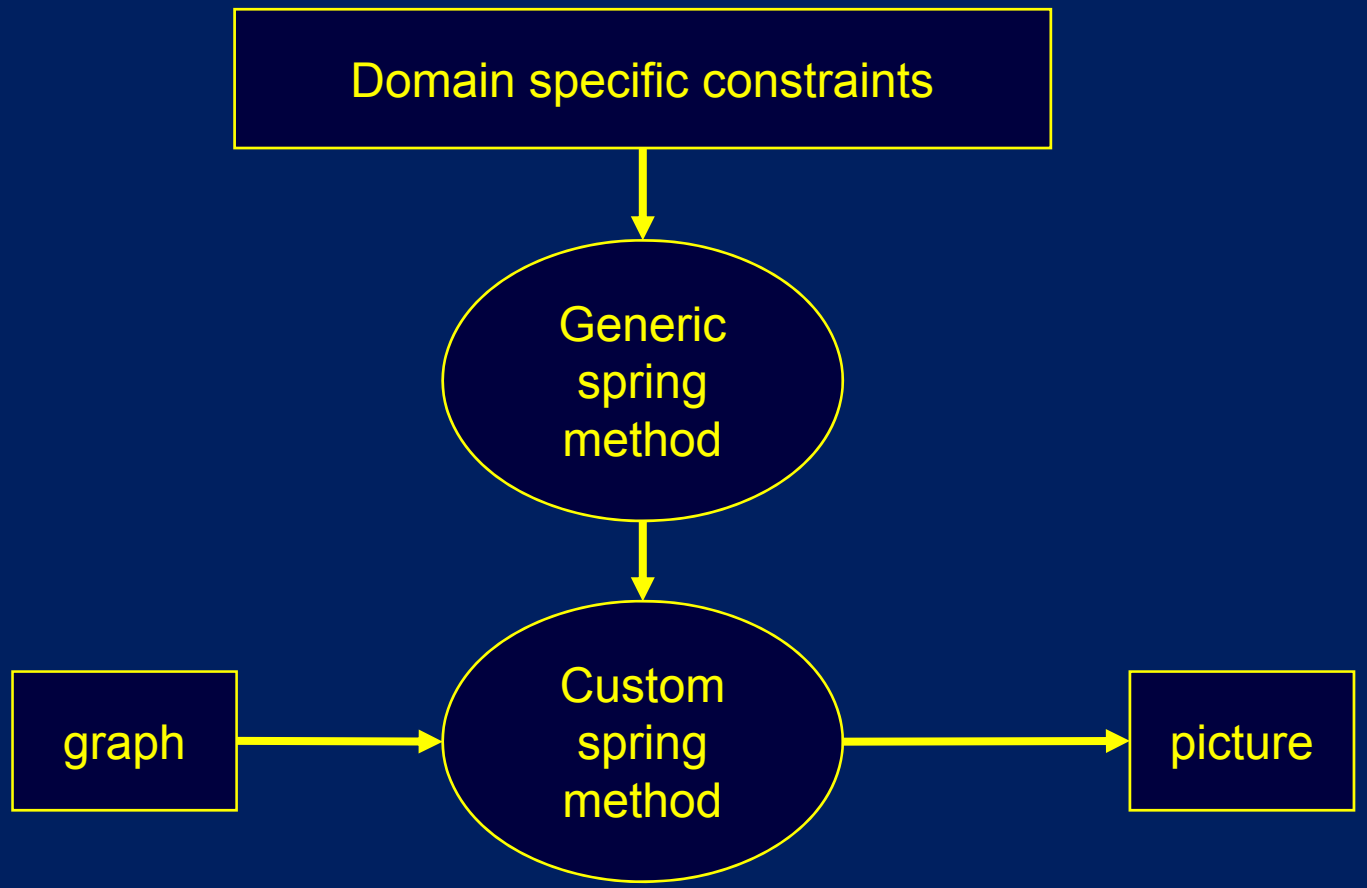


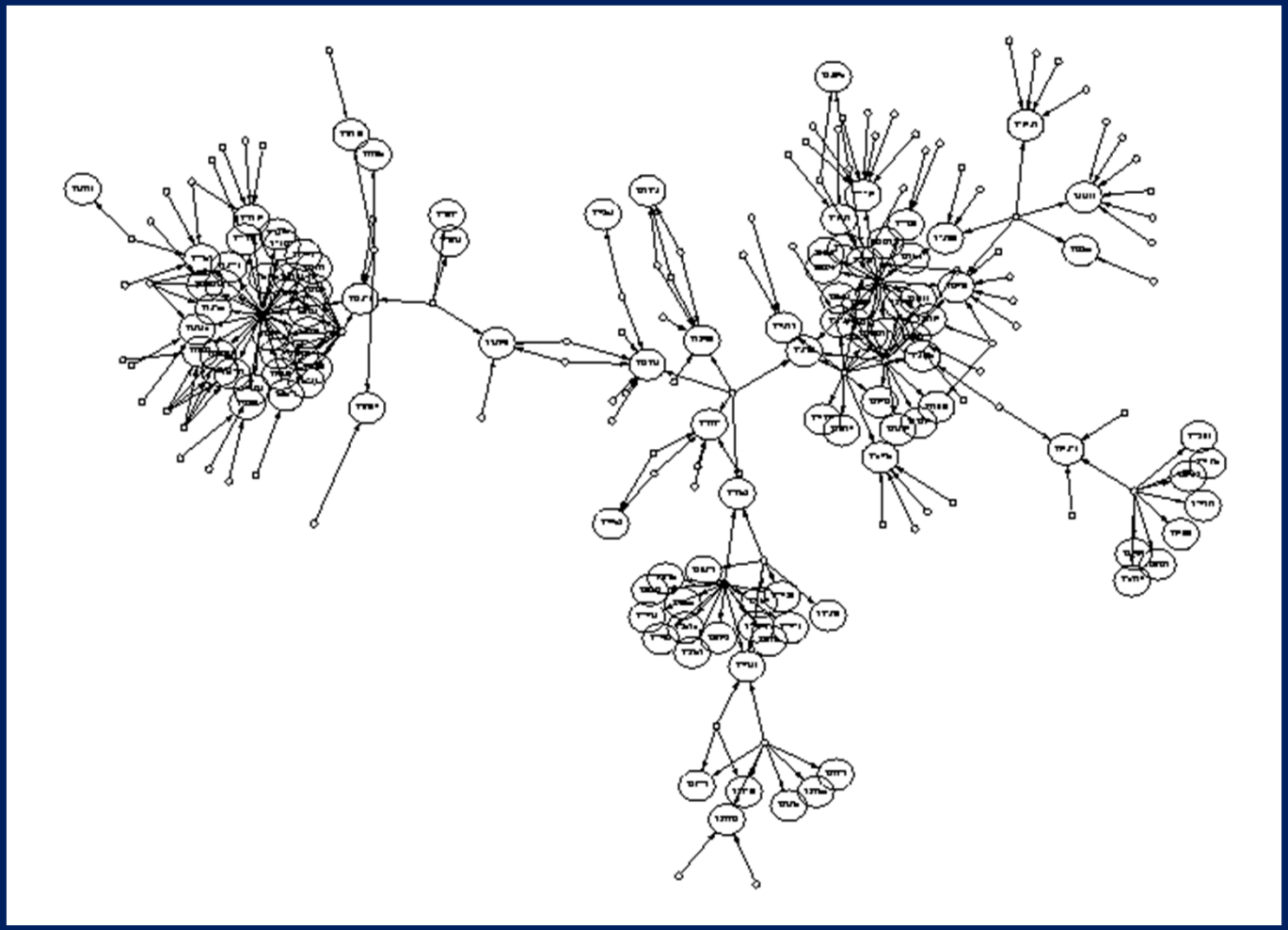
Attractive forces can be used to keep clusters together.

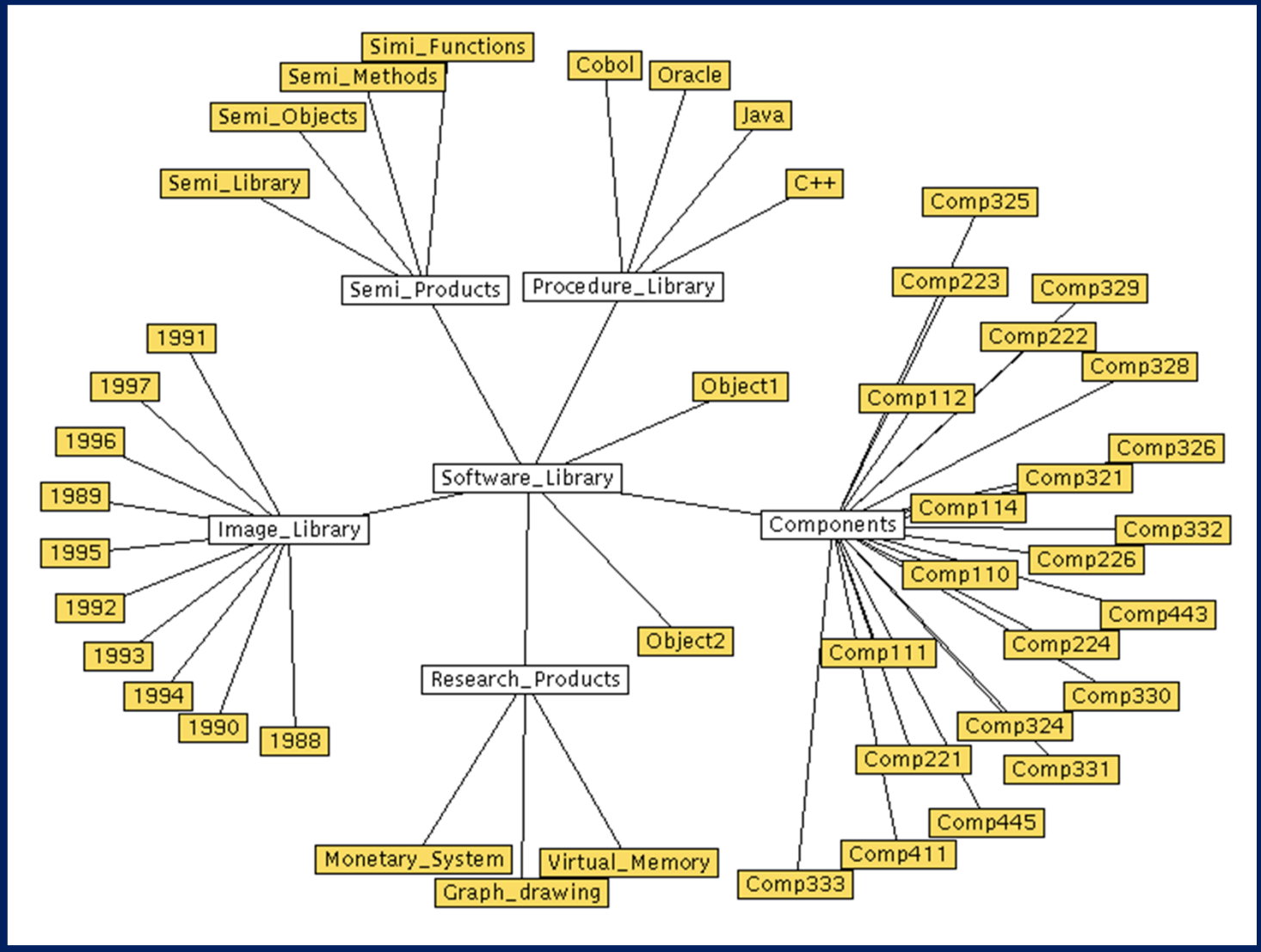


These constraints are very useful in customizing the general spring method to a specific domain.








Simi_Functions
Semi_Methods
Semi_Objects
Semi_Library
Cobol
Oracle
Java
C++
Semi_Products
Procedure_Library
1991
1997
1996
1989
1995
1992
1993
1994
1990
1988
Image_Library
Object1
Software_Library
Object2
Research_Products
Monetary_System
Virtual_Memory
Graph_drawing
Comp325
Comp223
Comp329
Comp222
Comp328
Comp112
Comp326
Comp321
Comp114
Components
Comp332
Comp226
Comp110
Comp443
Comp224
Comp111
Comp330
Comp324
Comp221
Comp331
Comp445
Comp411
Comp333

Example:

Metro Maps

- *Damian Merrick*
- *SeokHee Hong*
- *Hugo do Nascimento*

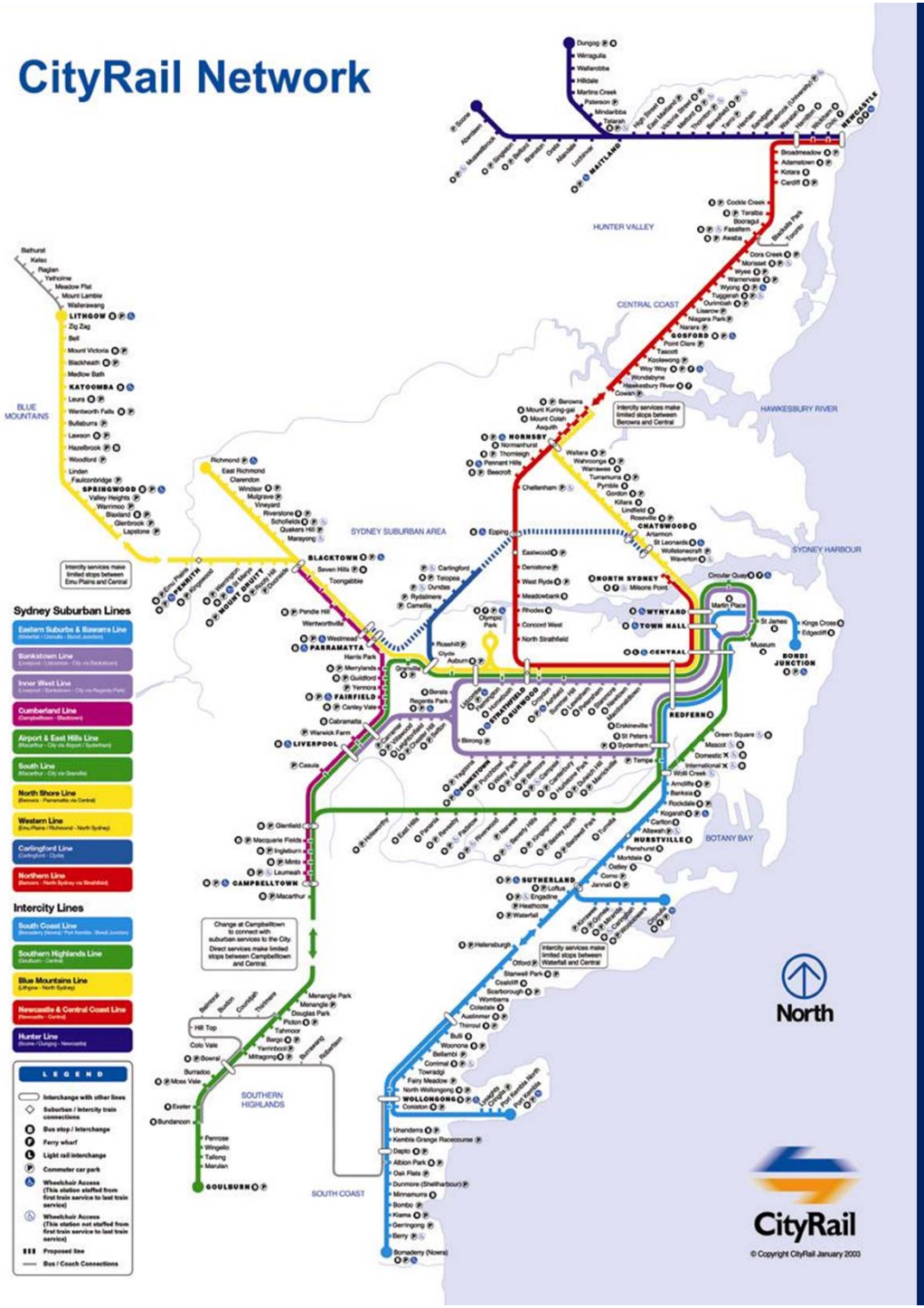
The Metro Map Problem

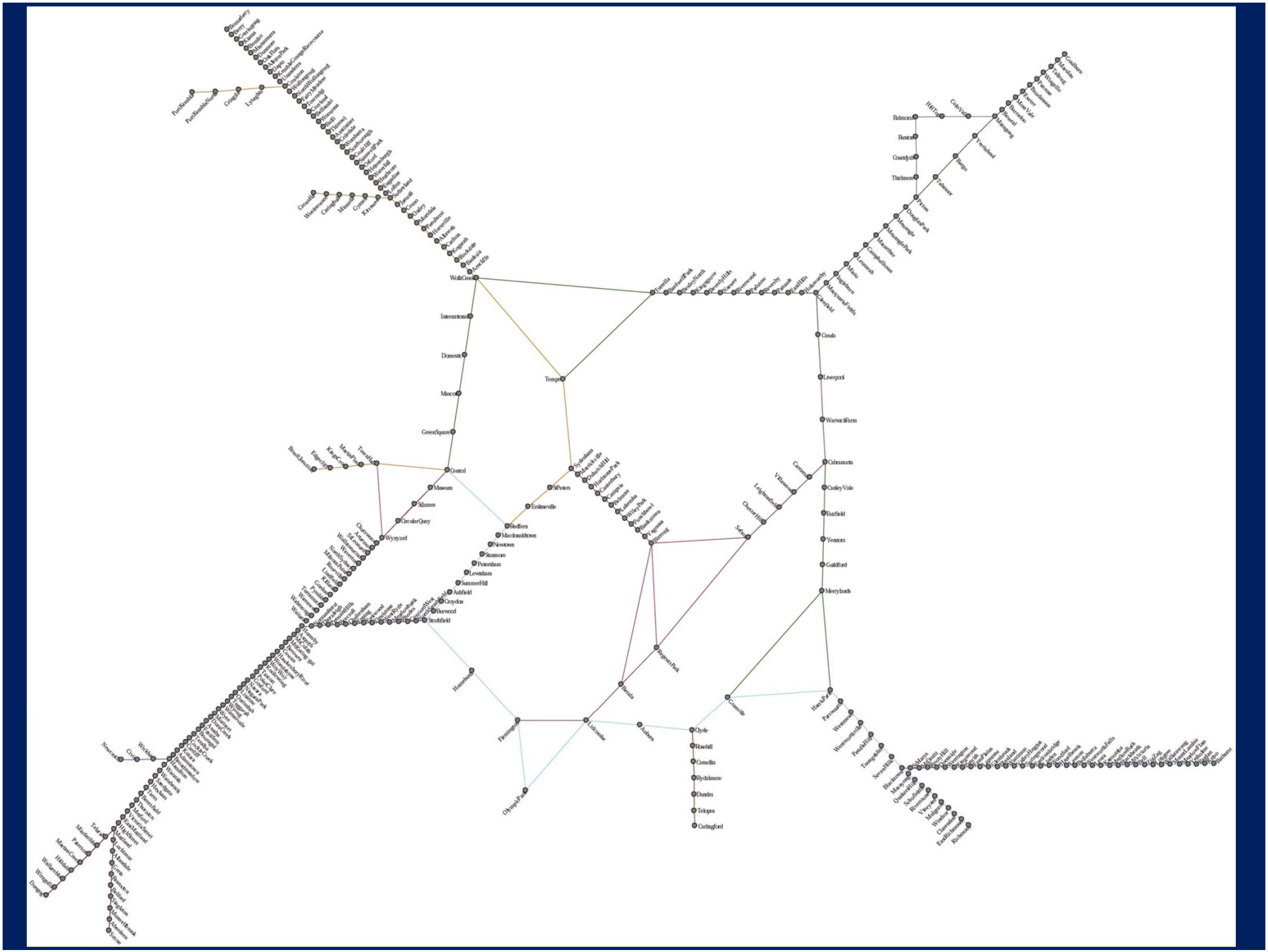
- Existing metro maps, produced by professional graphic artists, are ***excellent examples*** of network visualization

- *Challenge: Can we produce good metro maps* ***automatically?***

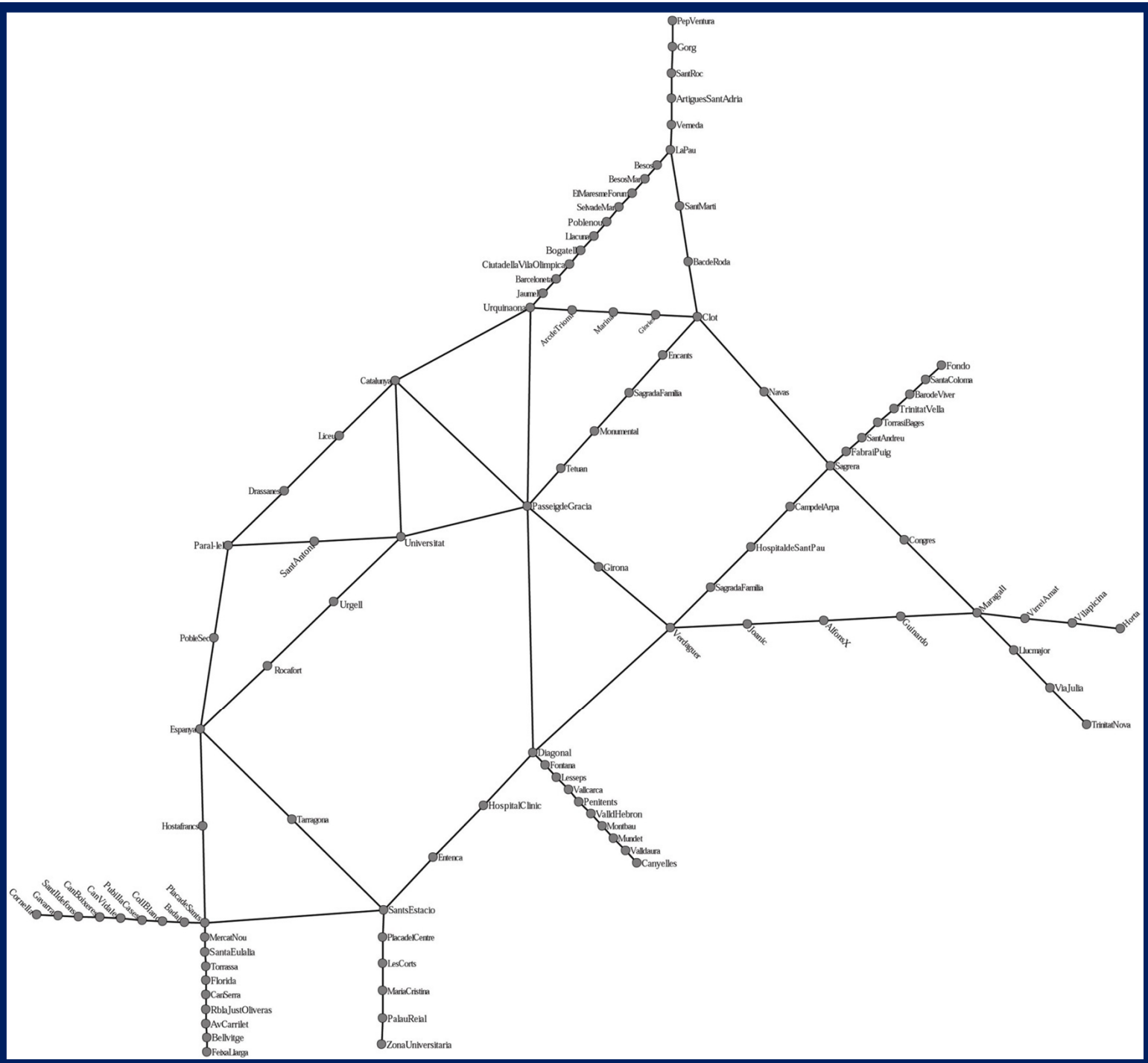
- ***Possible solution:***
  - Use a force-directed approach
  - Define forces that map good layout to low energy

CityRail Network





TMB
ATM
L3 Zona Universitària
L5 Cornellà
L1 Feixa Llarga
L3 Canyelles
L5 Horta
L11 Trinitat Nova
L4 Trinitat Nova
L11 Can Cuiàs
L1 Fondo
L4 La Pau
L2 Pep Ventura
L2 Paral·lel
Parc de Montjuïc
Esplugues de Llobregat
Cornellà de Llobregat
l'Hospitalet de Llobregat
Santa Coloma de Gramenet
Badalona
Sant Adrià de Besòs
Gavarra
Sant Ildefons
Can Boixeres
Can Vidalet
Pubilla Cases
Collblanc
Badal
Can Serra
Florida
Torrassa
Santa Eulàlia
Plaça de Sants
Rbla.Just Oliveras
Av. Carrilet
Bellvitge
Mercat Nou
Hostafrancs
Palau Reial
Maria Cristina
Les Corts
Plaça del Centre
Entença
Hospital Clínic
Diagonal
Verdaguer
Sants Estació
Tarragona
Espanya
Rocafort
Urgell
Universitat
Poble Sec
Sant Antoni
Catalunya
Liceu
Paral·lel
Drassanes
Passeig de Gràcia
Girona
Tetuan
Monumental
Arc de Triomf
Urquinaona
Jaume I
Barceloneta
Ciutadella Vila Olímpica
Bogatell
Llacuna
Poblenou
Selva de Mar
El Maresme Fòrum
Besòs Mar
Besòs
La Pau
Verneda
Artigues Sant Adrià
Sant Roc
Gorg
Bac de Roda
St. Martí
Clot
Glòries
Marina
Encants
Sagrada Família
Navas
Sagrera
Hospital de Sant Pau
Camp de l'Arpa
Congrés
Maragall
Guinardó
Joanic
Alfons X
Fontana
Lesseps
Vallcarca
Penitents
Vall d'Hebron
Montbau
Mundet
Valldaura
Vilapicina
Llucmajor
Virrei Amat
Via Júlia
Fabra i Puig
Torras i Bages
Sant Andreu
Trinitat Vella
Baró de Viver
Santa Coloma
Castell de Montjuïc
Mirador
Paral·lel


PepVentura
Gorg
SantRoc
ArtiguesSantAdria
Verneda
LaPau
Besos
BesosMar
ElMaresmeForum
SelvadeMar
Poblenou
Llacuna
Bogatell
CiutadellaVilaOlimpica
Barceloneta
JaumeI
SantMarti
BacdeRoda
Urquinaona
ArcdeTriomf
Marina
Glories
Clot
Encants
Catalunya
SagradaFamilia
Navas
Fondo
SantaColoma
BarodeViver
TrinitatVella
TorrasiBages
SantAndreu
FabraiPuig
Sagrera
Liceu
Monumental
Tetuan
Drassanes
PasseigdeGracia
CampdelArpa
Paral-lel
SantAntoni
Universitat
HospitaldeSantPau
Congres
Girona
SagradaFamilia
Maragall
VirrelAmat
Vilapicina
Horta
Urgell
Verdaguer
Joanic
AlfonsX
Guinardo
Llucmajor
PobleSec
Rocafort
ViaJulia
TrinitatNova
Espanya
Diagonal
Fontana
Lesseps
Vallcarca
Penitents
ValldHebron
Montbau
Mundet
Valldaura
Canyelles
HospitalClinic
Hostafrancs
Tarragona
Entenca
Cornella
Gavarra
SantIldefons
CanBoixeres
CanVidalJa
PubillaCases
CollBlanc
BadalPl
PlacadeSants
SantsEstacio
MercatNou
SantaEulalia
Torrassa
Florida
CanSerra
RblaJustOliveras
AvCarrilet
BellvitgeFeixaLlarga
PlacadelCentre
LesCorts
MariaCristina
PalauReial
ZonaUniversitaria

O'REILLY®

2003 OPEN SOURCE ROUTE MAP

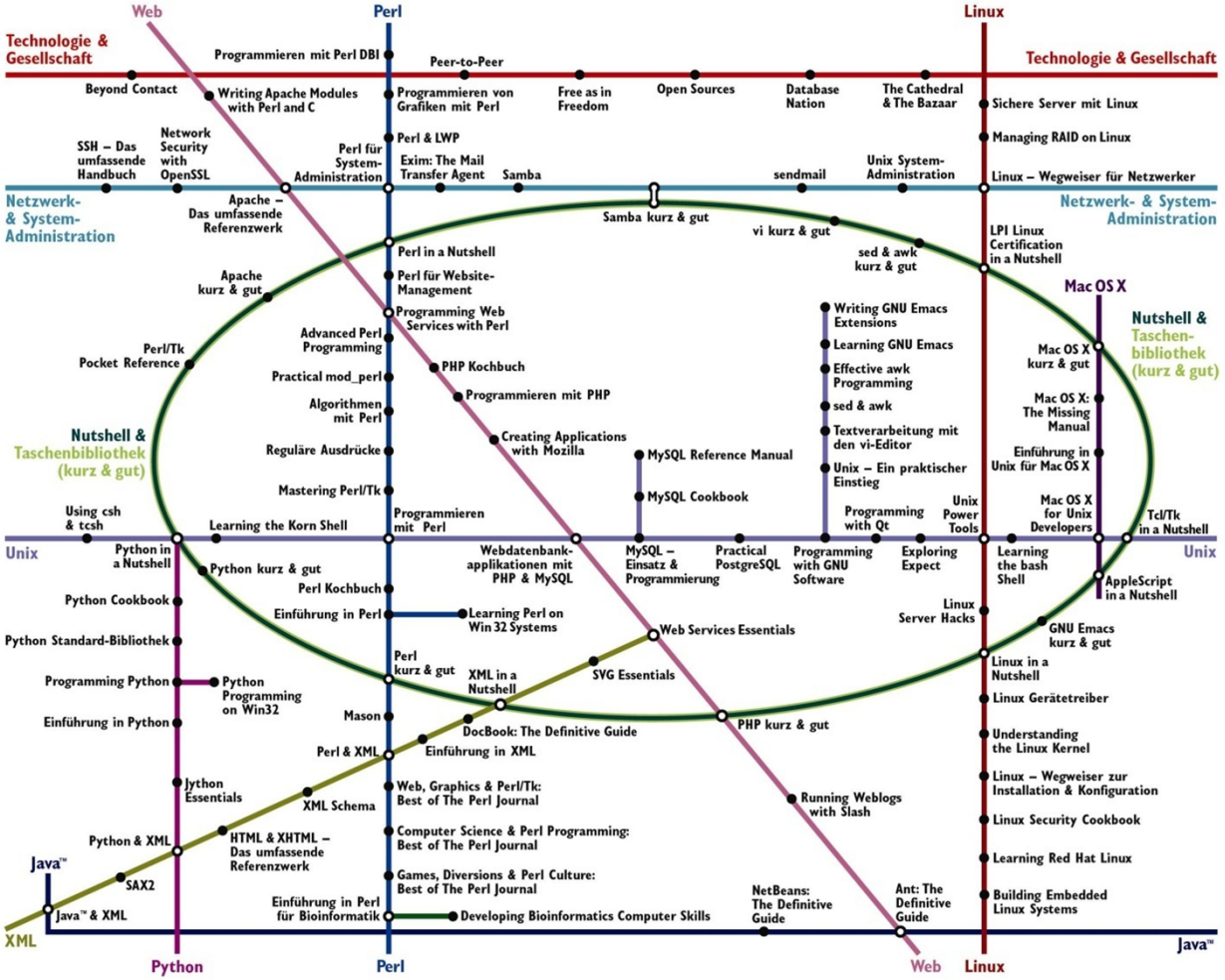
LINUX & KONSORTEN

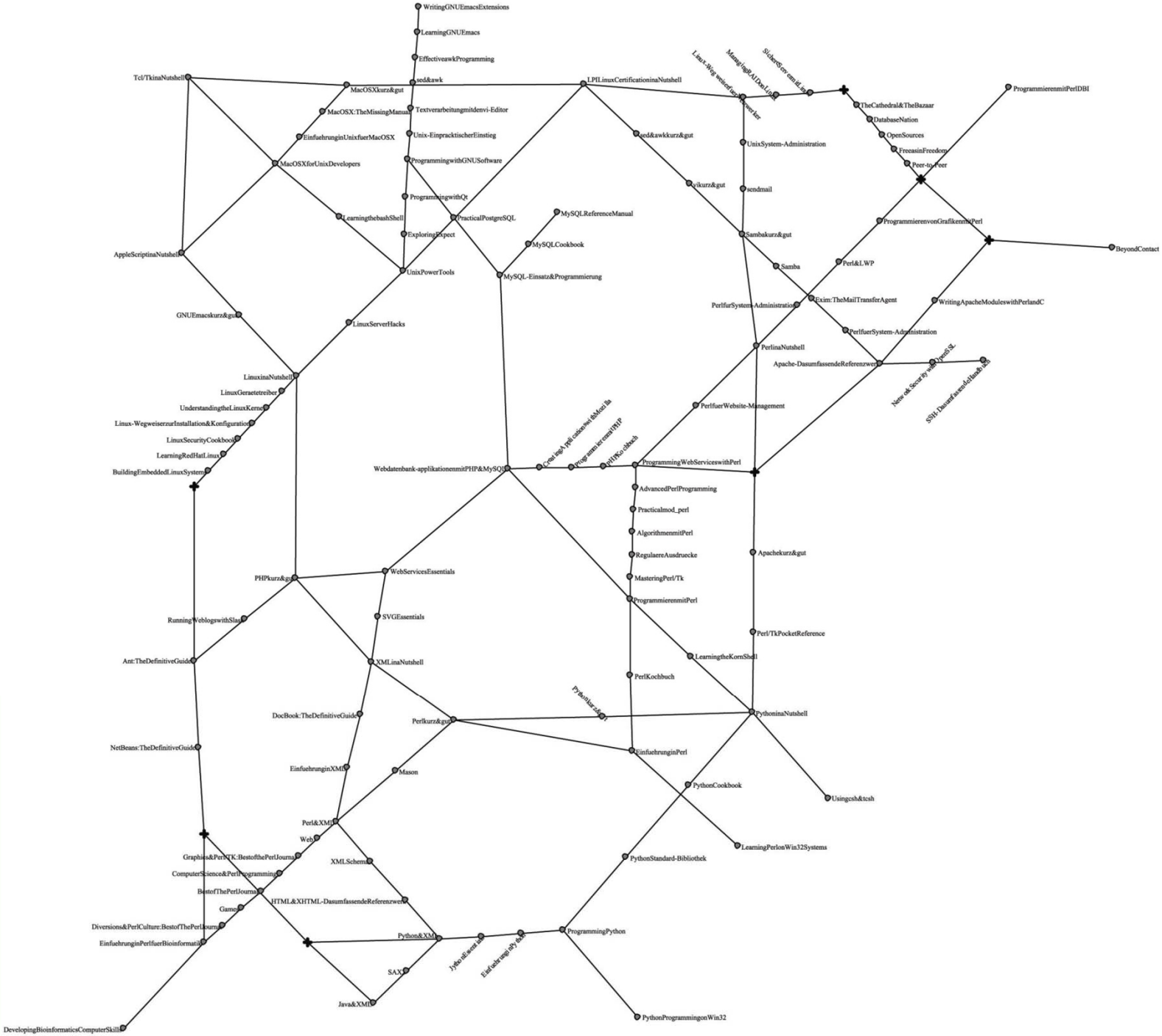
LEGENDE

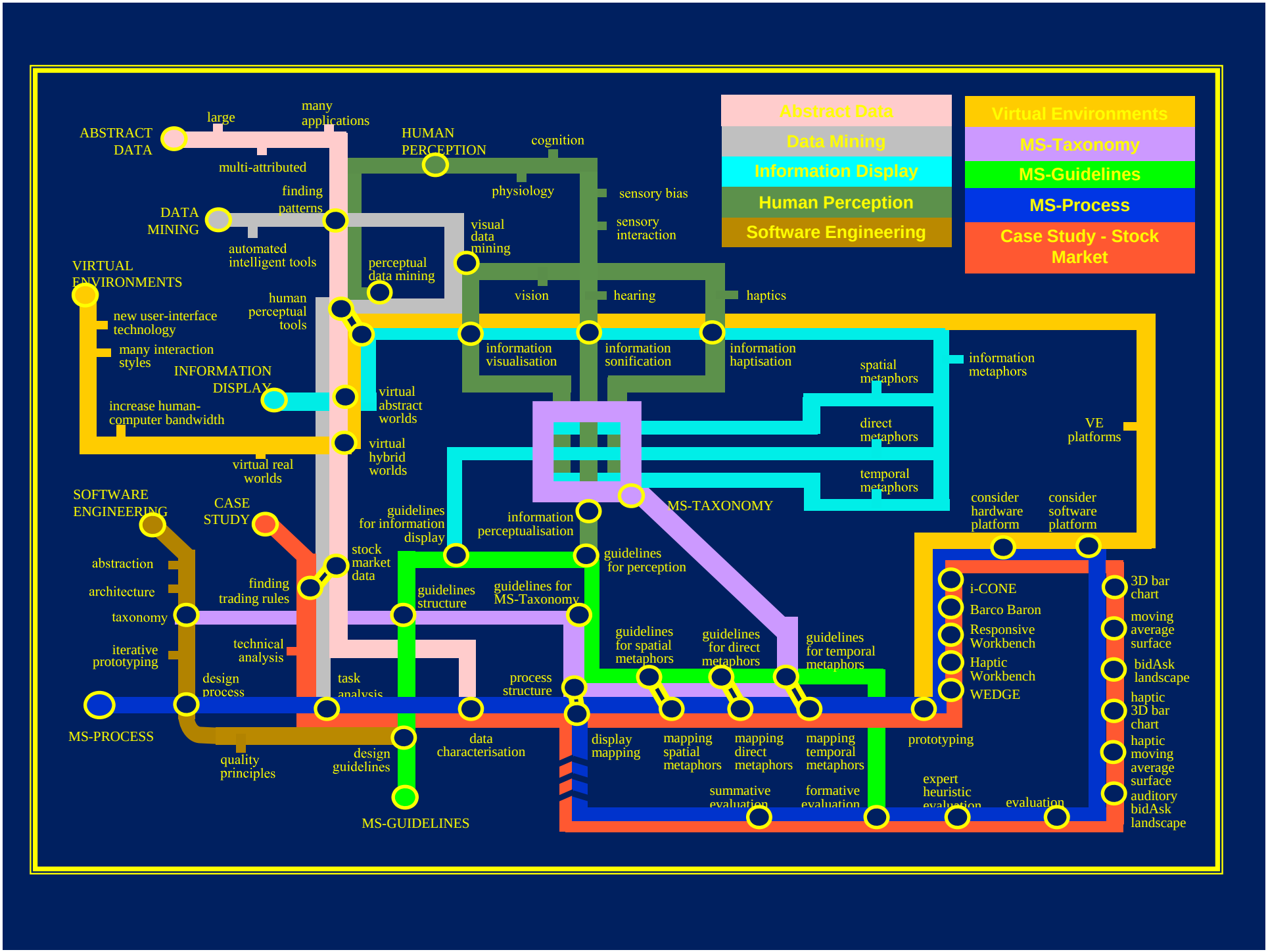
- Bioinformatik
- Java™
- Linux
- Mac OS X
- Netzwerk- & System-Administration
- Nutshell & Taschenbibliothek (kurz & gut)
- Perl
- Python
- Technologie & Gesellschaft
- Unix
- Web
- XML

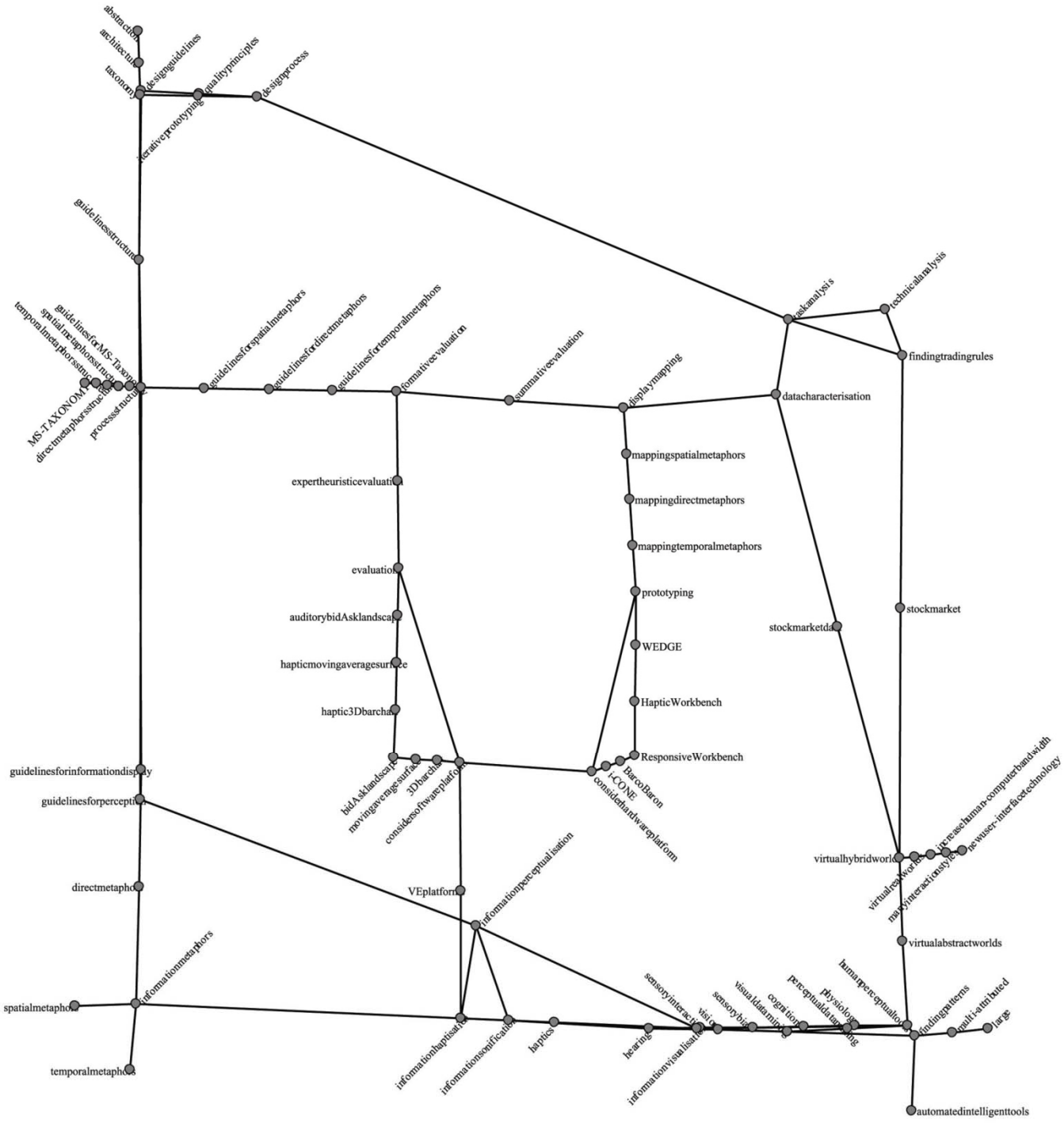
www.oreilly.de

Die Grübelei hat ein Ende!
Bücher von O'Reilly

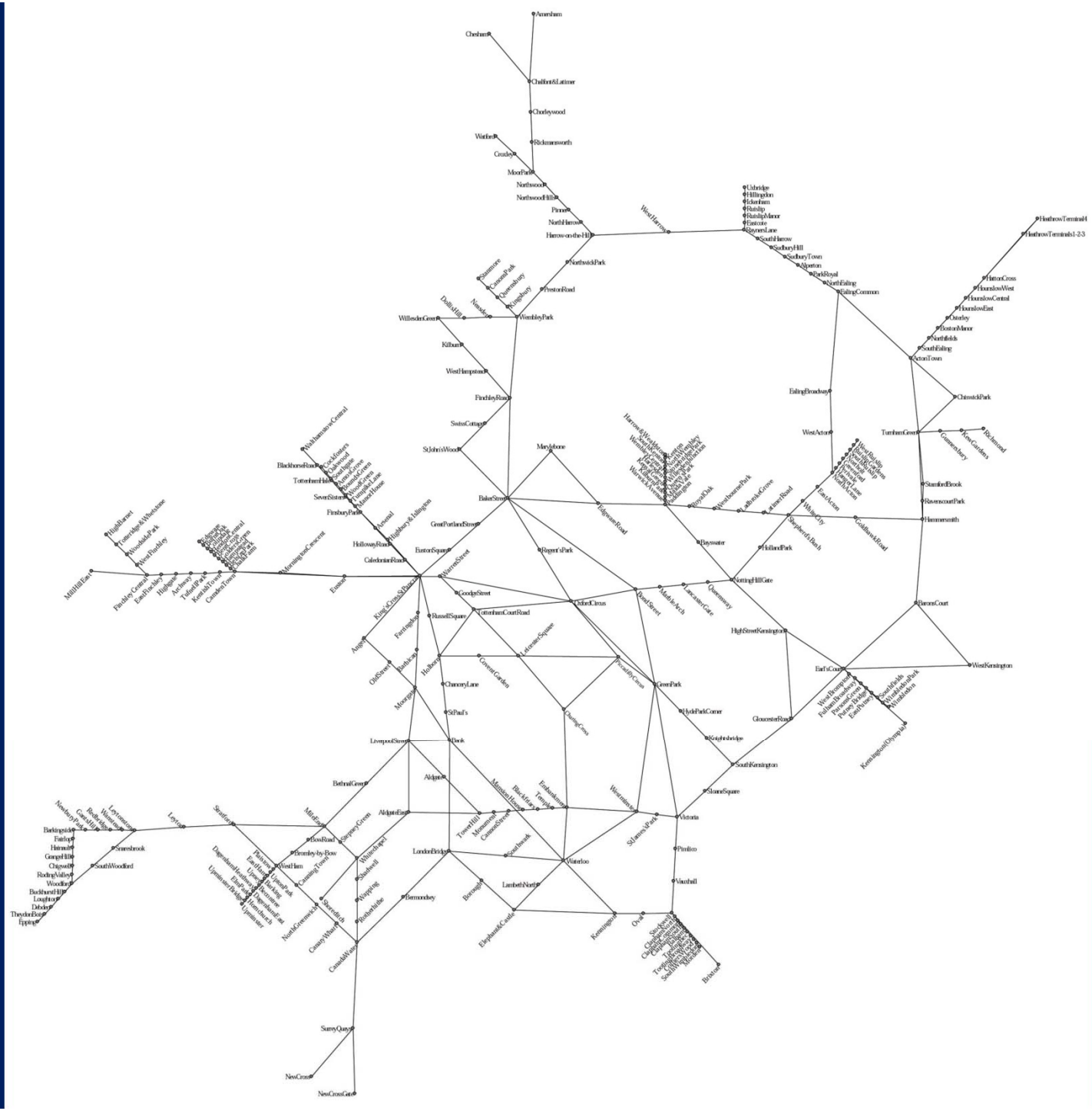


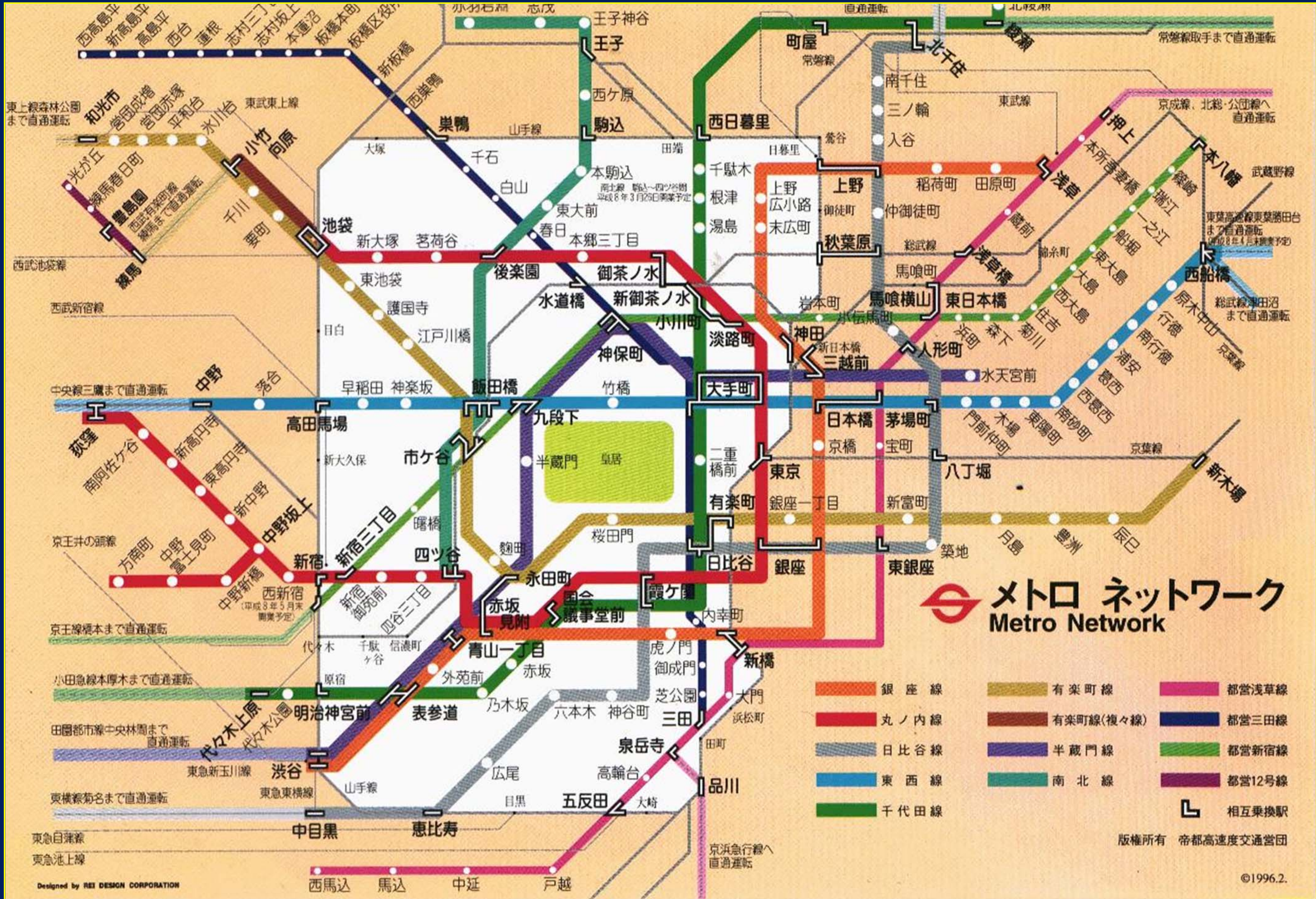

WritingGNUEmacsExtensions
LearningGNUEmacs
EffectiveawkProgramming
sed&awk
Tcl/TkinaNutshell
MacOSXkurz&gut
LPILinuxCertificationinaNutshell
MacOSX:TheMissingManual
Textverarbeitungmitdenvi-Editor
EinfuehrunginUnixfuerMacOSX
Unix-EinpracktischerEinstieg
sed&awkkurz&gut
UnixSystem-Administration
MacOSXforUnixDevelopers
ProgrammingwithGNUSoftware
vikurz&gut
sendmail
ProgrammingwithQt
LearningthebashShell
PracticalPostgreSQL
MySQLReferenceManual
ExploringExpect
MySQLCookbook
Sambakurz&gut
AppleScriptinaNutshell
UnixPowerTools
MySQL-Einsatz&Programmierung
Samba
GNUEmacskurz&gut
LinuxServerHacks
PerlfurSystem-Administration
Exim:TheMailTransferAgent
PerlfuerSystem-Administration
LinuxinaNutshell
PerlinaNutshell
LinuxGeraetetreiber
UnderstandingtheLinuxKernel
PerlfuerWebsite-Management
Linux-WegweiserzurInstallation&Konfiguration
LinuxSecurityCookbook
LearningRedHatLinux
BuildingEmbeddedLinuxSystems
Webdatenbank-applikationenmitPHP&MySQL
ProgrammingWebServiceswithPerl
AdvancedPerlProgramming
Practicalmod_perl
AlgorithmenmitPerl
RegulaereAusdruecke
Apachekurz&gut
PHPkurz&gut
WebServicesEssentials
MasteringPerl/Tk
ProgrammierenmitPerl
RunningWeblogswithSlash
SVGEssentials
Perl/TkPocketReference
LearningtheKornShell
Ant:TheDefinitiveGuide
XMLinaNutshell
PerlKochbuch
DocBook:TheDefinitiveGuide
PythoninaNutshell
Perlkurz&gut
NetBeans:TheDefinitiveGuide
EinfuehrungPerl
EinfuehrunginXML
Mason
PythonCookbook
Usingcsh&tcsh
Perl&XML
Web
LearningPerlonWin32Systems
Graphics&PerlTK:BestofthePerlJournal
XMLSchema
PythonStandard-Bibliothek
ComputerScience&PerlProgramming
BestofThePerlJournal
HTML&XHTML-DasumfassendeReferenzwerk
Games
Diversions&PerlCulture:BestofThePerlJournal
Python&XML
ProgrammingPython
EinfuehrungPerlfuerBioinformatik
SAX
Java&XML
PythonProgrammingonWin32
DevelopingBioinformaticsComputerSkills
ProgrammierenmitPerlDBI
TheCathedral&TheBazaar
DatabaseNation
OpenSources
FreeasinFreedom
Peer-to-Peer
ProgrammierenvonGrafikenmitPerl
BeyondContact
Perl&LWP
WritingApacheModuleswithPerlandC
Apache-DasumfassendeReferenzwerk


Abstract Data
Data Mining
Information Display
Human Perception
Software Engineering
Virtual Environments
MS-Taxonomy
MS-Guidelines
MS-Process
Case Study - Stock Market
ABSTRACT DATA
large
many applications
multi-attributed
DATA MINING
finding patterns
automated intelligent tools
HUMAN PERCEPTION
cognition
physiology
sensory bias
sensory interaction
visual data mining
perceptual data mining
VIRTUAL ENVIRONMENTS
new user-interface technology
many interaction styles
human perceptual tools
vision
hearing
haptics
information visualisation
information sonification
information haptisation
INFORMATION DISPLAY
increase human-computer bandwidth
virtual abstract worlds
virtual hybrid worlds
virtual real worlds
spatial metaphors
direct metaphors
temporal metaphors
information metaphors
VE platforms
SOFTWARE ENGINEERING
CASE STUDY
abstraction
architecture
taxonomy
iterative prototyping
stock market data
finding trading rules
technical analysis
guidelines for information display
information perceptualisation
MS-TAXONOMY
guidelines for perception
guidelines structure
guidelines for MS-Taxonomy
guidelines for spatial metaphors
guidelines for direct metaphors
guidelines for temporal metaphors
consider hardware platform
consider software platform
i-CONE
Barco Baron
Responsive Workbench
Haptic Workbench
WEDGE
3D bar chart
moving average surface
bidAsk landscape
haptic 3D bar chart
haptic moving average surface
auditory bidAsk landscape
design process
task analysis
process structure
MS-PROCESS
quality principles
design guidelines
data characterisation
display mapping
mapping spatial metaphors
mapping direct metaphors
mapping temporal metaphors
prototyping
summative evaluation
formative evaluation
expert heuristic evaluation
evaluation
MS-GUIDELINES

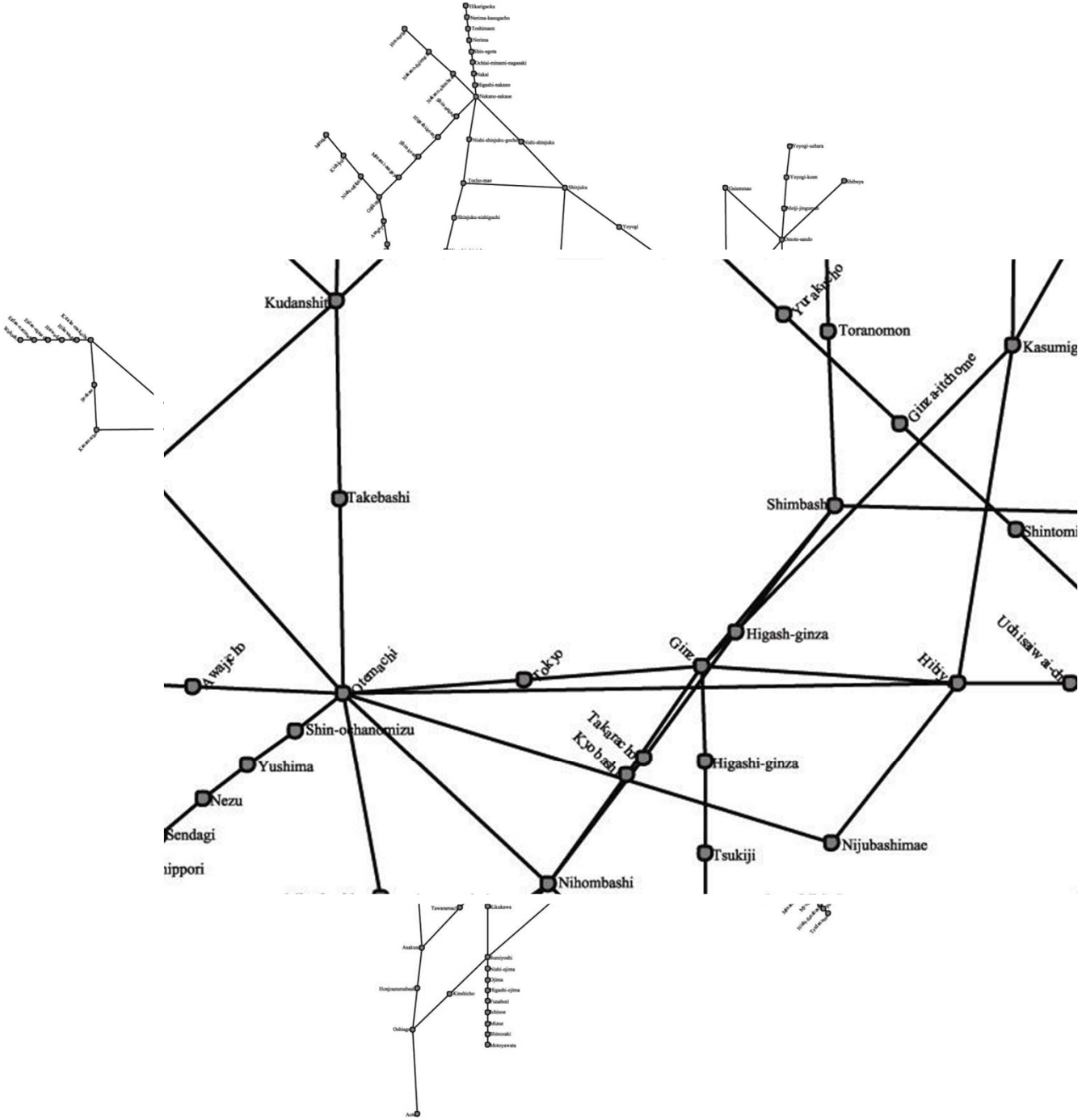

abstraction
architecture
taxonomy
designguidelines
qualityprinciples
designprocess
iterativeprototyping
guidelinesstructure
guidelinesforMS-Taxonomy
spatialmetaphorsstructure
temporalmetaphorsstructure
MS-TAXONOMY
directmetaphorsstructure
processstructure
guidelinesforspatialmetaphors
guidelinesfordirectmetaphors
guidelinesfortemporalmetaphors
formativeevaluation
summativeevaluation
displaymapping
taskanalysis
technicalanalysis
findingtradingrules
datacharacterisation
mappingspatialmetaphors
expertheuristicevaluation
mappingdirectmetaphors
mappingtemporalmetaphors
evaluation
prototyping
auditorybidAsklandscape
WEDGE
stockmarketdata
stockmarket
hapticmovingaveragesurface
HapticWorkbench
haptic3Dbarchart
ResponsiveWorkbench
BarcoBaron
i-CONE
considerhardwareplatform
bidAsklandscape
movingaveragesurface
3Dbarchart
considersoftwareplatform
guidelinesforinformationdisplay
guidelinesforperception
virtualhybridworlds
increasehuman-computerbandwidth
virtualreality
increaseuser-interfacetechnology
mapinteractionstyles
directmetaphors
VEplatform
informationperceptualisation
virtualabstractworlds
informationmetaphors
sensoryinteraction
humanperceptualissues
visualdatamining
perceptualdatamining
physiology
findingpatterns
multi-attributed
large
spatialmetaphors
sensorybias
cognition
vision
temporalmetaphors
informationhaptisation
informationsonification
haptics
hearing
informationvisualisation
automatedintelligenttools


Chesham
Amersham
Chalfont & Latimer
Chorleywood
Rickmansworth
Watford
Croxley
Moor Park
Northwood
Northwood Hills
Pinner
North Harrow
Harrow on-the-Hill
West Ruislip
Ruislip
Ruislip Manor
Eastcote
Hillingdon
Uxbridge
Ickenham
Ruislip Gardens
South Ruislip
Northolt
Greenford
Perivale
Hanger Lane
Ealing Broadway
Ealing Common
Acton Town
South Ealing
Northfields
Boston Manor
Osterley
Hounslow East
Hounslow Central
Hounslow West
Hatton Cross
Heathrow Terminals 1, 2, 3
Heathrow Terminal 4
Rayners Lane
West Harrow
South Harrow
Sudbury Hill
Sudbury Town
Alperton
Park Royal
North Ealing
North Acton
West Acton
East Acton
Acton Central
South Acton
Chiswick Park
Turnham Green
Stamford Brook
Ravenscourt Park
Harrow & Wealdstone
Kenton
Preston Road
Northwick Park
South Kenton
North Wembley
Wembley Central
Stonebridge Park
Harlesden
Willesden Junction
Wembley Park
Stanmore
Canons Park
Queensbury
Kingsbury
Neasden
Dollis Hill
Willesden Green
Kilburn
Brondesbury Park
Kensal Rise
Brondesbury
Kensal Green
Queen's Park
Kilburn Park
Maida Vale
Warwick Avenue
Shepherd's Bush
Shepherd's Bush
Kensington (Olympia)
Barons Court
Hammersmith
Goldhawk Road
Edgware
Burnt Oak
Colindale
Hendon Central
Brent Cross
Golders Green
Hampstead
Finchley Road & Frognal
Finchley Road
Swiss Cottage
St. John's Wood
West Hampstead
Edgware Road
Marylebone
Paddington
Paddington
Edgware Road
Bayswater
Queensway
Lancaster Gate
Marble Arch
Bond Street
Baker Street
Great Portland Street
Regent's Park
Oxford Circus
Warren Street
Euston
Euston Square
Euston
Goodge Street
Russell Square
Holborn
Tottenham Court Road
Chancery Lane
Notting Hill Gate
High Street Kensington
Holland Park
White City
Latimer Road
Ladbroke Grove
Westbourne Park
Royal Oak
Mill Hill East
High Barnet
Totteridge & Whetstone
Woodside Park
West Finchley
Finchley Central
East Finchley
Highgate
Archway
Tufnell Park
Belsize Park
Hampstead Heath
Gospel Oak
Kentish Town West
Kentish Town
Camden Road
Camden Town
Chalk Farm
Mornington Crescent
King's Cross St. Pancras
Angel
Essex Road
Old Street
Farringdon
Barbican
Moorgate
Liverpool Street
Bethnal Green
Mile End
Shoreditch
Aldgate East
Aldgate
St. Paul's
Bank
Cannon Street
Monument
Tower Hill
Tower Gateway
Mansion House
Blackfriars
Temple
Charing Cross
Embankment
Leicester Square
City Thameslink
Covent Garden
Piccadilly Circus
Green Park
Hyde Park Corner
Knightsbridge
Gloucester Road
Sloane Square
South Kensington
Victoria
St. James's Park
Westminster
Earl's Court
West Brompton
Fulham Broadway
Parsons Green
Putney Bridge
East Putney
Southfields
Wimbledon Park
Wimbledon
Pimlico
Vauxhall
Waterloo
Southwark
Lambeth North
Elephant & Castle
Borough
London Bridge
Kennington
Oval
Stockwell
Clapham North
Clapham Common
Clapham South
Balham
Tooting Bec
Tooting Broadway
Colliers Wood
South Wimbledon
Morden
Brixton
Cockfosters
Oakwood
Southgate
Arnos Grove
Bounds Green
Wood Green
Turnpike Lane
Manor House
Arsenal
Holloway Road
Caledonian Road
Caledonian Road & Barnsbury
Drayton Park
Finsbury Park
Highbury & Islington
Canonbury
Dalston Kingsland
Hackney Central
Homerton
Hackney Wick
Tottenham Hale
Seven Sisters
Blackhorse Road
Walthamstow Central
Epping
Theydon Bois
Debden
Loughton
Buckhurst Hill
Roding Valley
Chigwell
Woodford
South Woodford
Snaresbrook
Redbridge
Wanstead
Gants Hill
Grange Hill
Hainault
Fairlop
Barkingside
Newbury Park
Leytonstone
Leyton
Stratford
Pudding Mill Lane
Bow Road
Bow Church
Bromley-by-Bow
Devons Road
Stepney Green
Whitechapel
All Saints
Westferry
Poplar
East India
Blackwall
Limehouse
Shadwell
Wapping
Rotherhithe
Canada Water
Bermondsey
Surrey Quays
West India Quay
Canary Wharf
Heron Quays
South Quay
Crossharbour & London Arena
Mudchute
Island Gardens
North Greenwich
New Cross Gate
New Cross
Cutty Sark
Greenwich
Deptford Bridge
Elverson Road
Lewisham
Upminster
Upminster Bridge
Hornchurch
Elm Park
Dagenham East
Dagenham Heathway
Becontree
Upney
Barking
East Ham
Upton Park
Plaistow
West Ham
Canning Town
Royal Victoria
Custom House
Prince Regent
Royal Albert
Beckton Park
Cyprus
Gallions Reach
Beckton
Silvertown & London City Airport
North Woolwich
River Thames
Key to lines
Bakerloo
Central
peak hours only
Circle
District †
East London
peak hours and Sunday mornings
Hammersmith & City †
Jubilee
under construction
Metropolitan
peak hours only
Northern
Piccadilly †
Victoria
Waterloo & City †
Docklands Light Railway
under construction
National Railways
restricted service
Interchange stations
Connections with National Railways
Connections with National Railways within walking distance
Airport interchange
Closed Sundays
Closed Saturdays and Sundays
Served by Piccadilly line trains early morning and late evening
† For opening times see poster journey planners.
Certain stations are closed on public holidays.
UNDERGROUND
London Travel Information 0171-222 1234 24 hours
Minicom 0171-918 3015
© London Regional Transport


Amersham
Chesham
Chalfont&Latimer
Chorleywood
Rickmansworth
Watford
Croxley
MoorPark
Northwood
NorthwoodHills
Pinner
NorthHarrow
Harrow-on-the-Hill
WestHarrow
Uxbridge
Hillingdon
Ickenham
Ruislip
RuislipManor
Eastcote
RaynersLane
SouthHarrow
SudburyHill
SudburyTown
Alperton
ParkRoyal
NorthEaling
EalingCommon
HeathrowTerminal4
HeathrowTerminals1-2-3
HattonCross
HounslowWest
HounslowCentral
HounslowEast
Osterley
BostonManor
Northfields
SouthEaling
EltonTown
ChiswickPark
EalingBroadway
WestActor
TurnhamGreen
GunnersburyRichmond
KewGardens
StamfordBrook
RavenscourtPark
Hammersmith
GoldhawkRoad
ShepherdsBush
WhiteCity
EastActon
WestBromptonFulhamBroadwayParsonsGreenPutneyBridgeEarlsCourtSouthfieldsWimbledonParkWimbledonKensingtonOlympia
EarlsCourt
WestKensington
BaronsCourt
HighStreetKensington
GloucesterRoad
SouthKensington
SloaneSquare
Victoria
StJames'sPark
Pimlico
Vauxhall
Waterloo
SouthBank
LondonBridge
Borough
LambethNorth
ElephantCastle
Kennington
Oval
Stockwell
ClaphamNorth
ClaphamCommon
ClaphamSouth
TootingBec
TootingBroadway
SouthWimbledon
Morden
Bank
LiverpoolStreet
Monument
MansionHouse
BlackFriars
Temple
Embankment
Westminster
Aldgate
AldgateEast
TowerHill
CannonStreet
StPaul's
ChanceryLane
CoventGarden
LeicesterSquare
PiccadillyCircus
GreenPark
HydeParkCorner
KnightsbridgeChanceryLane
ChariingCross
TottenhamCourtRoad
RussellSquare
FarringdonAngel
OldStreet
Barbican
Moorgate
KingsCrossSt.Pancras
GoodgeStreet
OxfordCircus
RegentStreet
ManchesterSquare
LancasterGate
Queensway
NottingHillGate
Bayswater
HollandPark
RegentsPark
EustonSquare
WarrenStreet
CaledonianRoad
HollowayRoad
Arsenal
GreatPortlandStreet
EdgwareRoad
BakerStreet
Marylebone
FinsburyPark
SevenSisters
TottenhamHale
BlackhorseRoad
WalthamstowCentral
WoodGreen
Turnpike Lane
Manor House
Finchley Road
SwissCottage
StJohn'sWood
WestHampstead
Kilburn
WillesdenGreen
DollisHill
Neasden
WembleyPark
Stanmore
CanonsPark
QueensBury
KingsBury
PrestonRoad
NorthwickPark
MorningtonCrescent
Euston
MillHillEast
FinchleyCentral
EastFinchley
Highgate
Archway
TufnellPark
KentishTown
CamdenTown
HighBarnet
TowerHill&Park
WoodsidePark
WestFinchley
Edgware
BurnhamGreen
Colindale
HendonCentral
ColindaleHendon
BrentCross
CricklewoodHarlesden
WillesdenJunction
KensalGreen
WestBrompton
SouthKensington
Knightsbridge
HydeParkCorner
GreenPark
PiccadillyCircus
LeicesterSquare
CoventGarden
ChanceryLane
StPaul's
Bank
LiverpoolStreet
Monument
MansionHouse
BlackFriars
Temple
Embankment
Westminster
Aldgate
AldgateEast
TowerHill
CannonStreet
LondonBridge
Borough
LambethNorth
ElephantCastle
Kennington
Oval
Stockwell
ClaphamNorth
ClaphamCommon
ClaphamSouth
TootingBec
TootingBroadway
SouthWimbledon
Morden
Victoria
StJames'sPark
Pimlico
Vauxhall
Waterloo
SouthBank
Bermondsey
Canada Water
SurreyQuays
NewCross
NewCrossGate
Stratford
LeytonstoneLeyton
Wanstead
RedbridgeGantsHill
NewbeyPark
Barkingside
Fairlop
Hainault
GrangeHill
Chigwell
RodingValley
Woodford
BuckhurstHill
Loughton
Debden
TheydonBois
Epping
SouthWoodford
Snaresbrook
MileEnd
BowRoad
Bromley-by-Bow
WestHam
CanningTown
PlaistowUpton Park
EastHam
BecktonPark
BeckonSE
CustomHouse
GallionsReach
DagenhamHeathway
DagenhamEast
UpminsterBridge
Upminster
NorthGreenwich
CuttyWharf
Shadwell
Wapping
RotherhitheCanada Water
Shortlands
Stepney Green
WhitechapelShadwell
Bermondsey
Borough
LambethNorth
ElephantCastle
Kennington
Oval
Stockwell
ClaphamNorth
ClaphamCommon
ClaphamSouth
TootingBec
TootingBroadway
SouthWimbledon
Morden
Victoria
StJames'sPark
Pimlico
Vauxhall
Waterloo
SouthBank
LondonBridge
Bermondsey
Borough
LambethNorth
ElephantCastle
Kennington
Oval
Stockwell
ClaphamNorth
ClaphamCommon
ClaphamSouth
TootingBec
TootingBroadway
SouthWimbledon
Morden

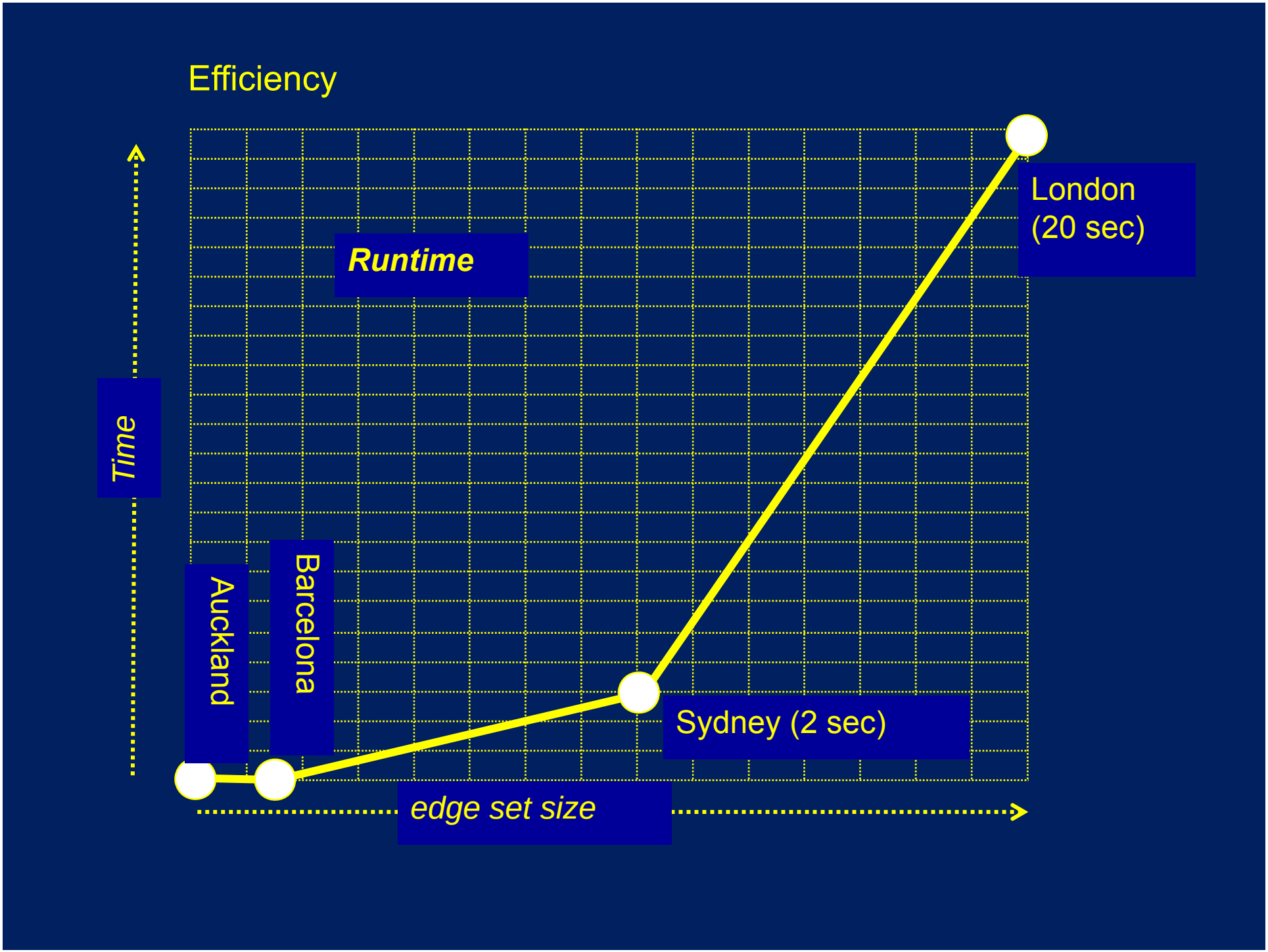


メトロ ネットワーク
Metro Network
銀座線
丸ノ内線
日比谷線
東西線
千代田線
有楽町線
有楽町線(複々線)
半蔵門線
南北線
都営浅草線
都営三田線
都営新宿線
都営12号線
相互乗換駅
版権所有 帝都高速度交通営団
©1996.2.
Designed by REI DESIGN CORPORATION
常磐線取手まで直通運転
京成線、北総・公団線へ直通運転
東上線森林公園まで直通運転
中央線三鷹まで直通運転
京王線橋本まで直通運転
小田急線本厚木まで直通運転
田園都市線中央林間まで直通運転
東横線菊名まで直通運転
京浜急行線へ直通運転
総武線津田沼まで直通運転
西武池袋線
西武新宿線
東武東上線
東武線
京葉線
京王井の頭線
東急新玉川線
東急東横線
東急目蒲線
東急池上線
山手線
王子
駒込
西日暮里
北千住
町屋
綾瀬
和光市
小竹向原
池袋
巣鴨
上野
浅草
秋葉原
押上
本八幡
西船橋
中野
高田馬場
飯田橋
九段下
大手町
日本橋
茅場町
八丁堀
新木場
荻窪
中野坂上
新宿
新宿三丁目
市ケ谷
四ツ谷
永田町
赤坂見附
国会議事堂前
霞ケ関
日比谷
有楽町
銀座
東銀座
東京
新橋
三田
品川
五反田
泉岳寺
渋谷
表参道
明治神宮前
代々木上原
中目黒
恵比寿
青山一丁目
後楽園
御茶ノ水
新御茶ノ水
水道橋
神保町
淡路町
小川町
神田
三越前
人形町
馬喰横山
東日本橋
浅草橋
水天宮前
千駄木
根津
湯島
上野広小路
末広町
稲荷町
田原町
仲御徒町
南千住
三ノ輪
入谷
千石
白山
春日
本郷三丁目
東大前
本駒込
西ケ原
王子神谷
新大塚
茗荷谷
東池袋
護国寺
江戸川橋
早稲田
神楽坂
竹橋
落合
半蔵門
皇居
二重橋前
桜田門
麹町
曙橋
京橋
宝町
銀座一丁目
新富町
築地
月島
辰巳
門前仲町
木場
東陽町
南砂町
西葛西
葛西
浦安
南行徳
行徳
原木中山
虎ノ門
御成門
芝公園
神谷町
六本木
乃木坂
外苑前
広尾
白金台
高輪台
大門
西馬込
馬込
中延
戸越
西新宿
新中野
東高円寺
新高円寺
南阿佐ケ谷
方南町
中野富士見町
中野新橋
光が丘
練馬春日町
豊島園
練馬
平和台
氷川台
千川
要町
西高島平
新高島平
高島平
西台
蓮根
本蓮沼
板橋本町
板橋区役所前
新板橋
西巣鴨
森下
菊川
住吉
西大島
大島
東大島
船堀
一之江
瑞江
篠崎
内幸町


Kudanshita
Takebashi
Toranomon
Shimbashi
Shintomi
Higash-ginza
Otemachi
Tokyo
Hibiya
Shin-ochanomizu
Yushima
Nezu
Sendagi
Nihombashi
Higashi-ginza
Tsukiji
Nijubashimae

Informal conclusion:

- The force directed method is a *little bit effective*, but *not very effective*.


Efficiency
Time
Runtime
London
(20 sec)
Sydney (2 sec)
Barcelona
Auckland
edge set size

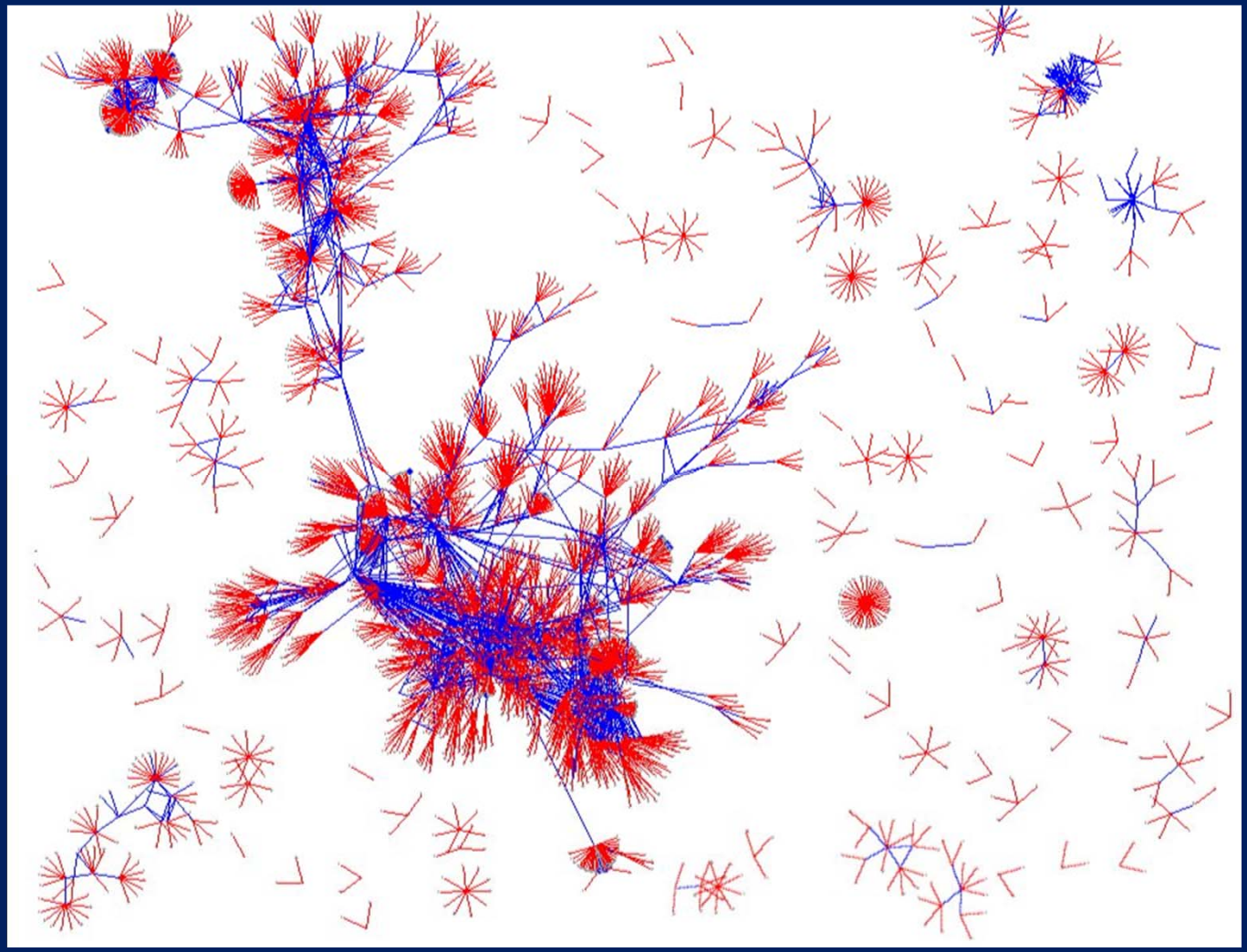
Informal conclusion:

- The force directed method for metro maps is *not computationally efficient*.

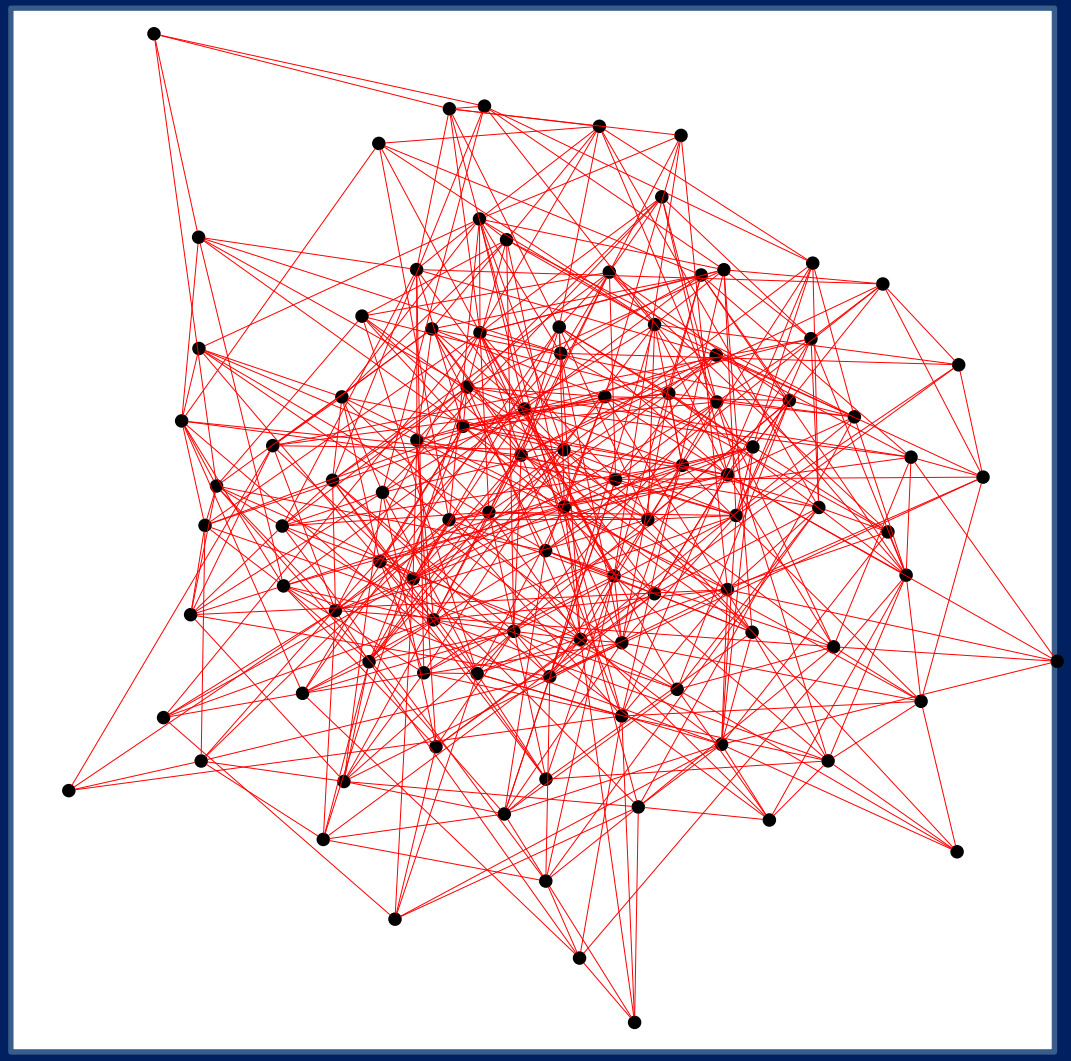
The performance of force directed methods on metro maps is typical force directed methods:

- Elegant and easy to implement
- Effective but not very effective
- Not very efficient

For some data sets, force directed methods give reasonably good drawings.



For some data sets, force directed methods give bad drawings.



How good are current force directed methods?

Elegance:

- Many simple methods, easy to implement
- Numerical software often available

Effectiveness:

- Very flexible ☑
- Straight-line edges ☑
- Planar graphs are *not* drawn planar ☒
- Very poor untangling for large graphs ☒

Efficiency:

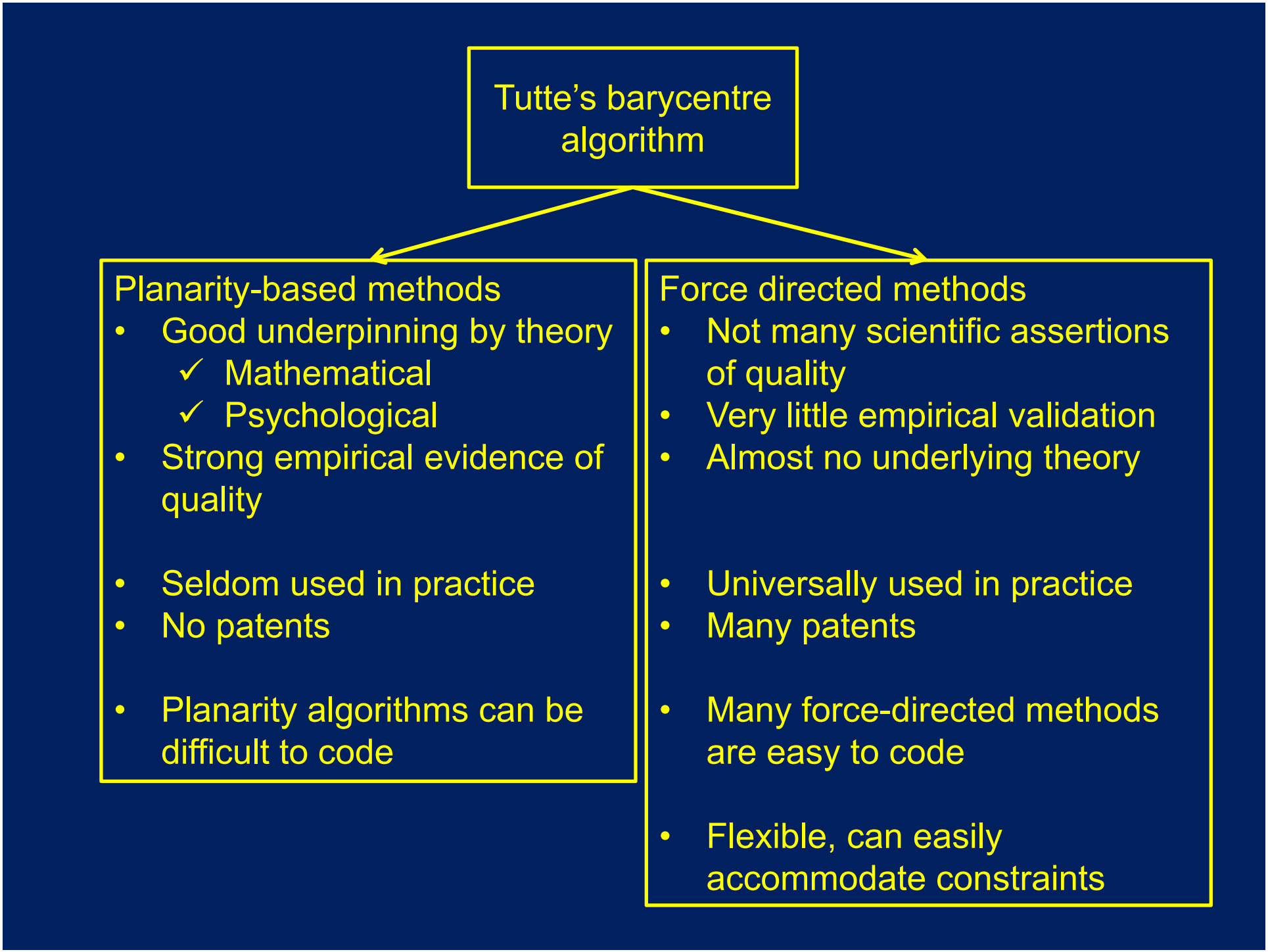
- OK for small graphs
- Sometimes OK for larger graphs (using sophisticated numerical methods)

The commercial state-of-the-art for force directed methods:

- Many commercial force-directed tools graph drawing methods are available
  - IBM (ILOG)
  - TomSawyer Software
  - yWorks
- Much free software available
  - GEOMI
  - GraphVis
- Many patents on variations of force-directed methods
- Force-directed methods account for 90% of commercial and free graph drawing software for undirected graphs

The scientific state-of-the-art for force directed methods

- Very few scientific human experiments have been done on the results of force directed methods.
- Very few theorems have been proven about force directed methods
  - Tutte's theorem
  - A theorem on symmetry of the output
  - Some theorems on multidimensional scaling can be applied.
- Some empirical comparisons (in terms of hard metrics) have been done.
- Many informal (unscientific?) investigations have been done
  - Appeal to developer intuition
  - Case studies in context


Tutte's barycentre algorithm
Planarity-based methods
• Good underpinning by theory
✓ Mathematical
✓ Psychological
• Strong empirical evidence of quality
• Seldom used in practice
• No patents
• Planarity algorithms can be difficult to code
Force directed methods
• Not many scientific assertions of quality
• Very little empirical validation
• Almost no underlying theory
• Universally used in practice
• Many patents
• Many force-directed methods are easy to code
• Flexible, can easily accommodate constraints

2. How to draw a planar graph?
   a) Before Tutte: 1920s – 1950s
   b) Tutte: 1960s
   c) After Tutte: 1970s – 1990s
   d) Recent work

Recent work

- *slightly non-planar graphs*

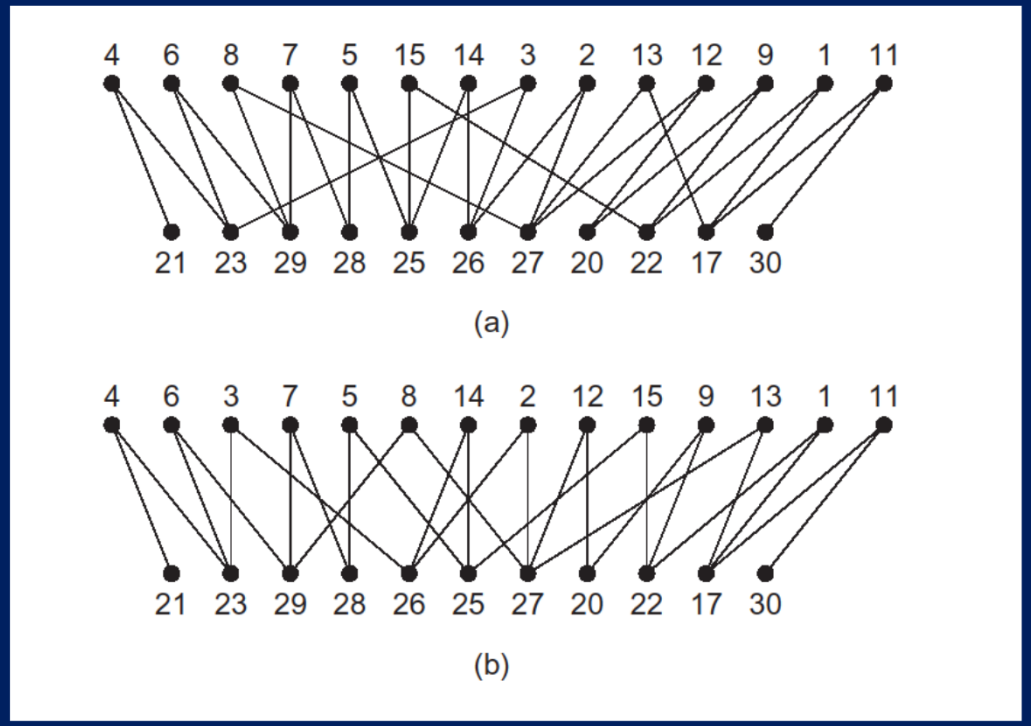
Motivation

Mutzel experiment 1997 - 98

- Informal “experiment”, performed at a talk
- Audience members were the “subjects”

Results

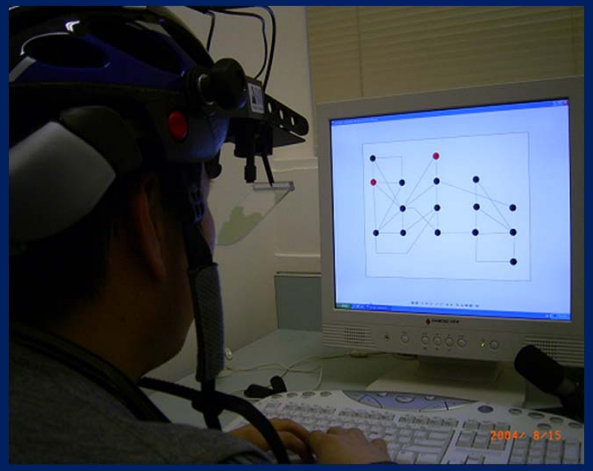
- People prefer (a) over (b)
- People erroneously see (a) as having fewer crossings than (b)



Motivation

Tony Huang 2003+

- Series of formal human experiments using eye-tracking.



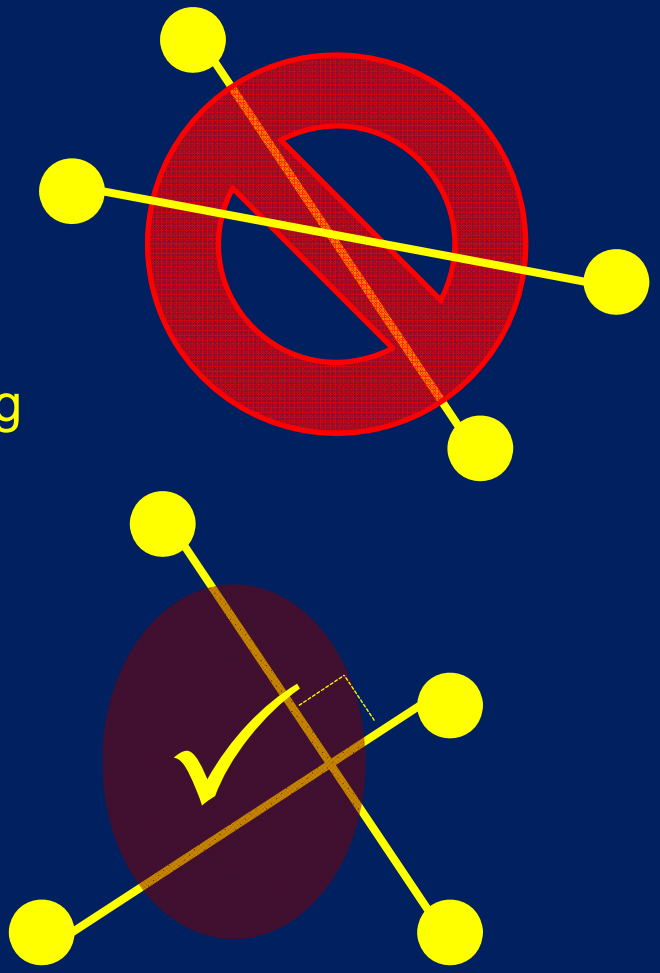
Huang's thesis

If the crossing angles are large, then non-planar drawings are OK.

Slightly non-planar graphs

Right-Angle Crossing (RAC) graphs:

- Straight-line edges
- If two edges cross, then the crossing makes a right angle



Questions for slightly non-planar graphs:

- How dense can a RAC graph be?

Theorem (Liotta, Didimo, Eades, 2009)

Suppose that ***G*** is a RAC graph with ***n*** vertices and ***m*** edges.
Then $m \leq 4n-10$.

Questions for slightly non-planar graphs:

- How dense can a RAC graph be?
- How can you compute a drawing of a RAC graph?

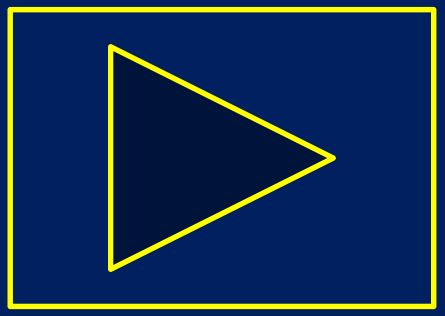
Theorem (Liotta, Eades, unpublished)*

The following problem is NP-hard:

*Input: A graph **G***

*Question: Is there a straight-line RAC drawing of **G**?*

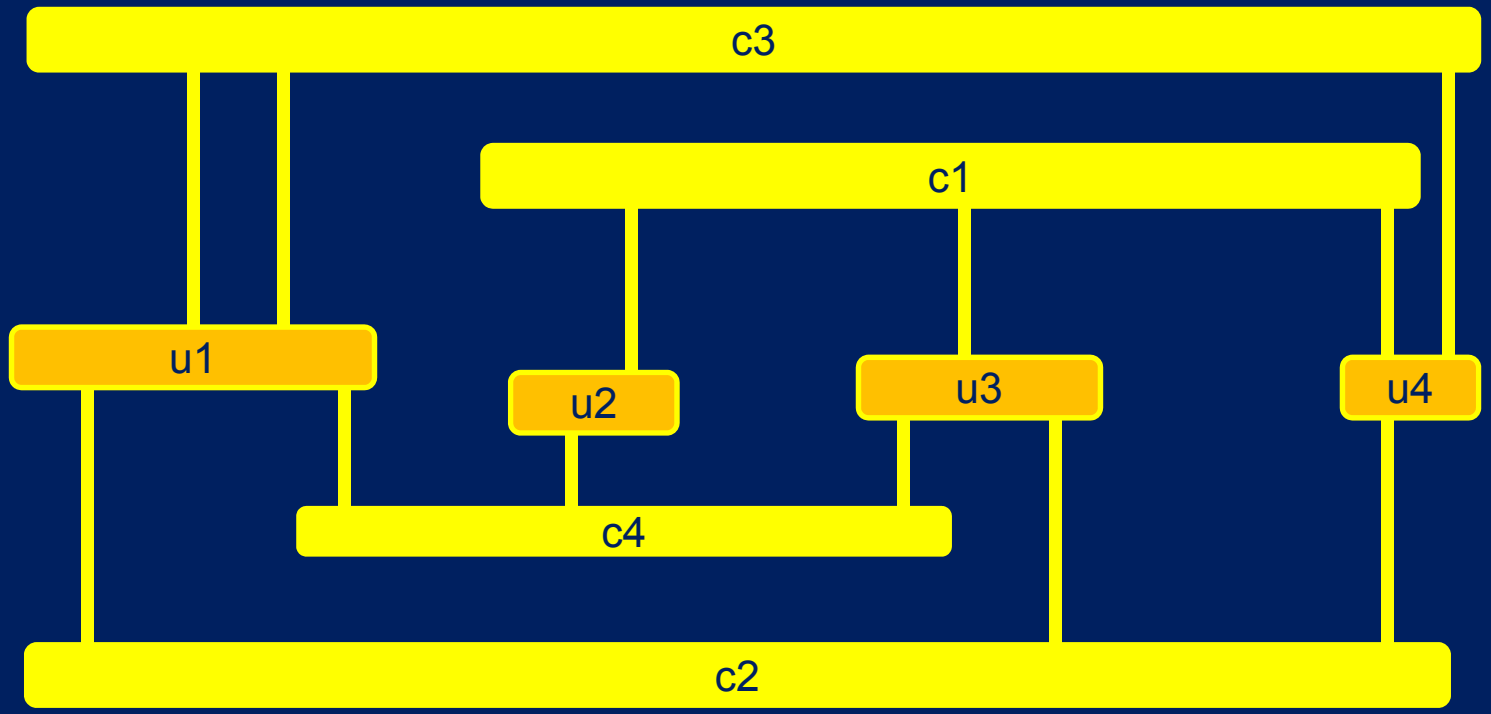
*Independently proved and published by Argyriou, Bekos and Symvonis



Proof

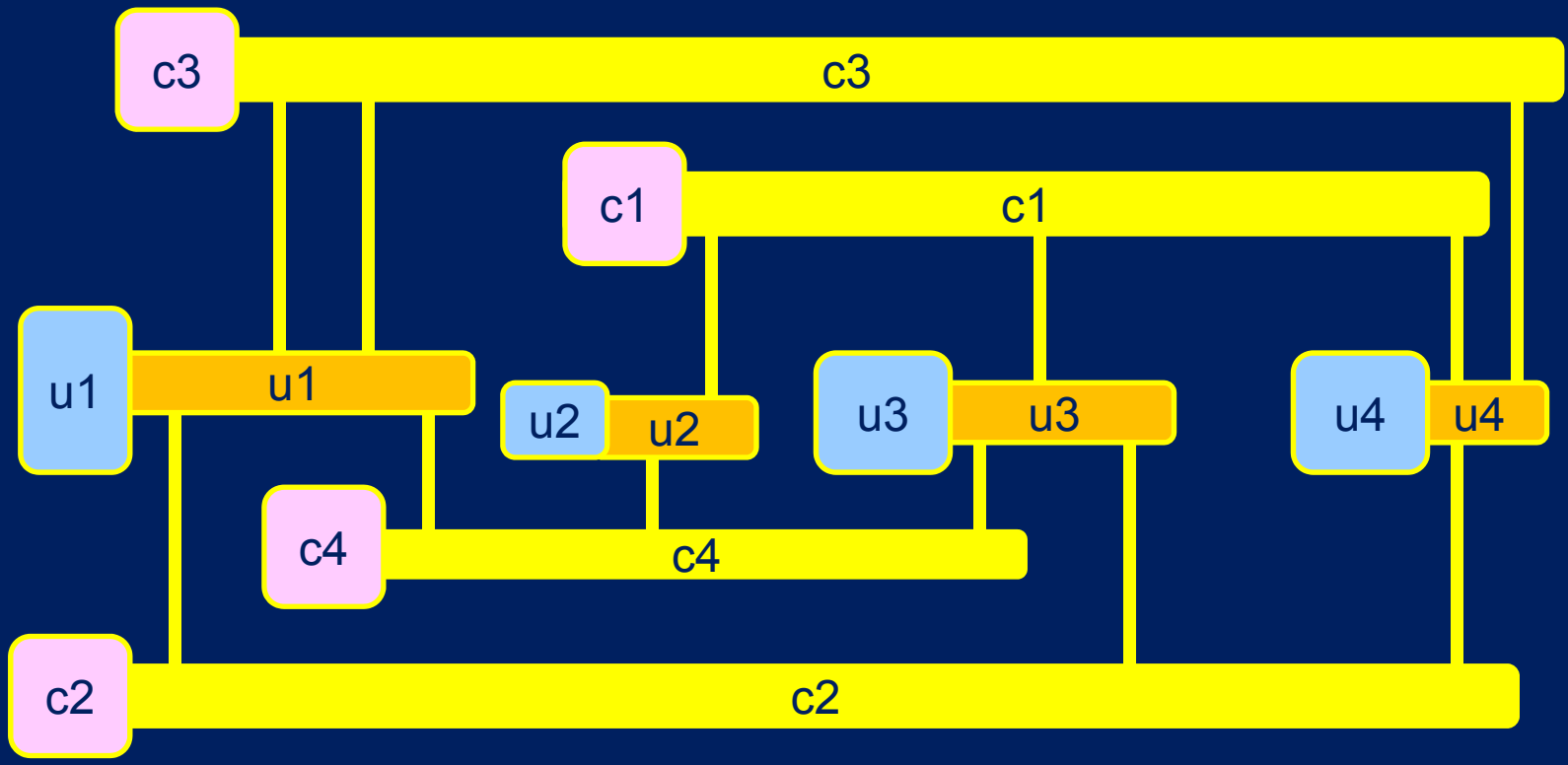
- Reduction from planar-3-sat
- Draw the instance ***H*** of planar-3-sat as a template
- Fill in details of the template ***H*** to form a graph ***G*** that has a RAC drawing if and only if ***H*** is satisfiable.

- Fairly generic proof strategy for NP-hardness for layout problems.

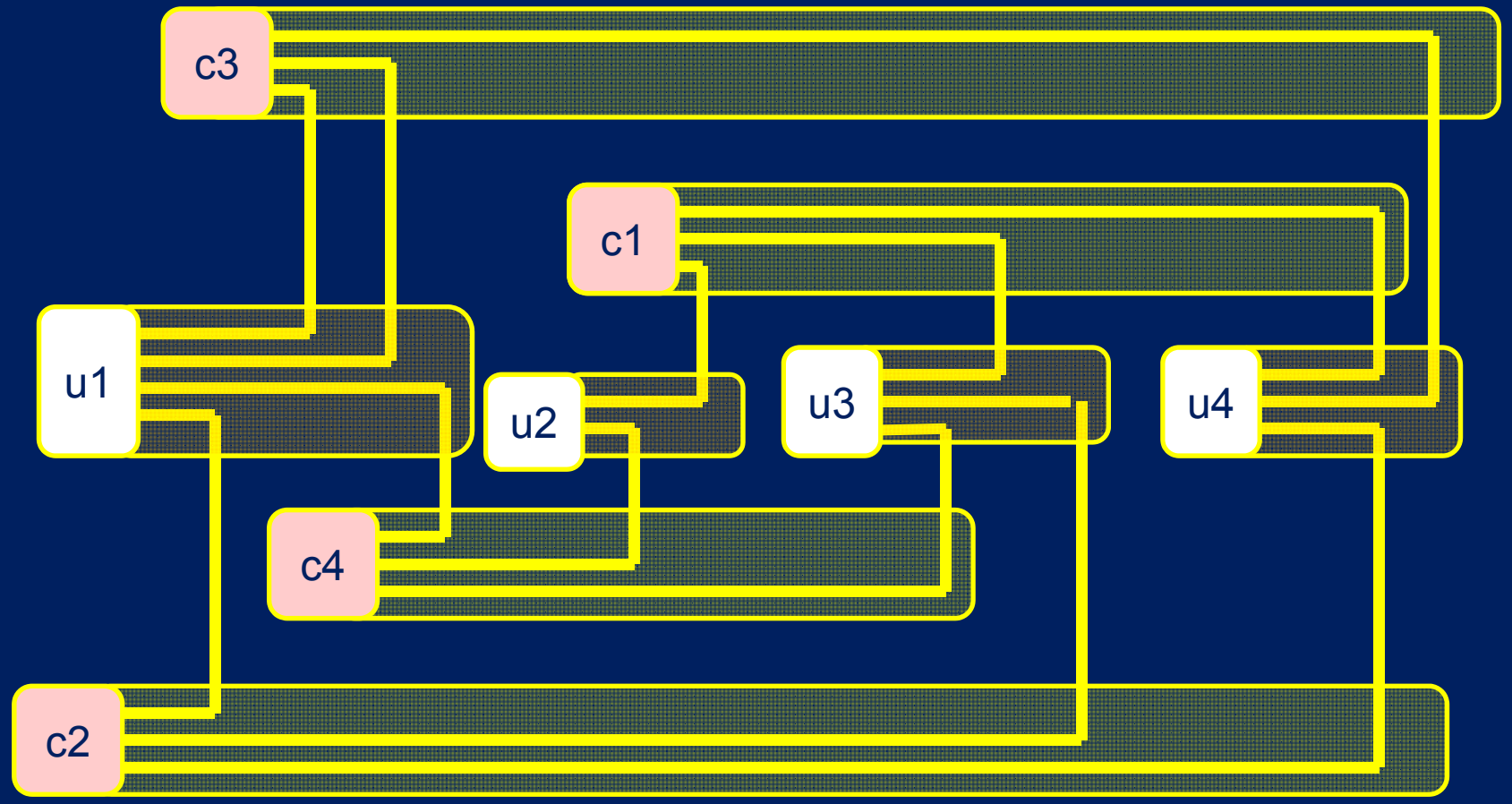


Instance ***H*** of planar 3-sat graph

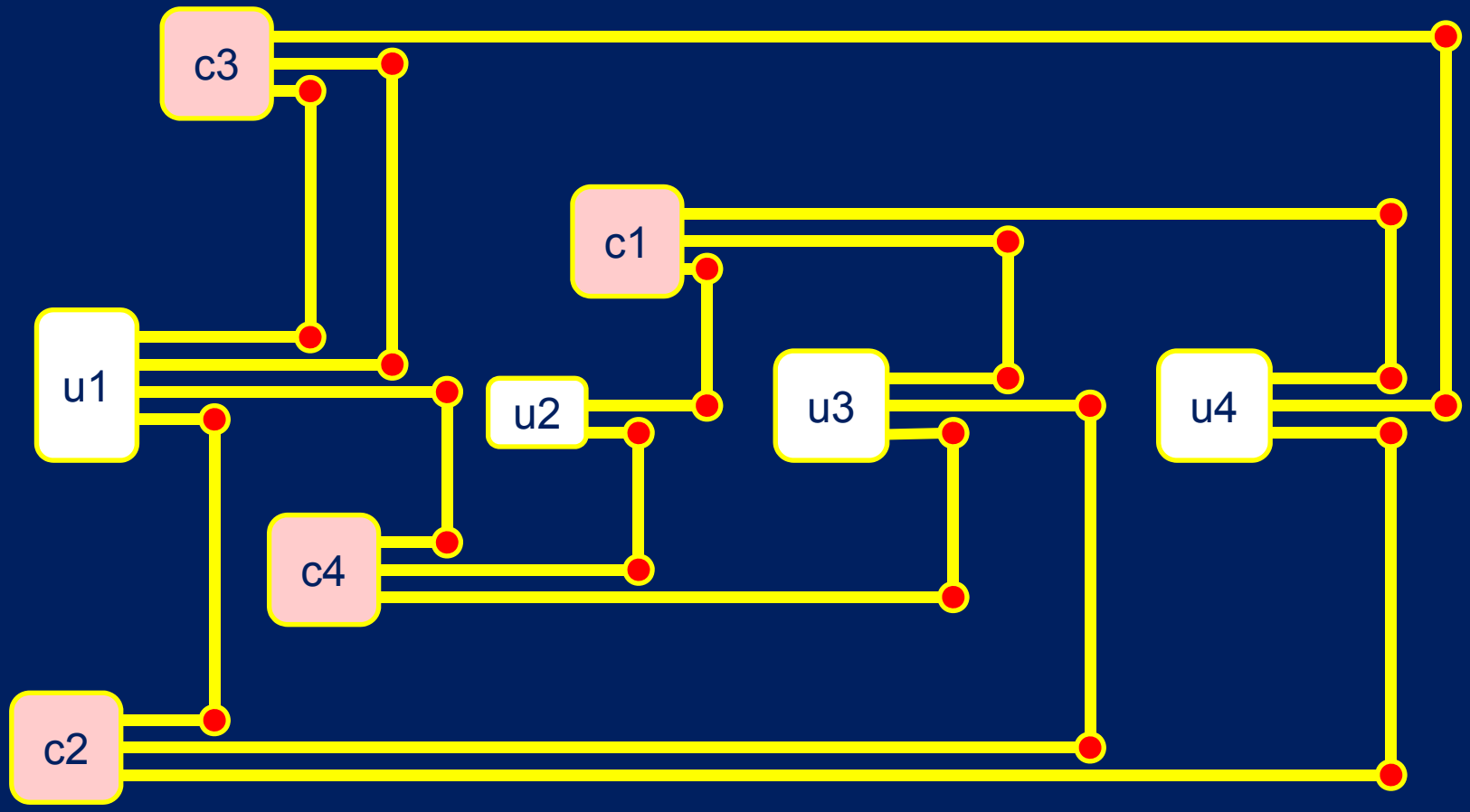
1. Draw ***H*** as a visibility drawing



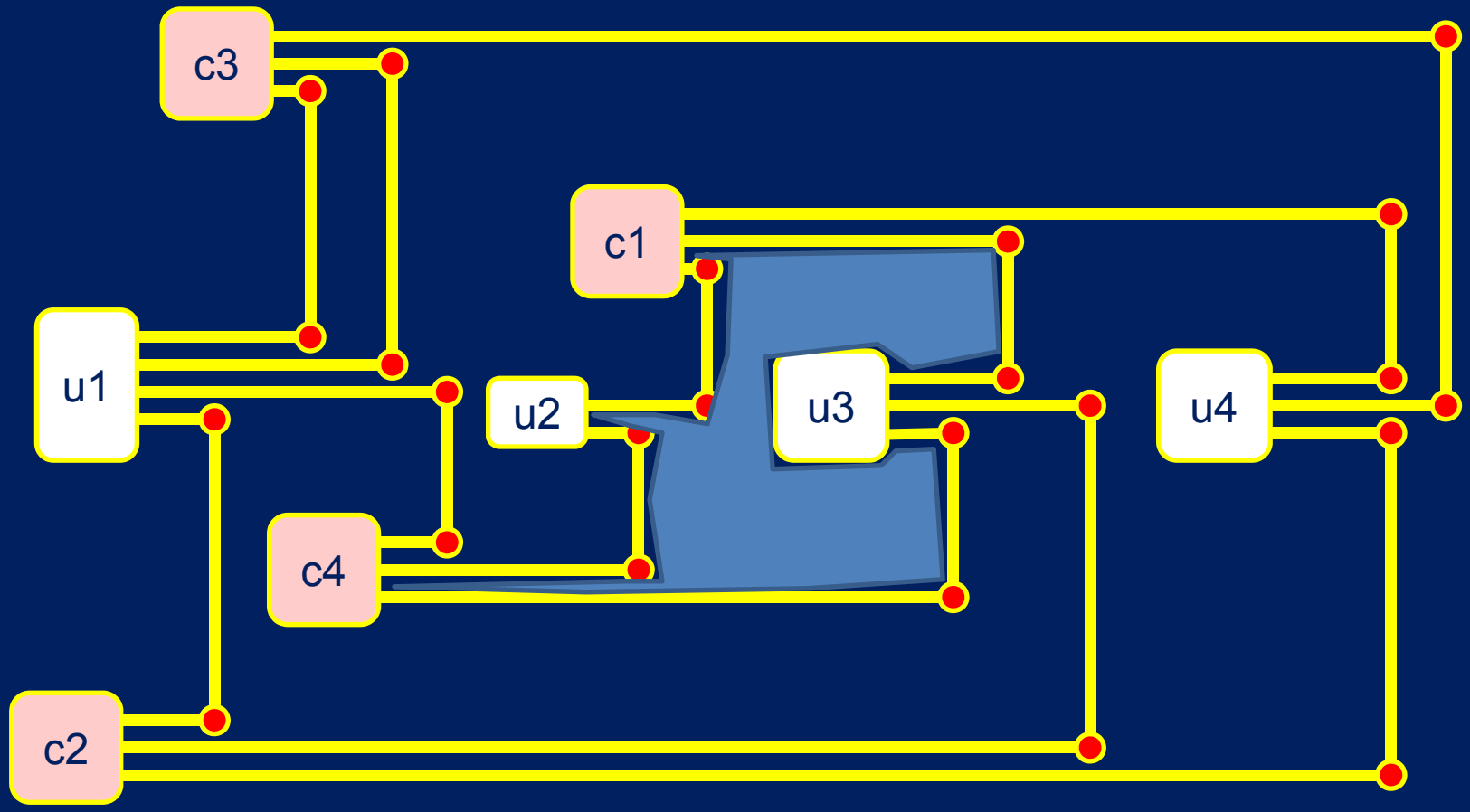
2. Enhance the drawing:
   - "node boxes" for
     - clauses c1, c2, …
     - variables u1, u2, …



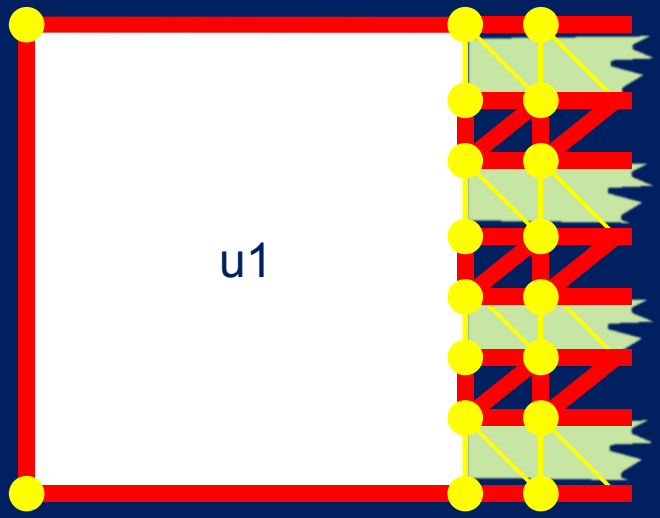
3. Transform to a 2-bend drawing
   - “pipes” to communicate between variables and clauses



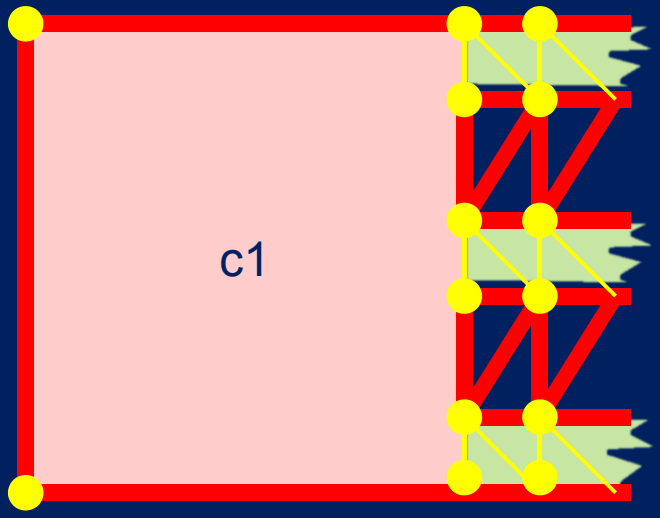
4. Transform to a no-bend drawing
   - extra nodes at bend points



5. Triangulate every face to make it impassable

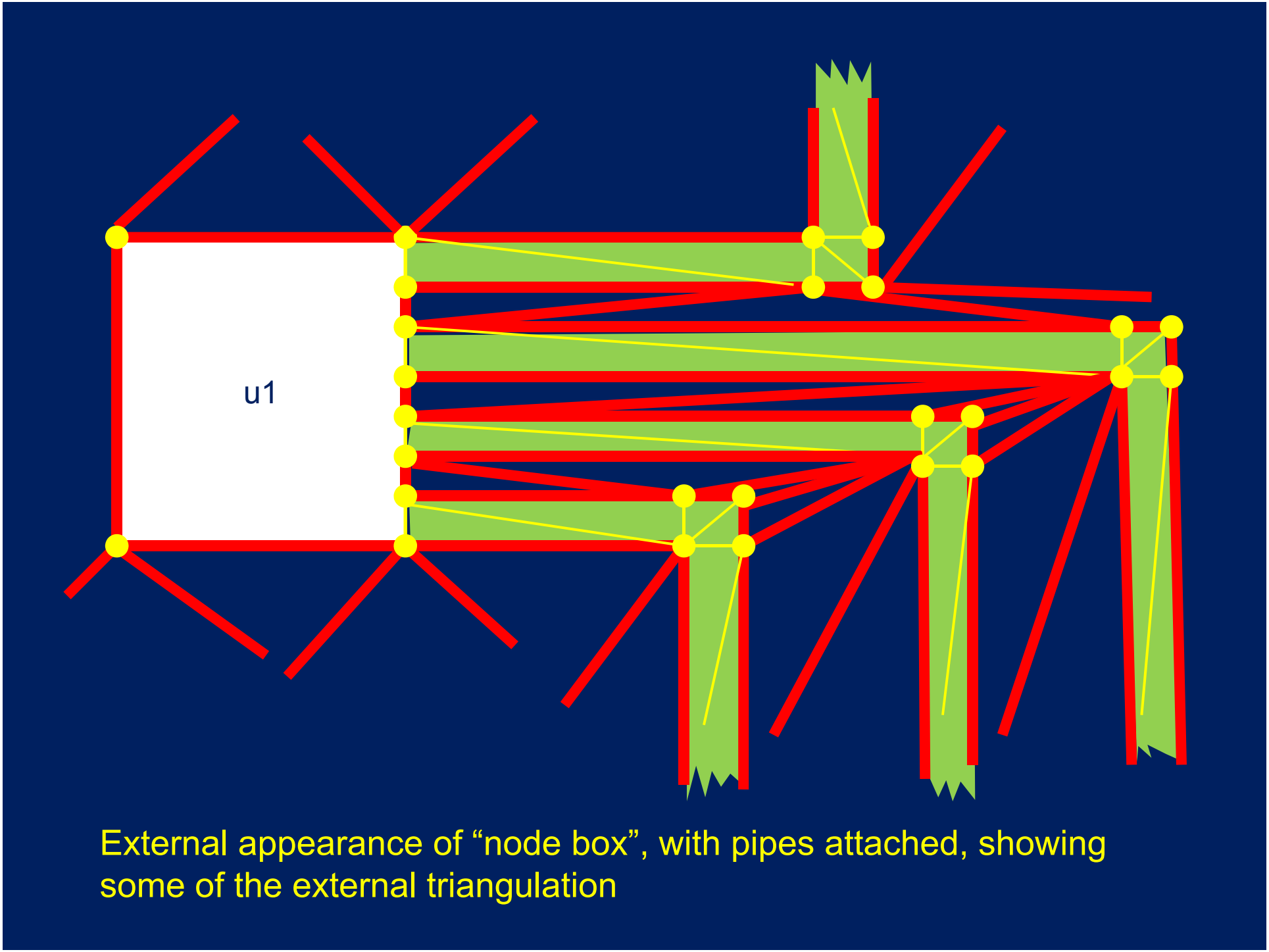


variable

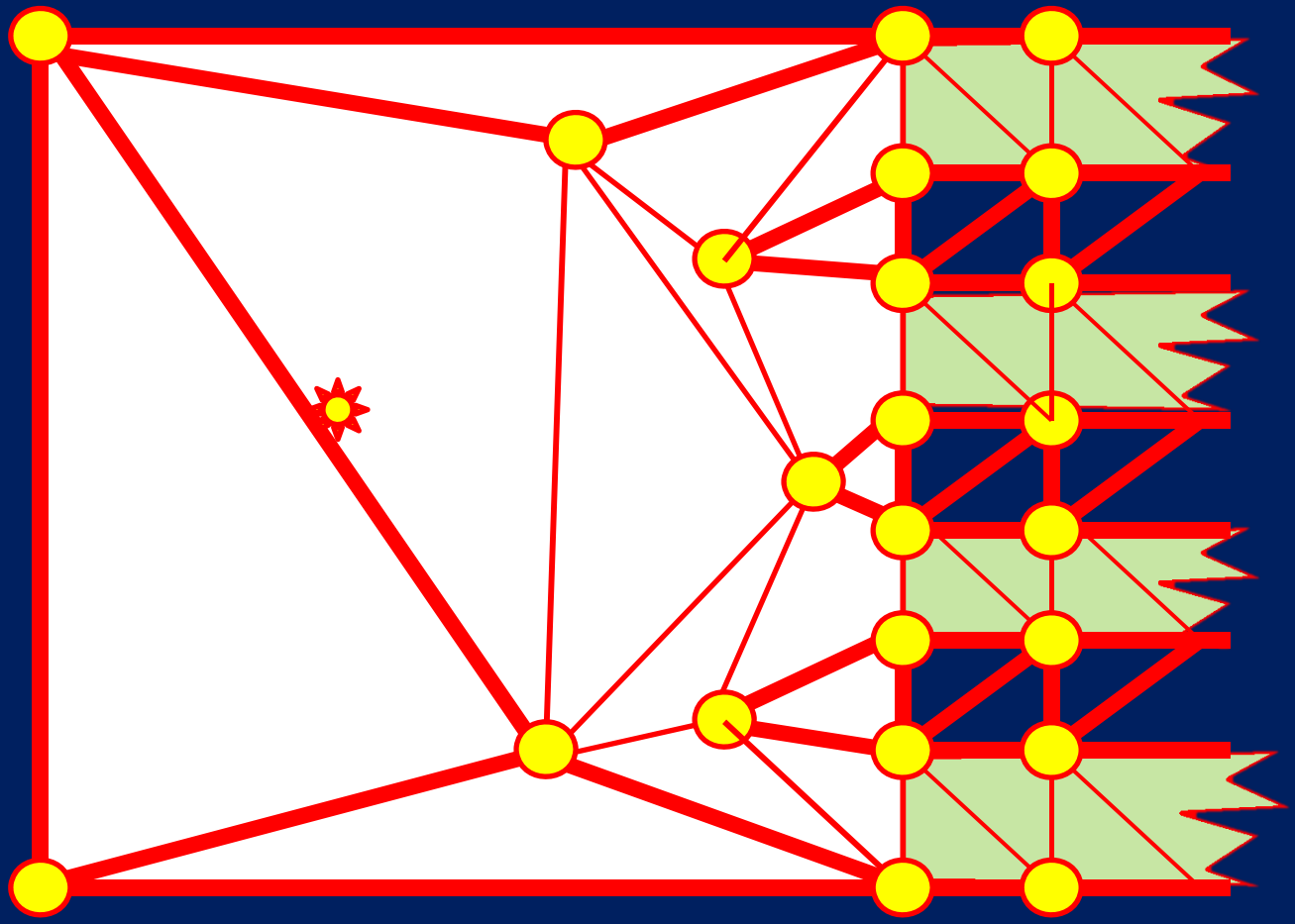


clause

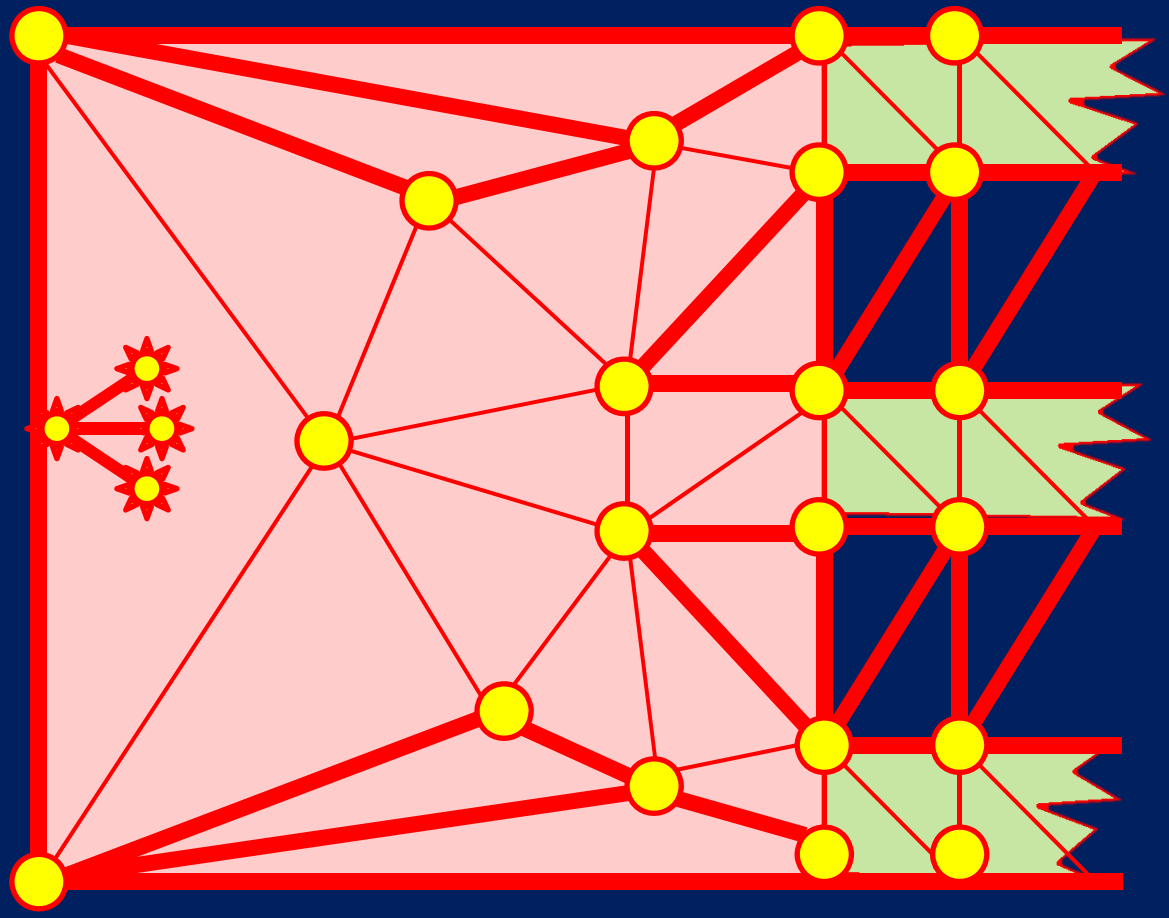
External appearance of “node boxes”, with “pipes” attached



External appearance of “node box”, with pipes attached, showing some of the external triangulation

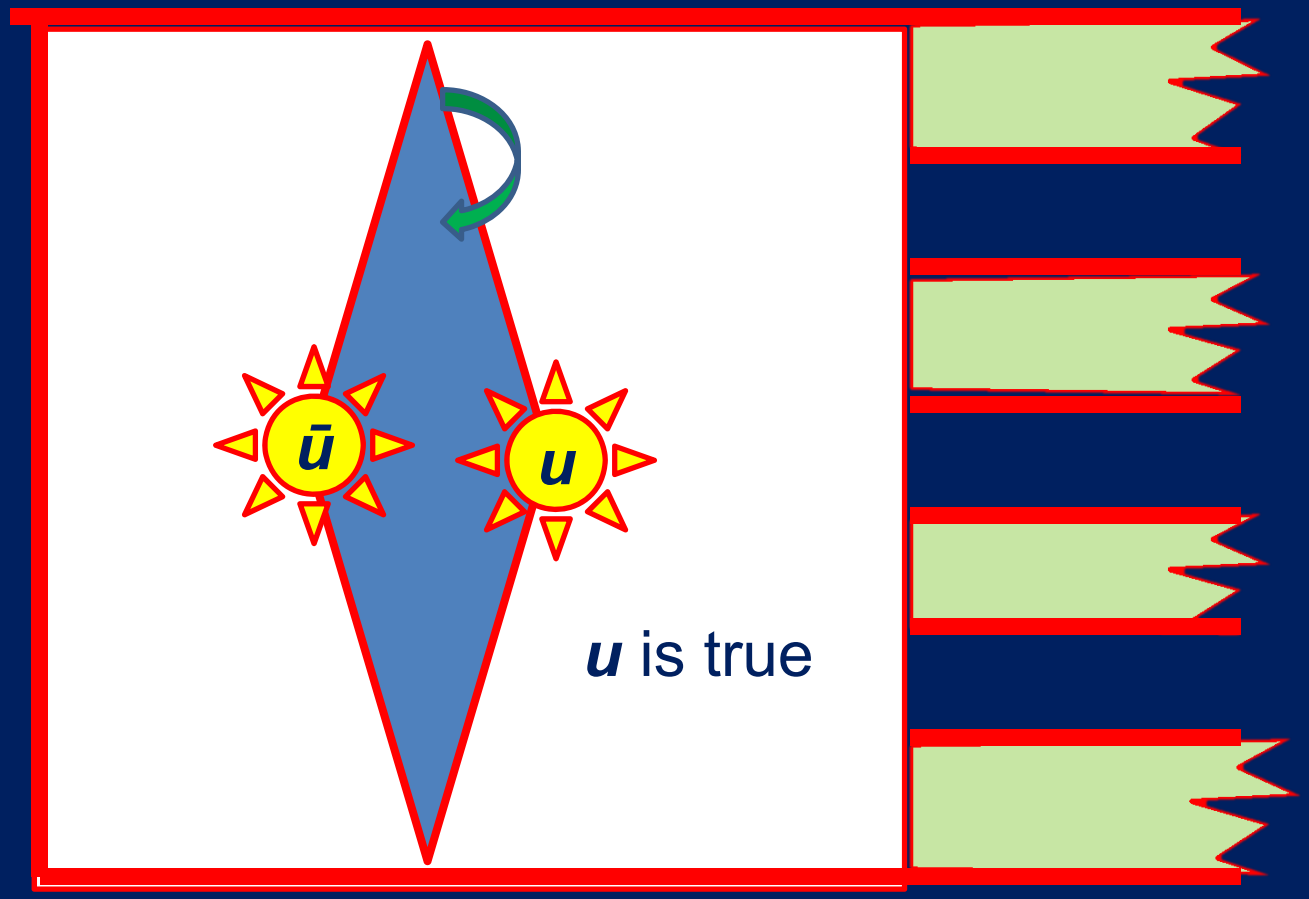


Variable gadget with pipes attached

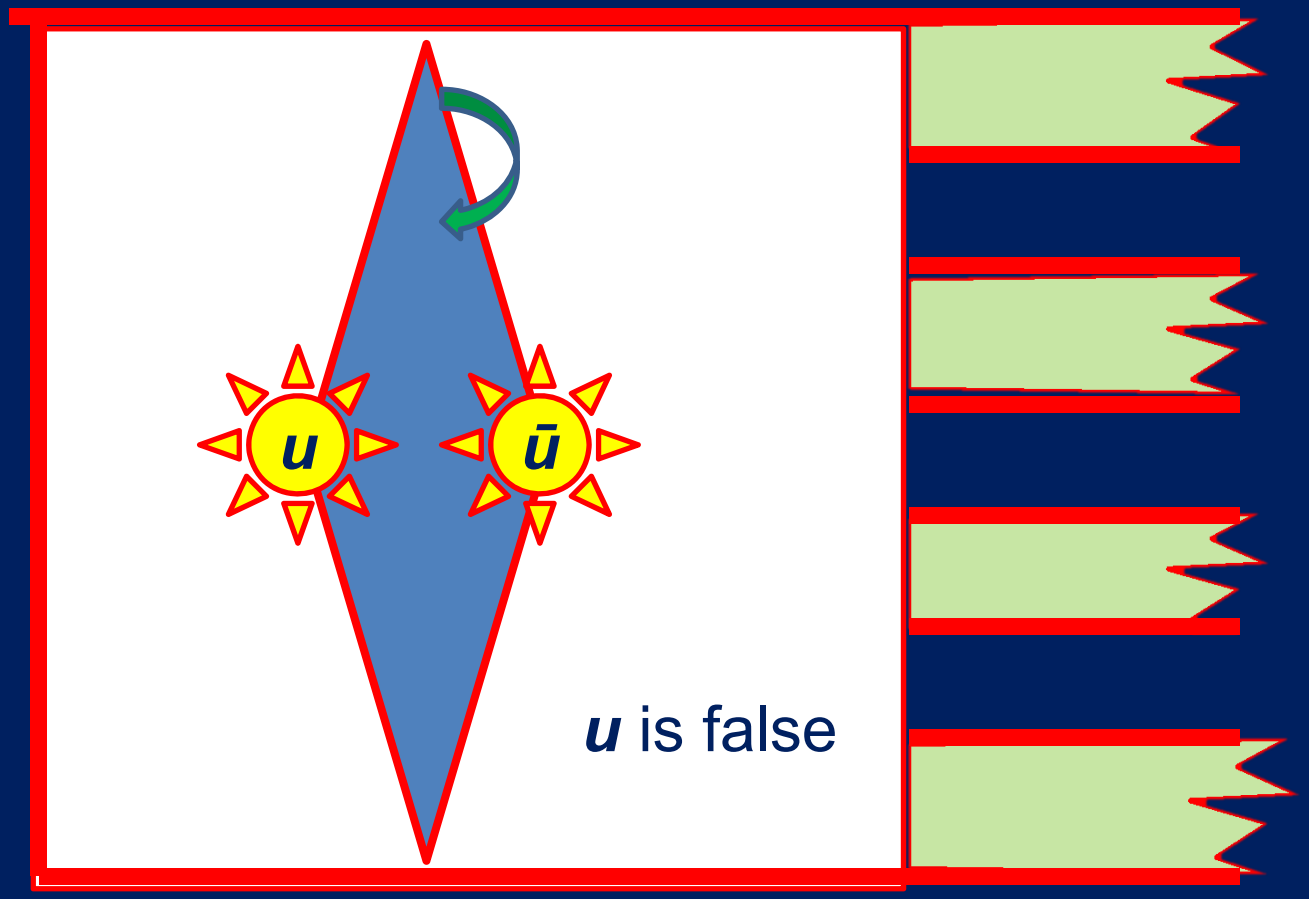


Clause gadget with pipes attached

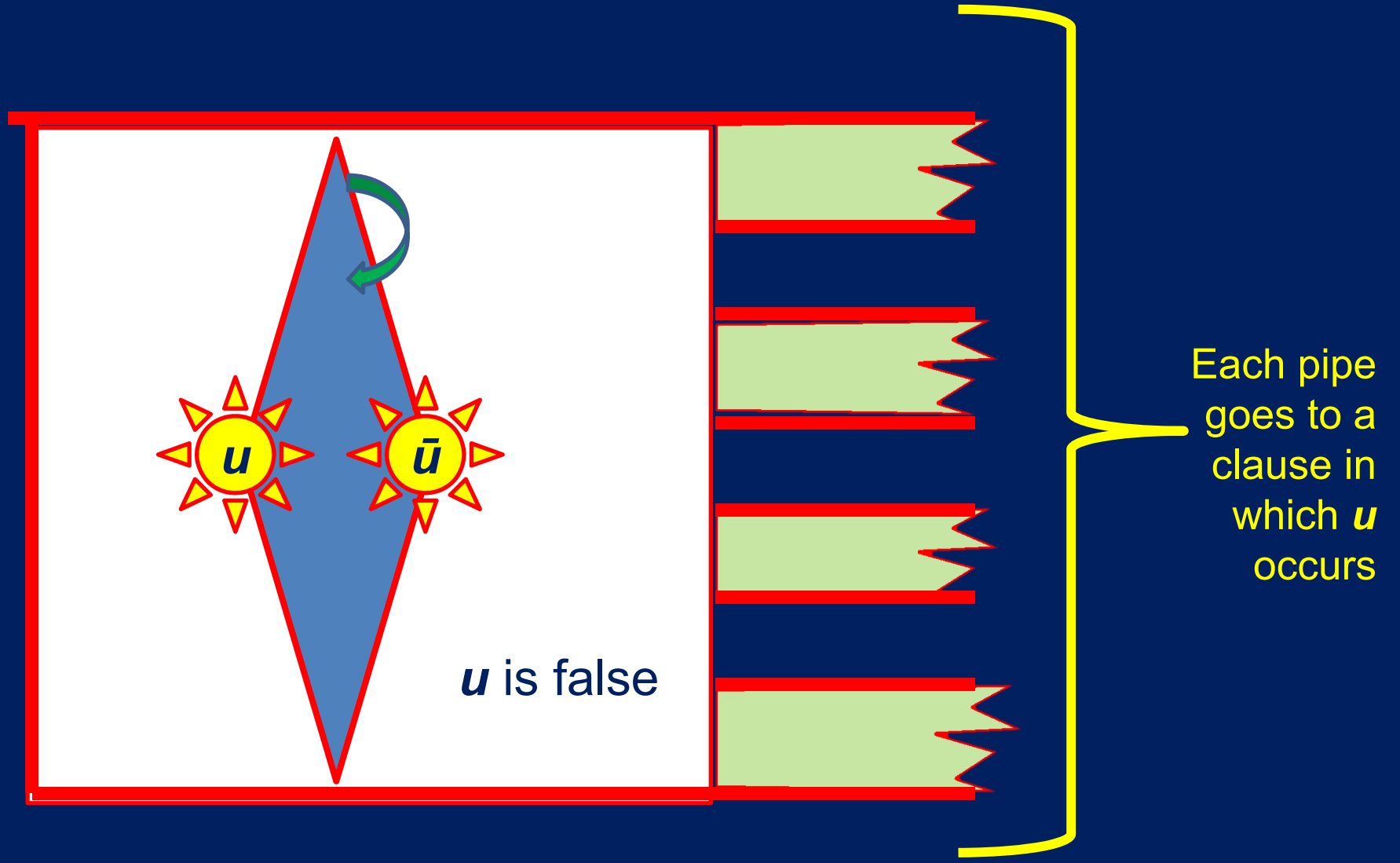
Logical view of variable gadget



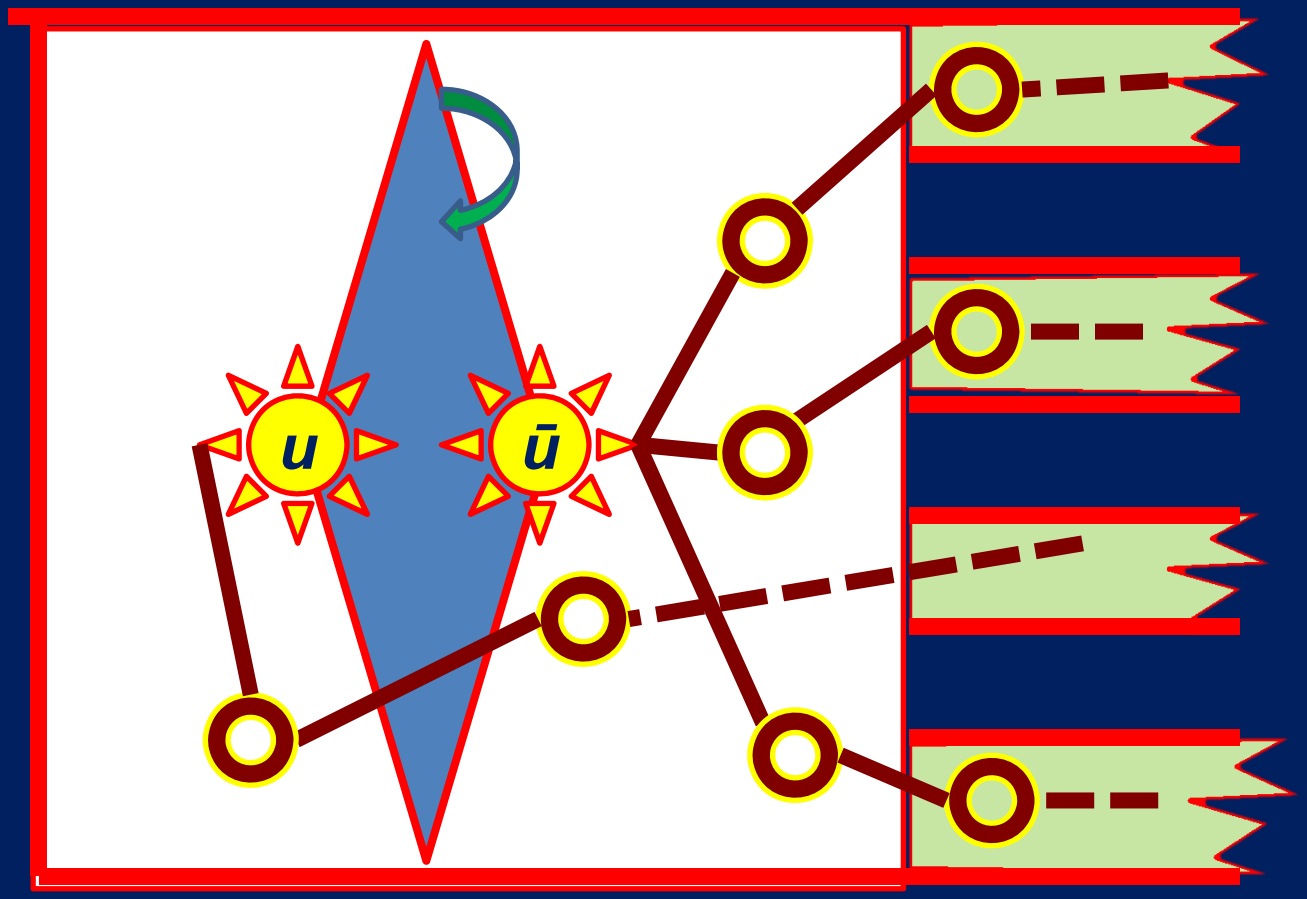
Logical view of variable gadget



Logical view of variable gadget

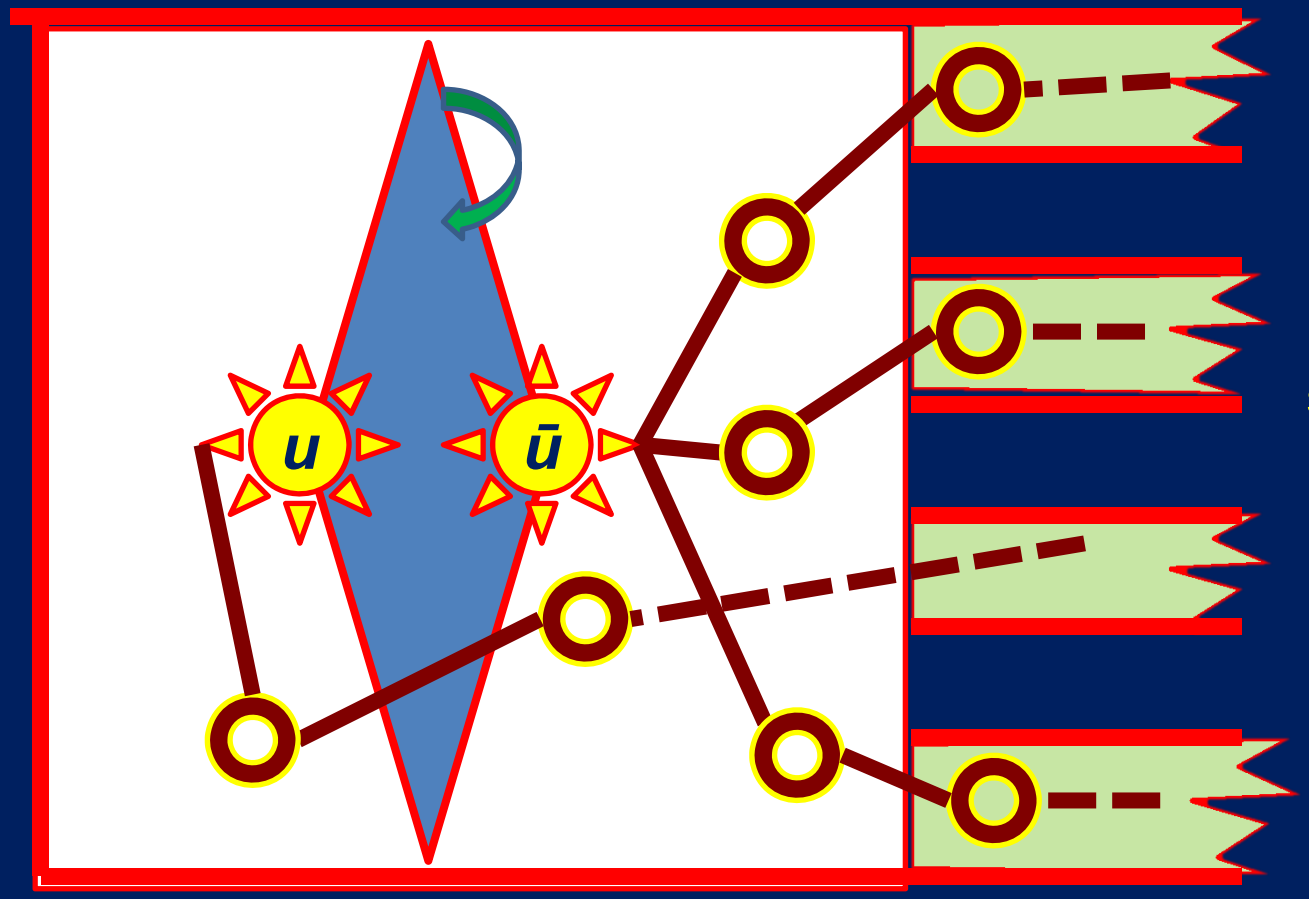


Logical view of variable gadget

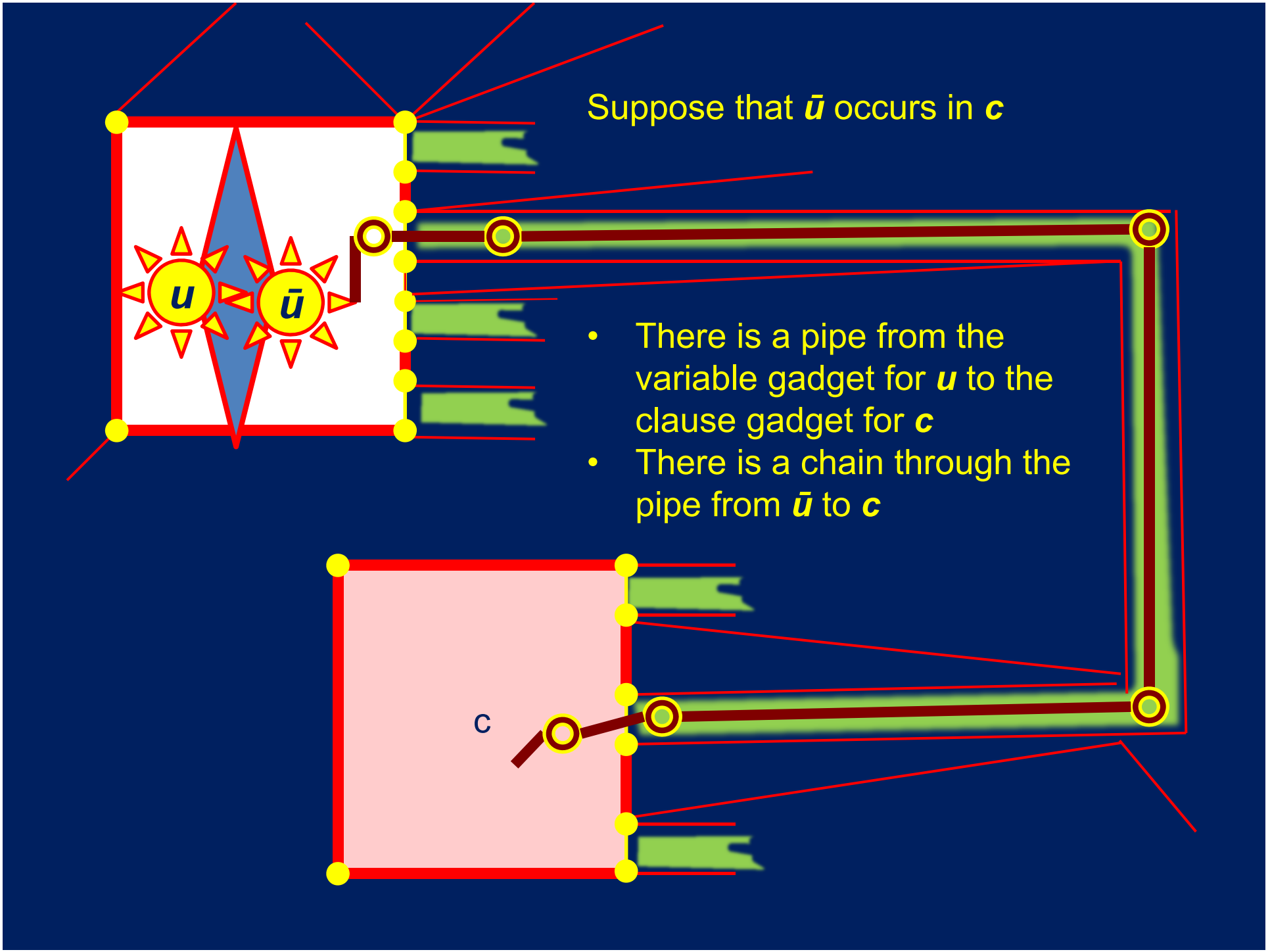


Literals are attached to the clauses in which they occur, using chains threaded through the pipes

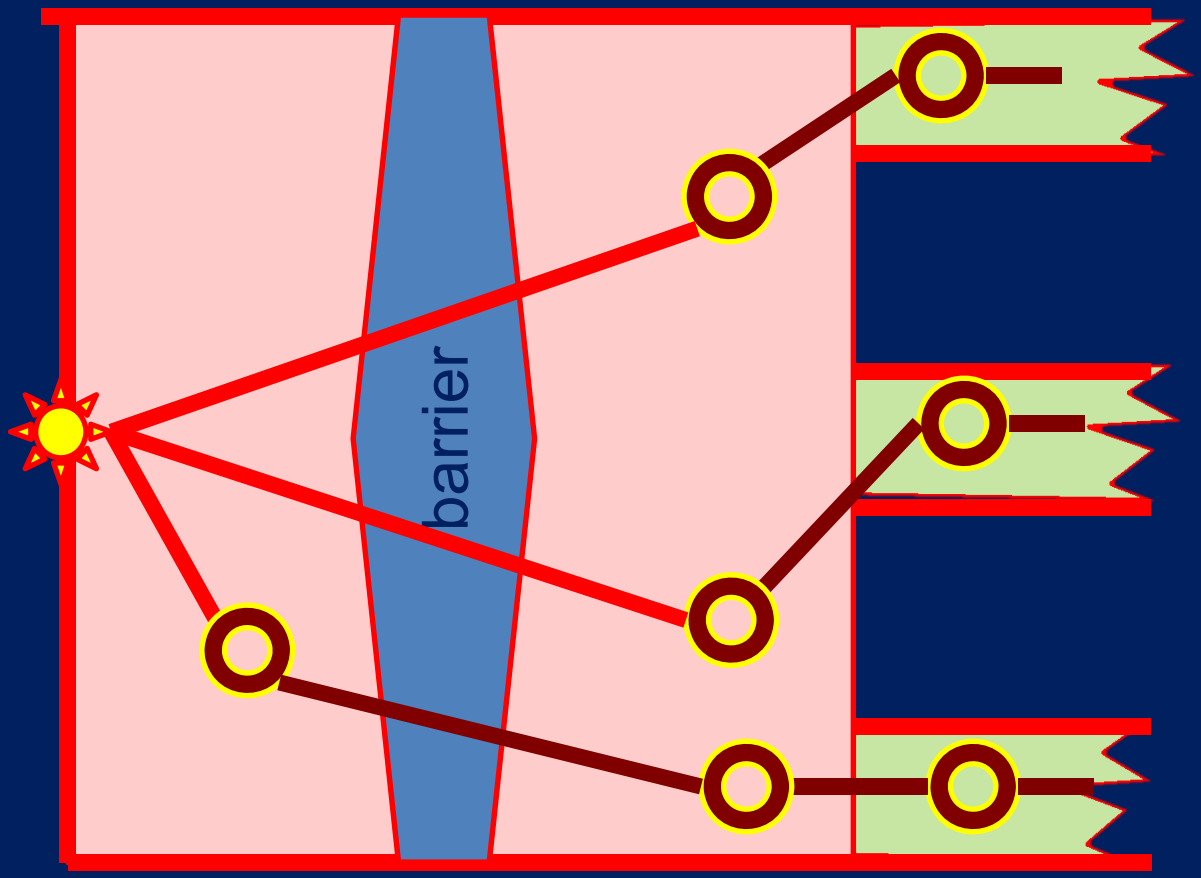
Logical view of variable gadget



Chains attached to the rear literal spend an extra link before getting into the pipe.


Suppose that $\bar{u}$ occurs in c
u
$\bar{u}$
• There is a pipe from the variable gadget for u to the clause gadget for c
• There is a chain through the pipe from $\bar{u}$ to c
c

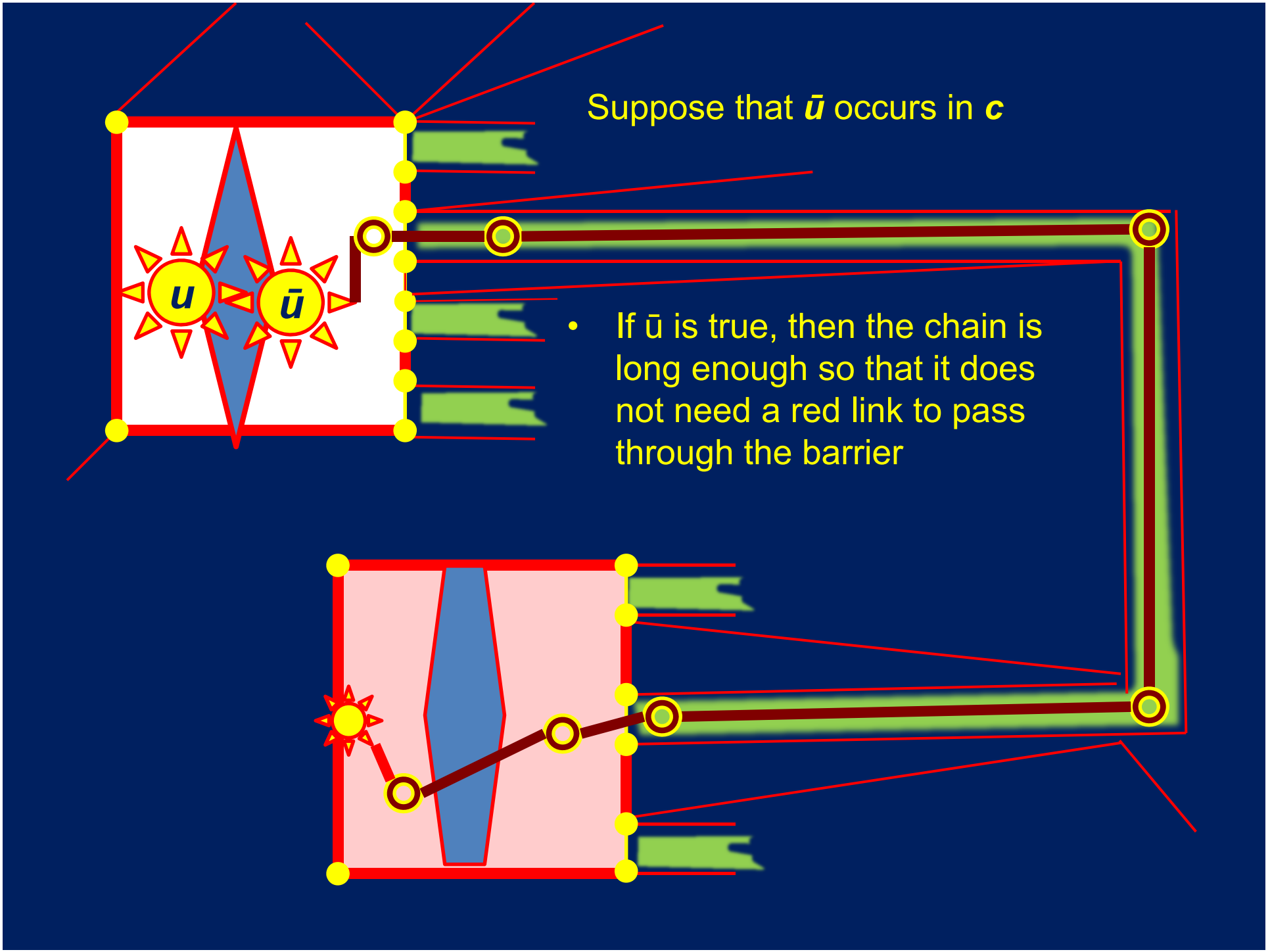
Logical view of clause gadget

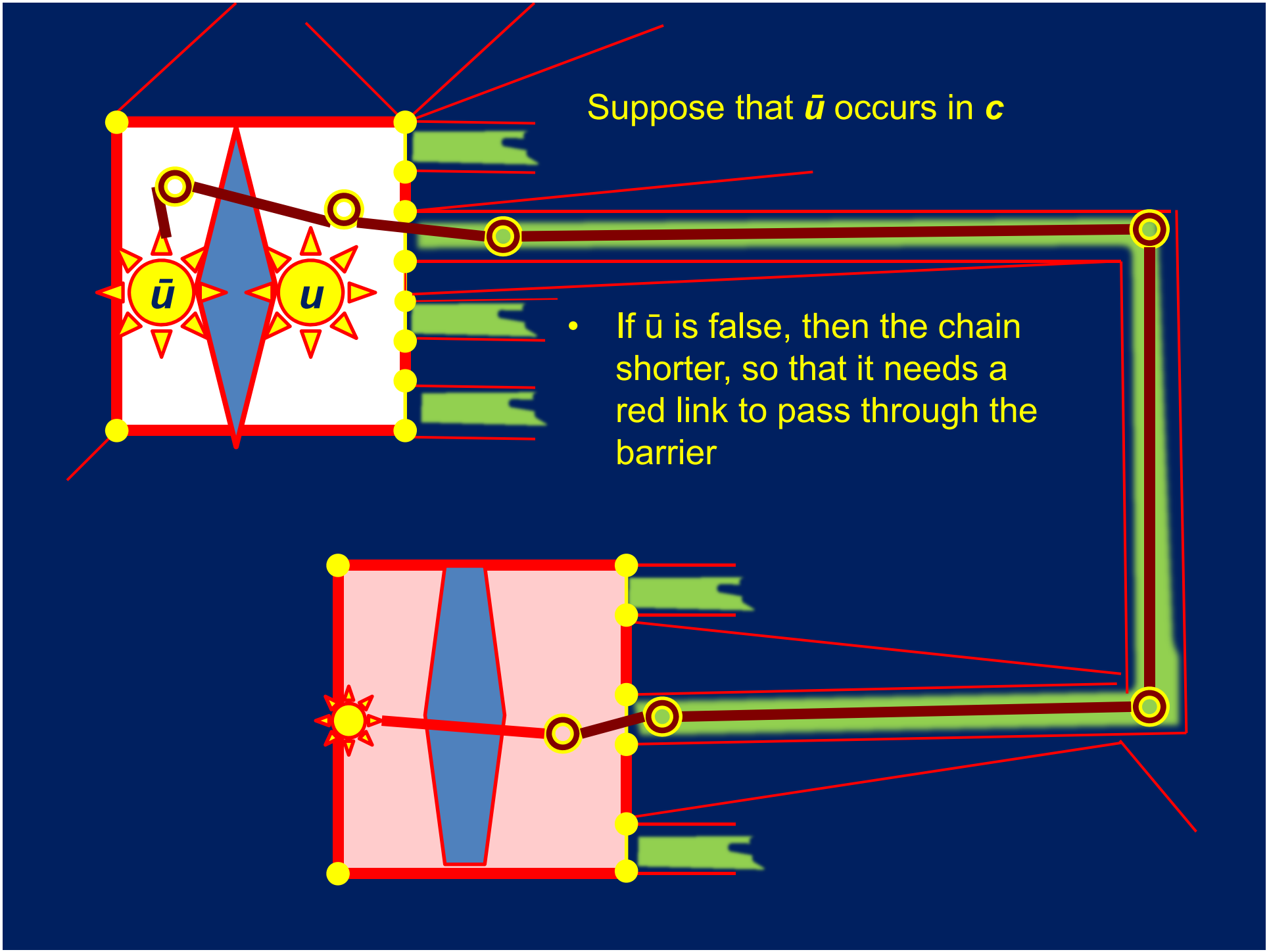


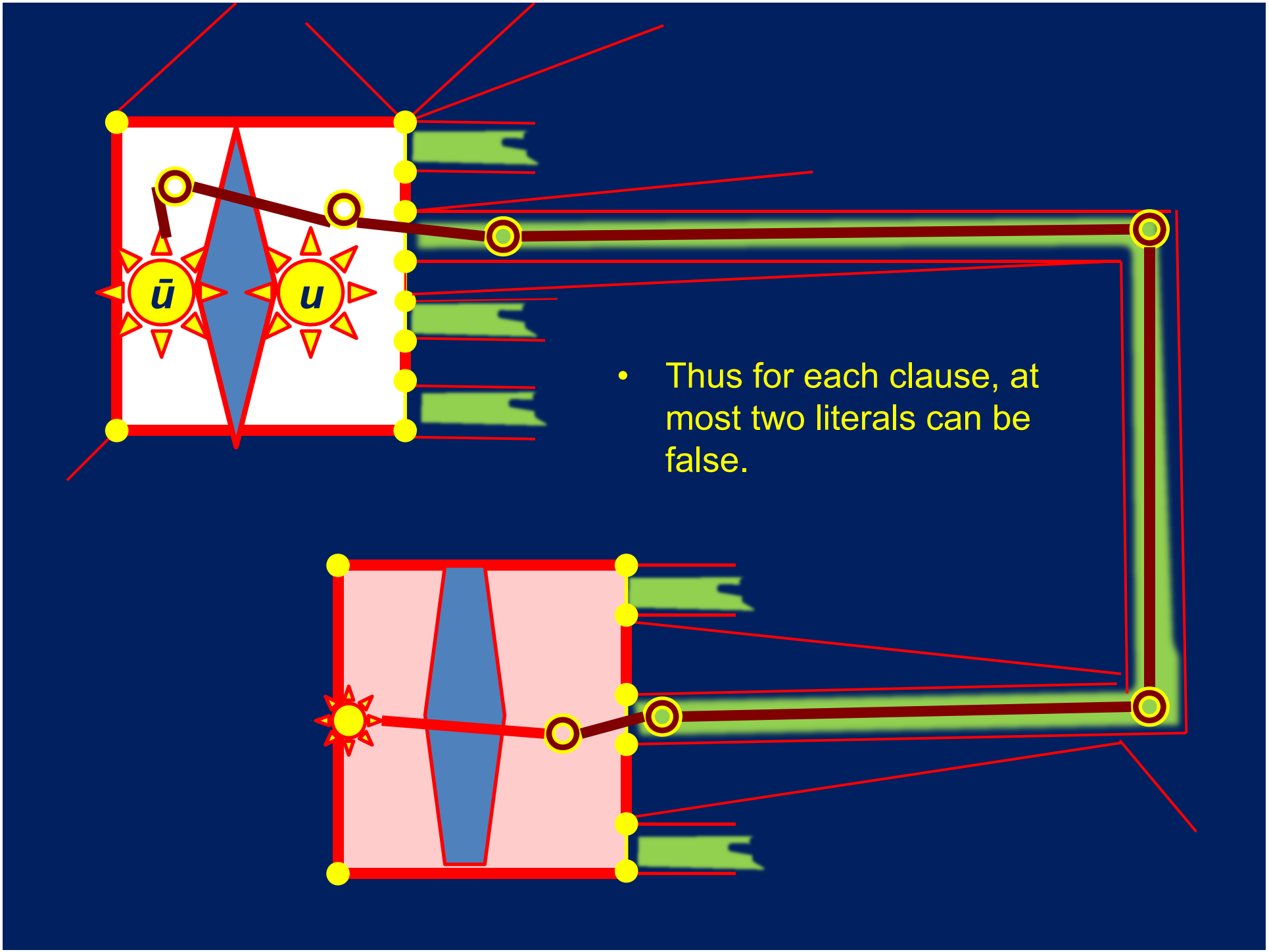
The barrier allows

- Any number of brown links to pass through
- At most two red links to pass through

Thus at least one chain needs to be long enough to reach past the barrier


Suppose that $\bar{u}$ occurs in c
u
ū
• If ū is true, then the chain is long enough so that it does not need a red link to pass through the barrier


Suppose that $\bar{u}$ occurs in c
$\bar{u}$
u
• If ū is false, then the chain shorter, so that it needs a red link to pass through the barrier



- Thus for each clause, at most two literals can be false.

Notes

- This is a *fairly generic proof strategy* for NP-hardness for layout problems.
- Details of clause and variable gadgets are straightforward but tedious
- The same proof works for *1-planar graphs*: just choose different gadgets for clauses and variables.

What about heuristics?

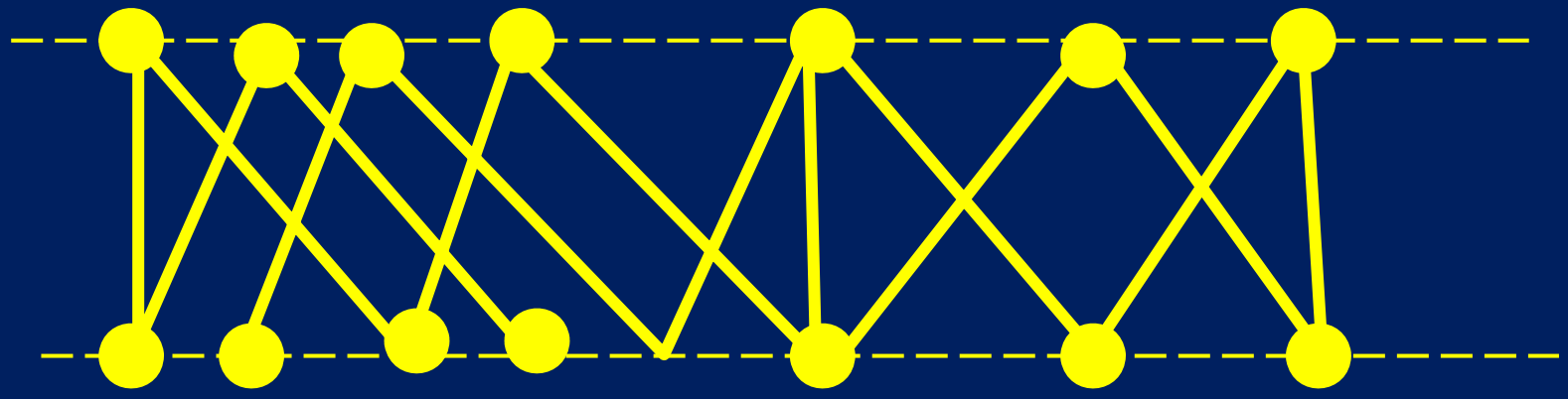
- Several force-directed methods (Huang, Nguyen, Hong, et al.) have been tested and seem to work OK to produce larger crossing angles than regular force-directed methods.

- No combinatorial heuristics known

What about special classes of graphs?

Theorem (Liotta, Didimo, di Giacomo, Eades)

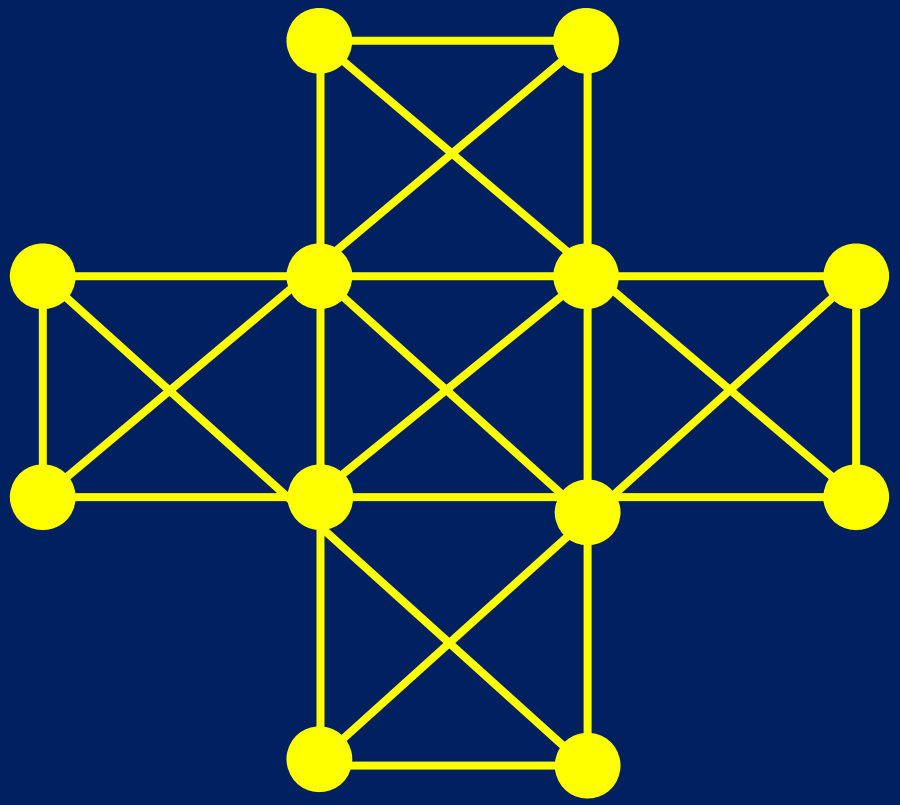
There is a linear time algorithm to test and draw graphs restricted to two layers with right-angle crossings.



What about special classes of graphs?

Theorem (Reisi)

There is a linear time algorithm to test and draw outer-1-planar graphs with right-angle crossings.



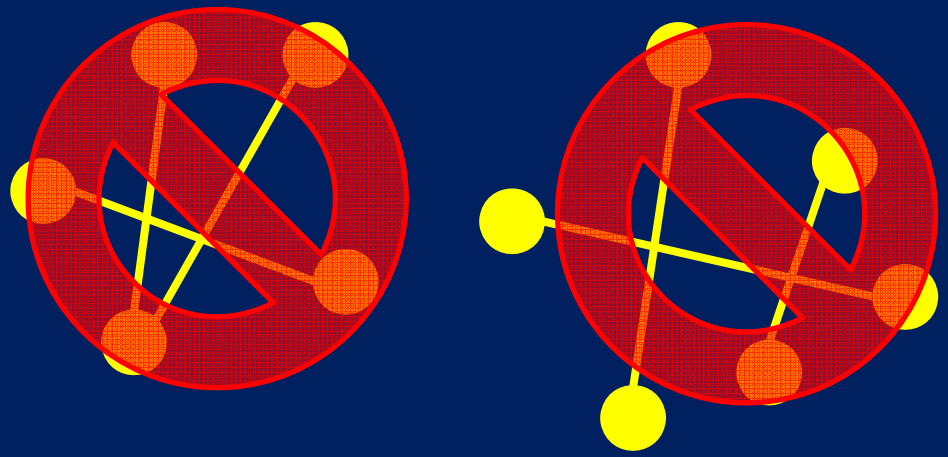
However:

- the problem of drawing graphs with large crossing angles remains mostly open, both from both practical and theoretical points of view.

More slightly-non-planar graphs

1-planar:

- No edge has more than one crossing



Questions for 1-planar graphs:

- How dense can a 1-planar graph be?

4n-8 (Ackerman-Tardos)

- How can you compute the topology of 1-planar graph?

NP-hard (Korzhik- Mohar)

- *Given the topology, how can you compute a drawing of a 1-planar graph?*

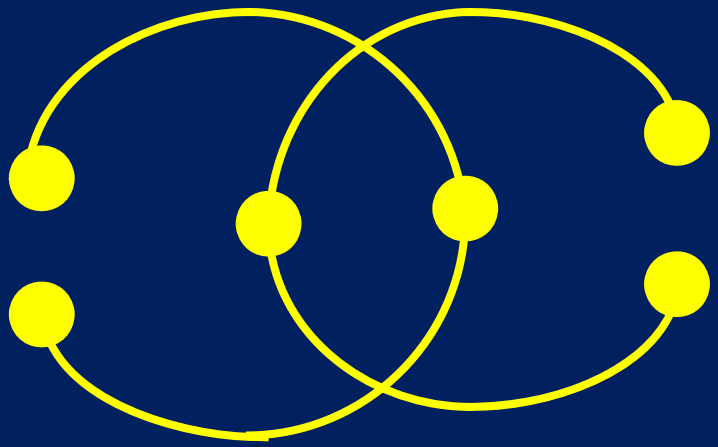
Recent algorithm (Hong-Eades-Liotta-Poon)

- *What is the relationship between 1-planar graphs and RAC graphs?*

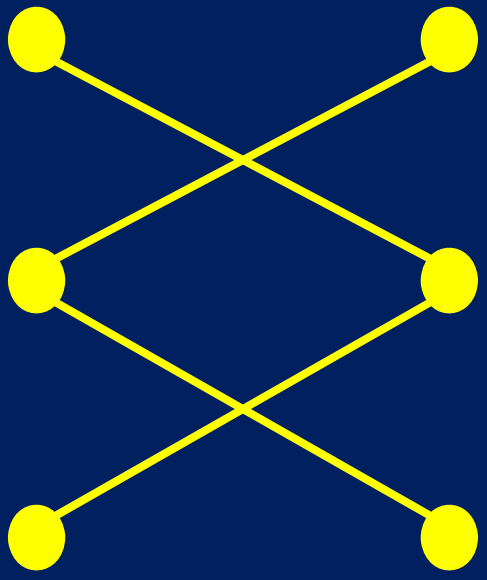
Recent theorem (Eades-Liotta)

Some 1-planar graphs have a 1-planar straight-line drawing,
others have no 1-planar straight-line drawing.

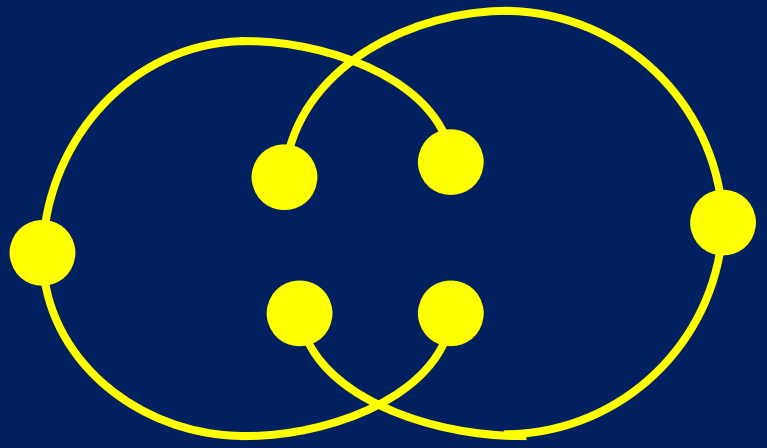
Chanel graph



The chanel graph has a straight-line drawing



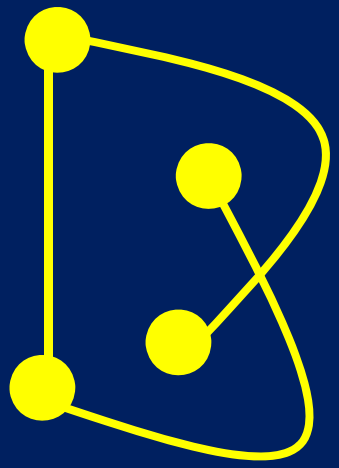
Gucci graph



Lemma:

The gucci graph has no straight-line drawing

Bulgari graph

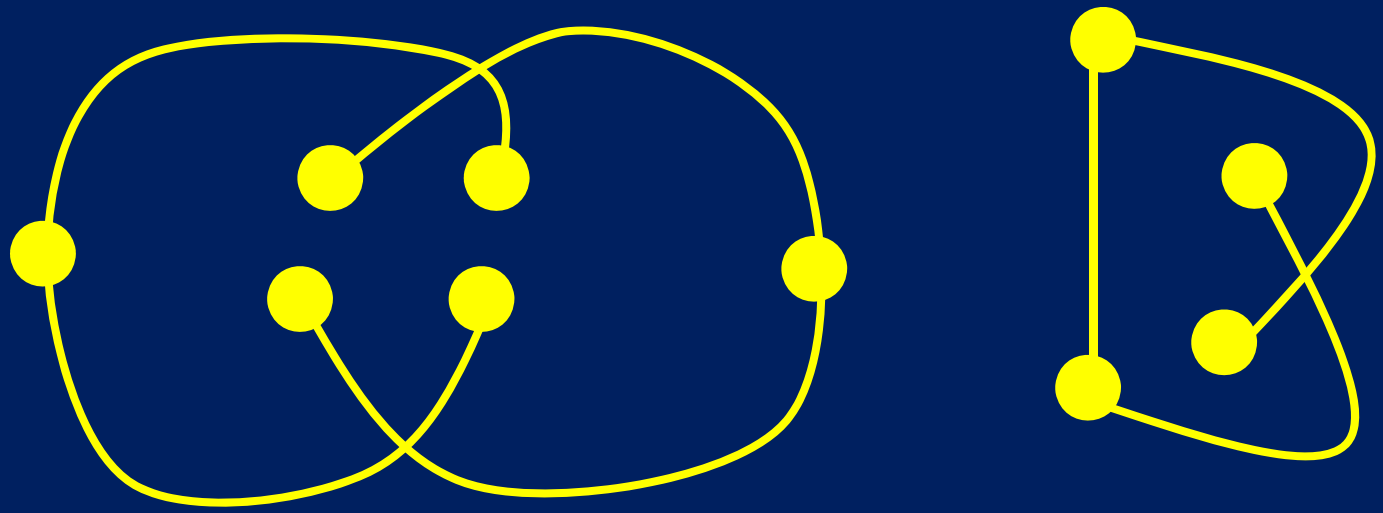


Lemma:

The bulgari graph has no straight-line drawing

Theorem (Hong, Liotta, Poon, Eades, 2011)

Suppose that ***G*** is a 1-planar topological embedding. Then ***G*** has a straight-line drawing if and only if ***G*** has no gucci subgraph and no bulgari subgraph.

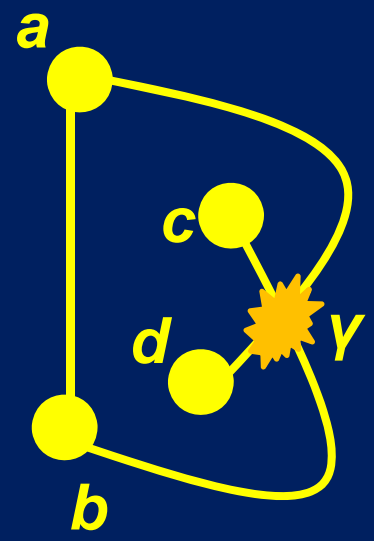


Theorem (Hong, Liotta, Poon, Eades, 2011)

Suppose that ***G*** is a 1-planar topological embedding. Then ***G*** has a straight-line drawing if and only if ***G*** has no gucci subgraph and no bulgari subgraph.

Proof

- *Necessity* for bulgari graph:
  - Angles in the triangle ***abγ*** add up to ***π***
  - Thus 3 angles at ***γ*** add up to strictly less than ***π***
  - This is impossible if ***ad*** and ***bc*** are straight lines.
- Similar argument for the gucci graph



Theorem (Hong, Liotta, Poon, Eades, 2011)

Suppose that ***G*** is a 1-planar topological embedding. Then ***G*** has a straight-line drawing if and only if ***G*** has no gucci subgraph and no bulgari subgraph.

Proof

- *Sufficiency*
  - Much more complicated
  - Provides an algorithm:

    Input: a 1-planar topological embedding ***G*** with no gucci subgraph and no bulgari subgraph.

    Output: a straight line drawing of ***G***

Algorithm for Straight-line 1-planar graph drawing

Input: a 1-planar topological embedding ***G***

Output: a straight line drawing of ***G***, if it exists

1. *Testing:* Test whether ***G*** has a gucci subgraph or a bulgari subgraph.
2. *Augmentation:* Add non-crossing edges to make an augmented graph G^+, with higher connectivity.
3. *Drawing*: Apply a convex drawing algorithm together with an SPQR trees to G^+.

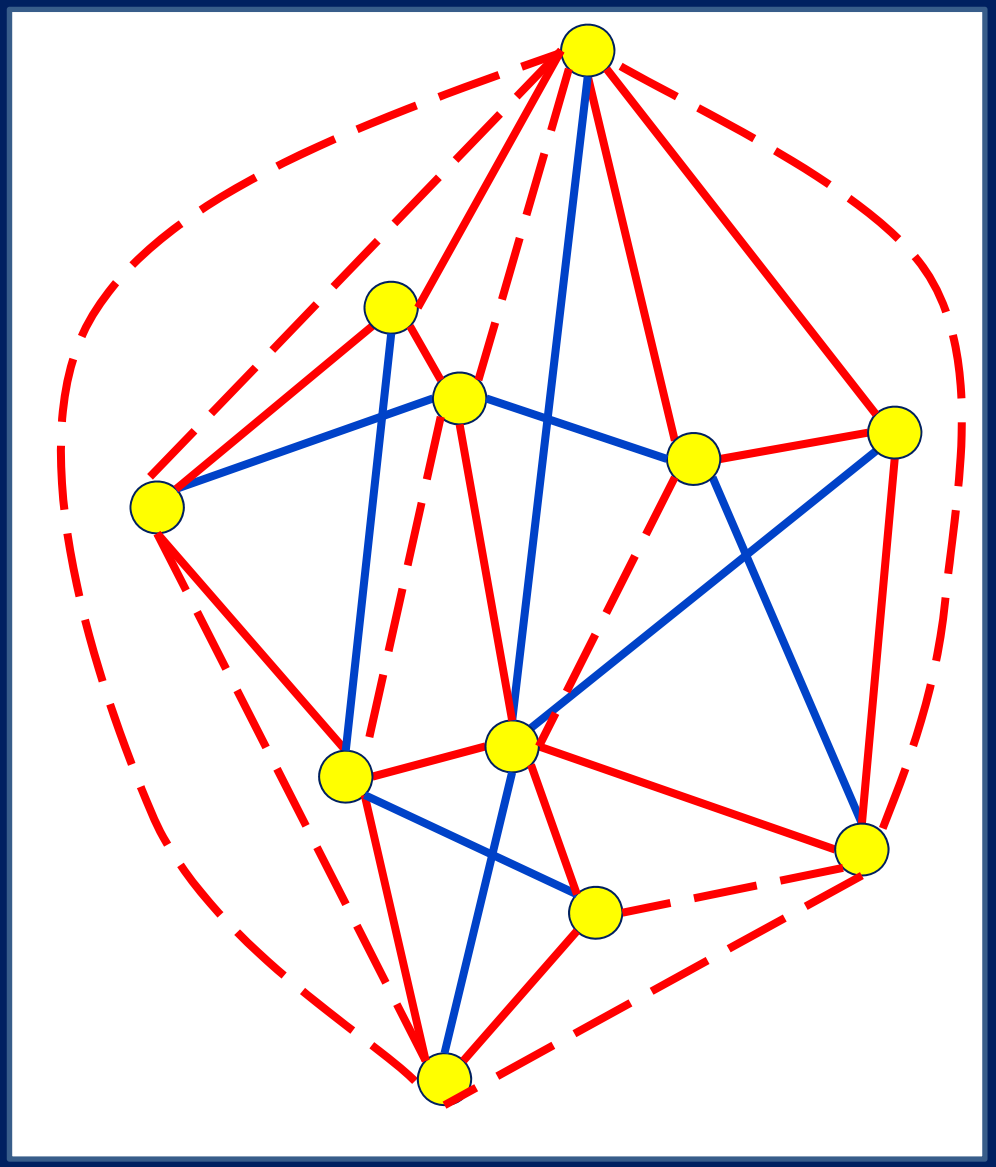
1. *Testing:* Test whether ***G*** has a gucci subgraph and no bulgari subgraph.
2. *Augmentation:* Add non-crossing edges to make an augmented graph G^+, with higher connectivity.
3. *Drawing*: Use a convex drawing algorithm together with an SPQR trees to G^+.

Red and blue edges

- *Red* edges: no crossings
- *Blue* edges: 1 crossing

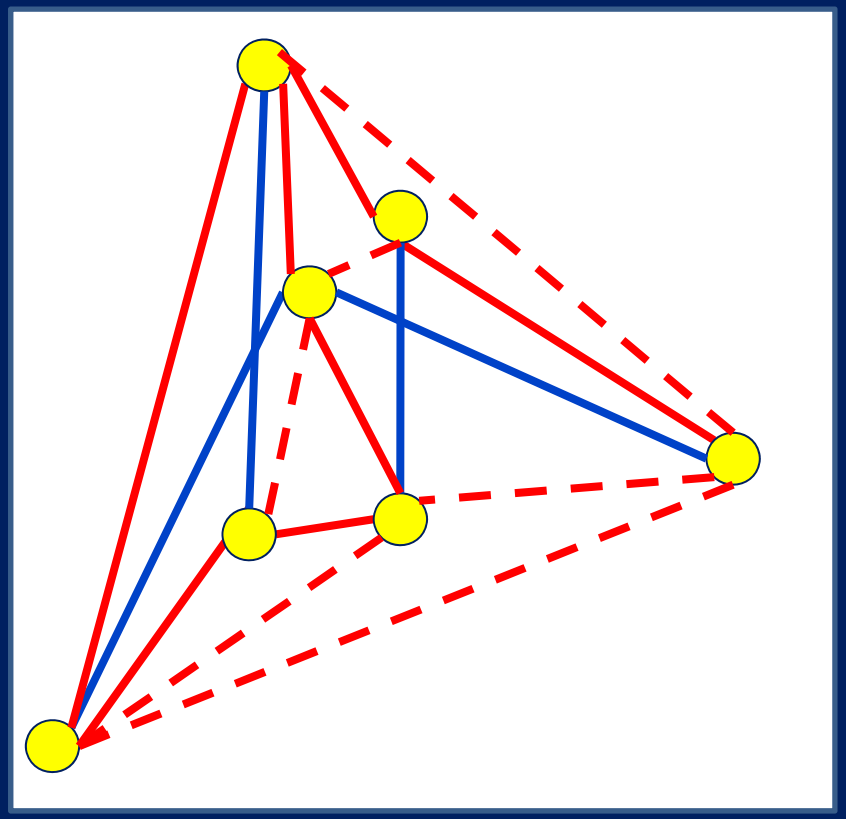
Red-maximal:

- A graph is *red-maximal* if adding an edge causes a crossing
- ie, a graph is *red-maximal* if whenever a pair ***a, b*** of vertices share a face, then ***(a,b)*** is an edge.



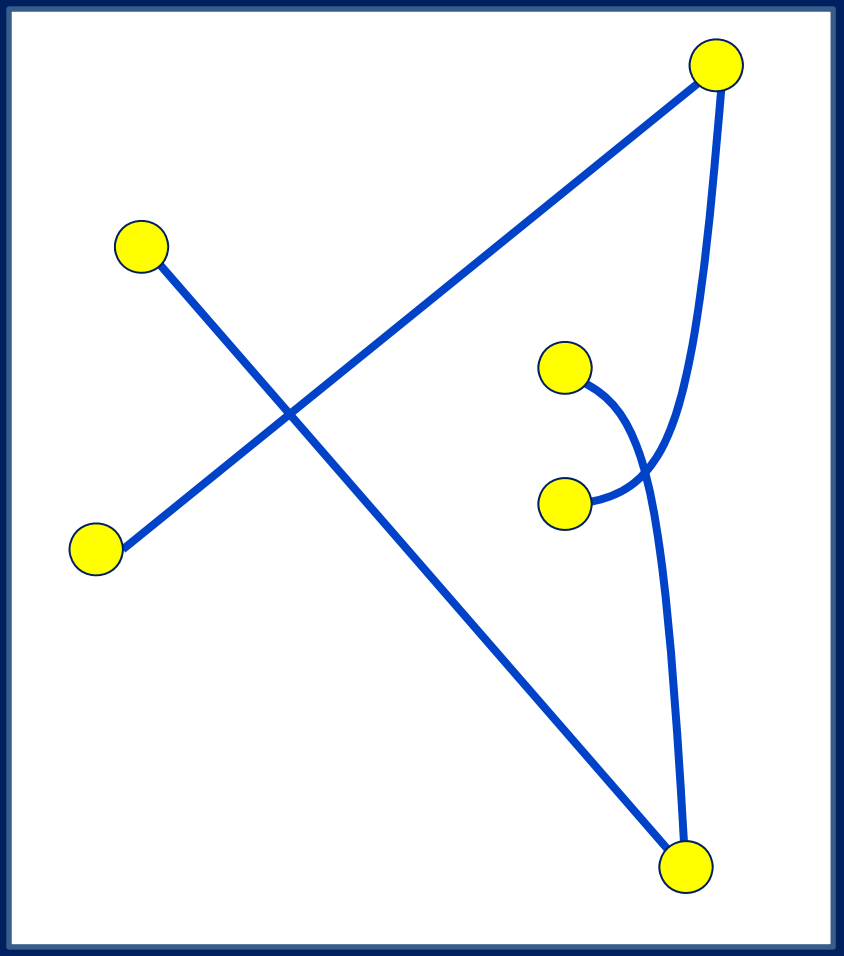
Naive augmentation algorithm
to obtain red-maximality:

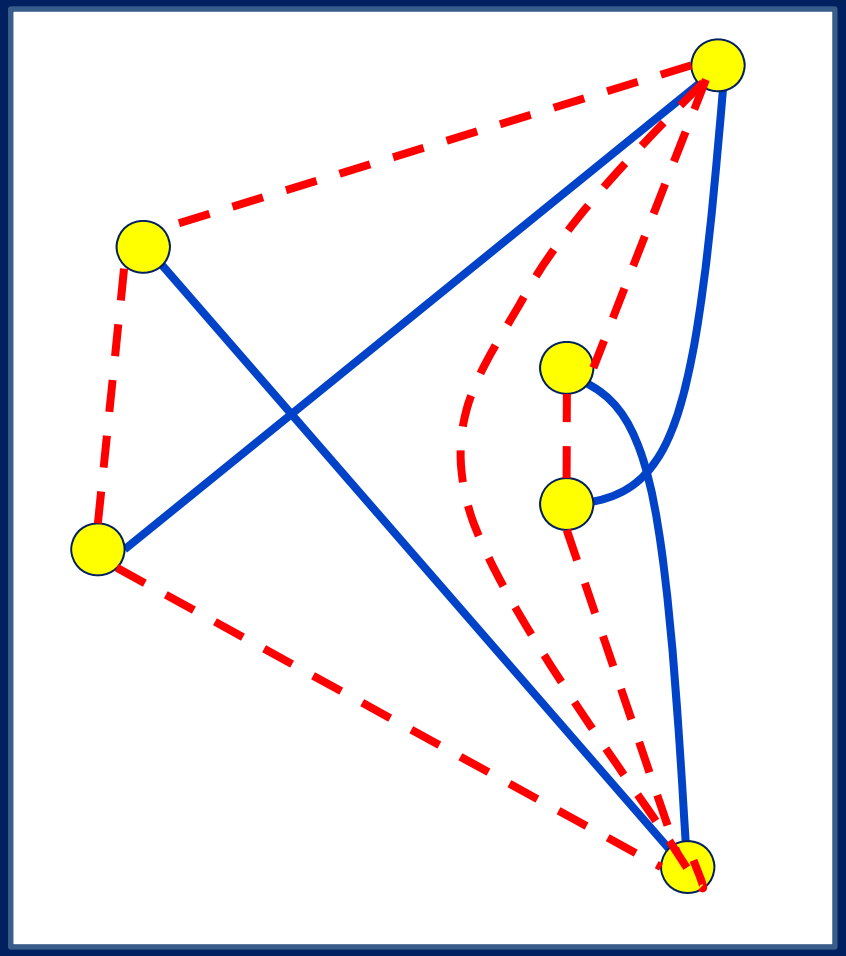
For each nonadjacent pair
a,b *of vertices that share*
a face, add the edge
(a,b)



- But: the naive algorithm does not work ... ...

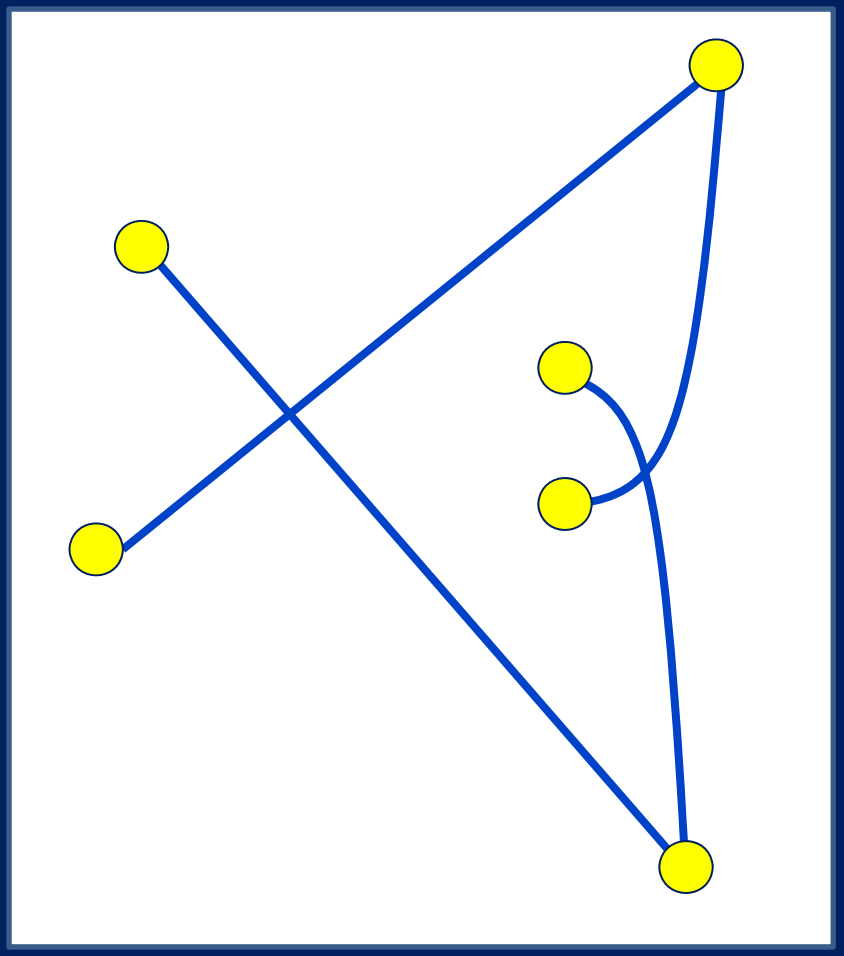
Some augmentations may not be correct

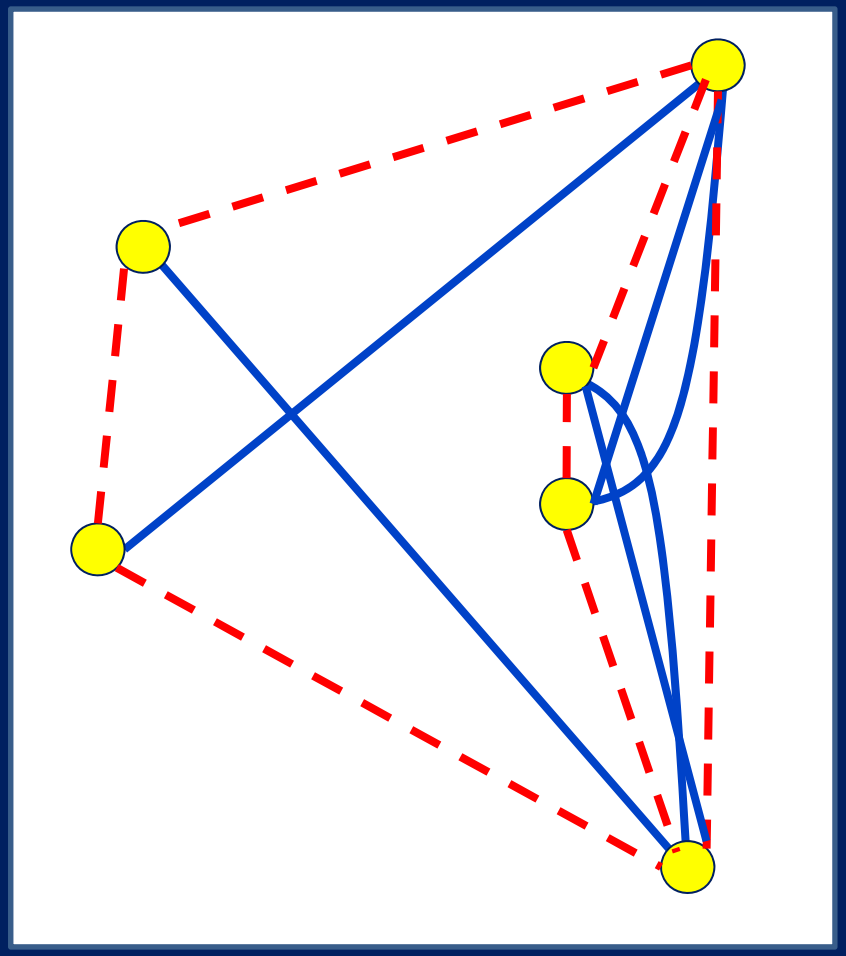




Augmentation 1: introduces a bulgari subgraph

Some augmentations may not be correct





Augmentation 2: no bulgari subgraph

1. *Testing:* Test whether ***G*** has a gucci subgraph and no bulgari subgraph.
2. *Augmentation:* Add non-crossing edges to make an augmented graph G^+, with higher connectivity.
3. *Drawing*: Use a convex drawing algorithm together with an SPQR trees to G^+.

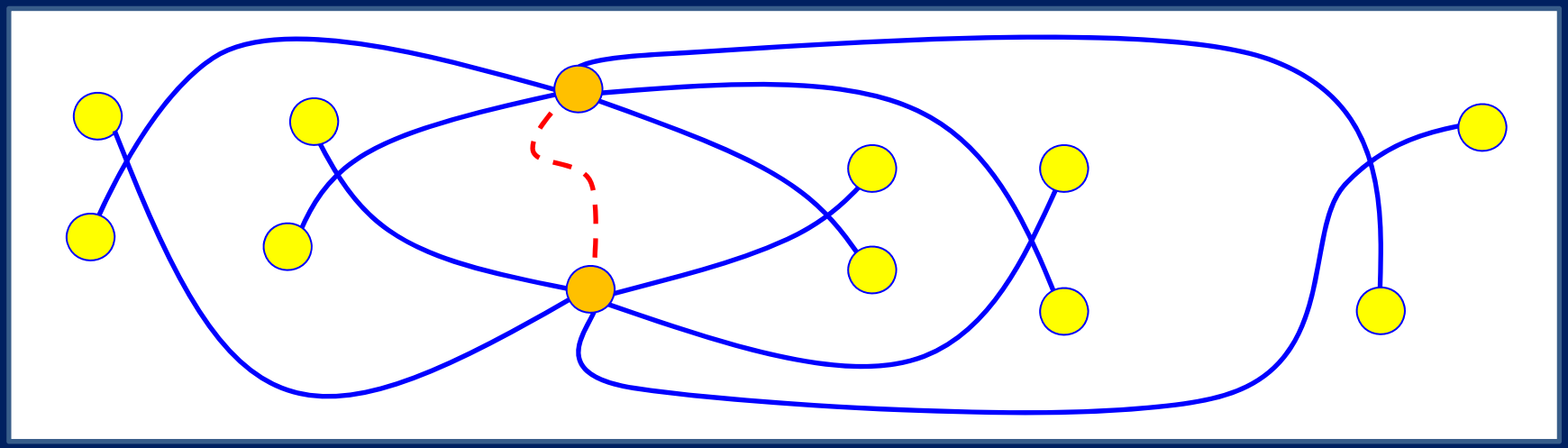
Problem:

- Naive algorithm does not work.
- Red-maximal supergraph is not unique
  - We need to choose a red-maximal supergraph with no bulgari/gucci subgraphs

1. *Testing:* Test whether ***G*** has a gucci subgraph and no bulgari subgraph.
2. *Augmentation:* Add non-crossing edges to make an augmented graph G^+, with higher connectivity.
3. *Drawing*: Use a convex drawing algorithm together with an SPQR trees to G^+.

Augmentation algorithm

- Augmentation can be done an edge at a time, but when two crossings share two endpoints, we must be carefully order the edge insertions.
- Can be done in linear time.



1. *Testing:* Test whether ***G*** has a gucci subgraph and no bulgari subgraph.
2. *Augmentation:* Add non-crossing edges to make an augmented graph G^+, with higher connectivity.
3. *Drawing*: Use a convex drawing algorithm together with an SPQR trees to G^+.

Drawing Algorithm

If the red subgraph ***R*** of G^+ is triconnected

Then

- Draw ***R*** using a convex drawing algorithm
- Insert crossing edges

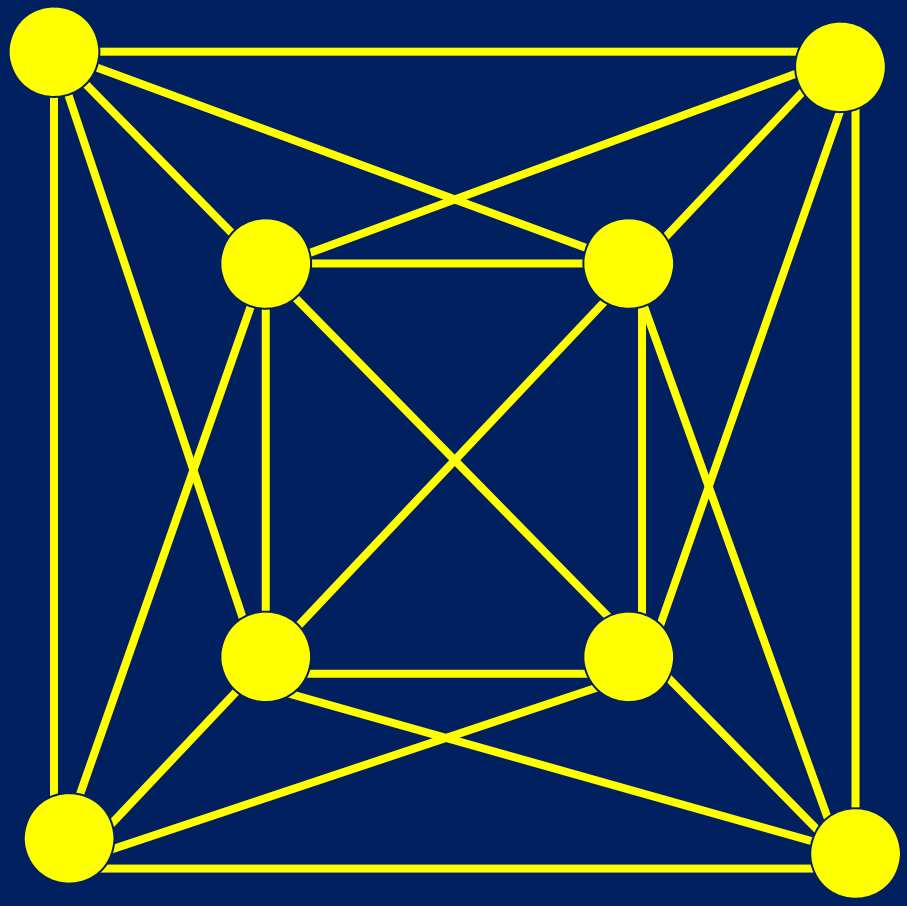
Else

- Use SPQR tree in a recursive way to build a drawing.

Relationships between the various classes of slightly non-planar graphs:-

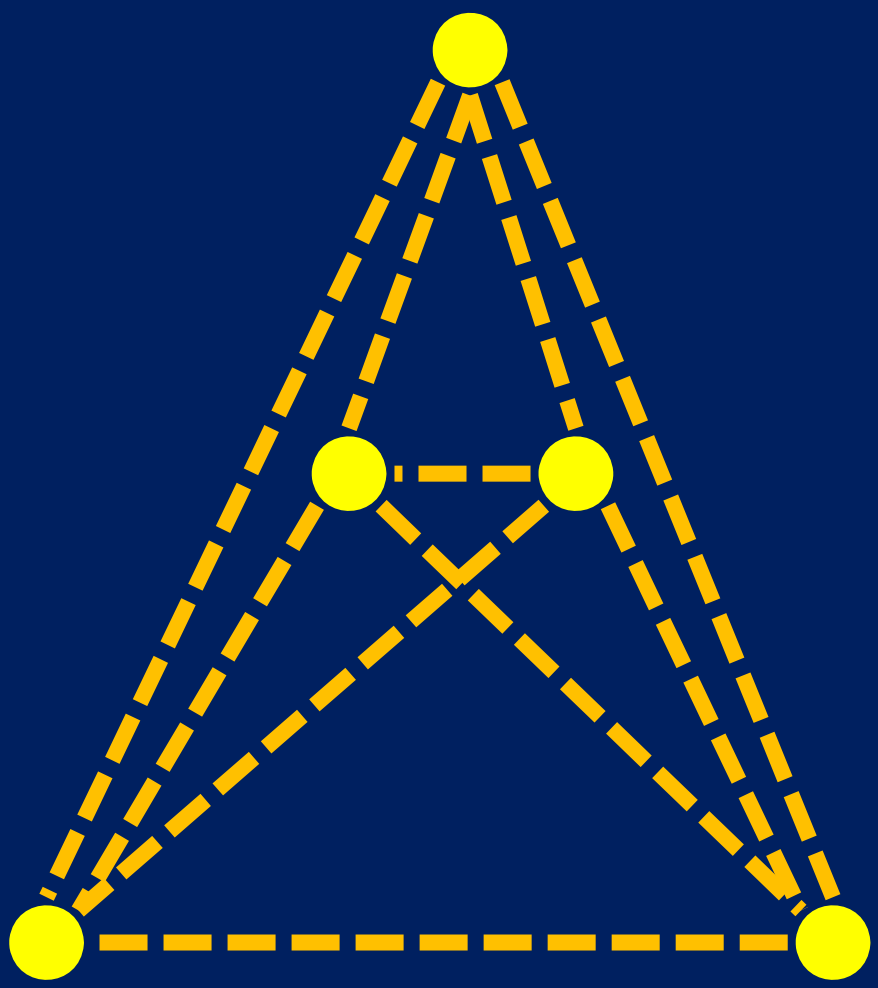
Observation1

There are 1-planar graphs that are not RAC.

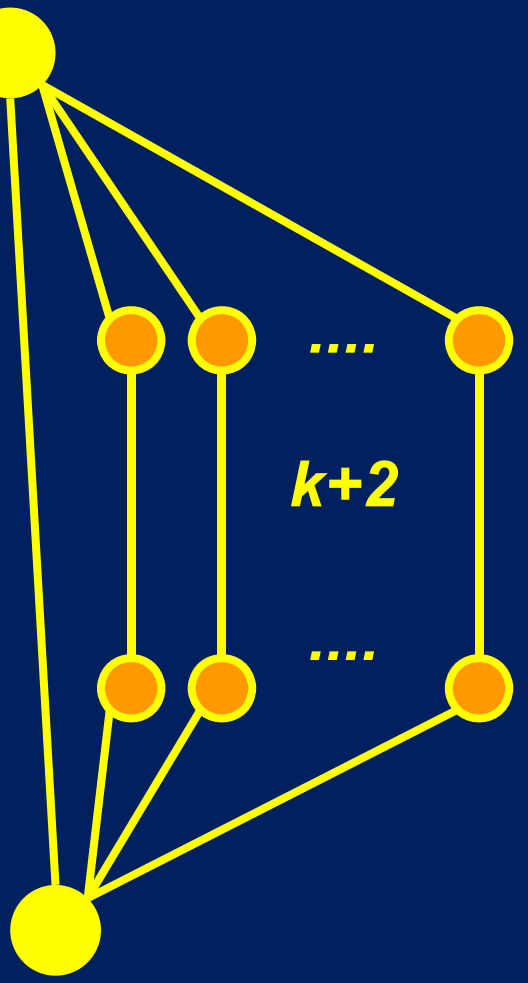


Observation 2

There are RAC graphs that are not 1-planar.

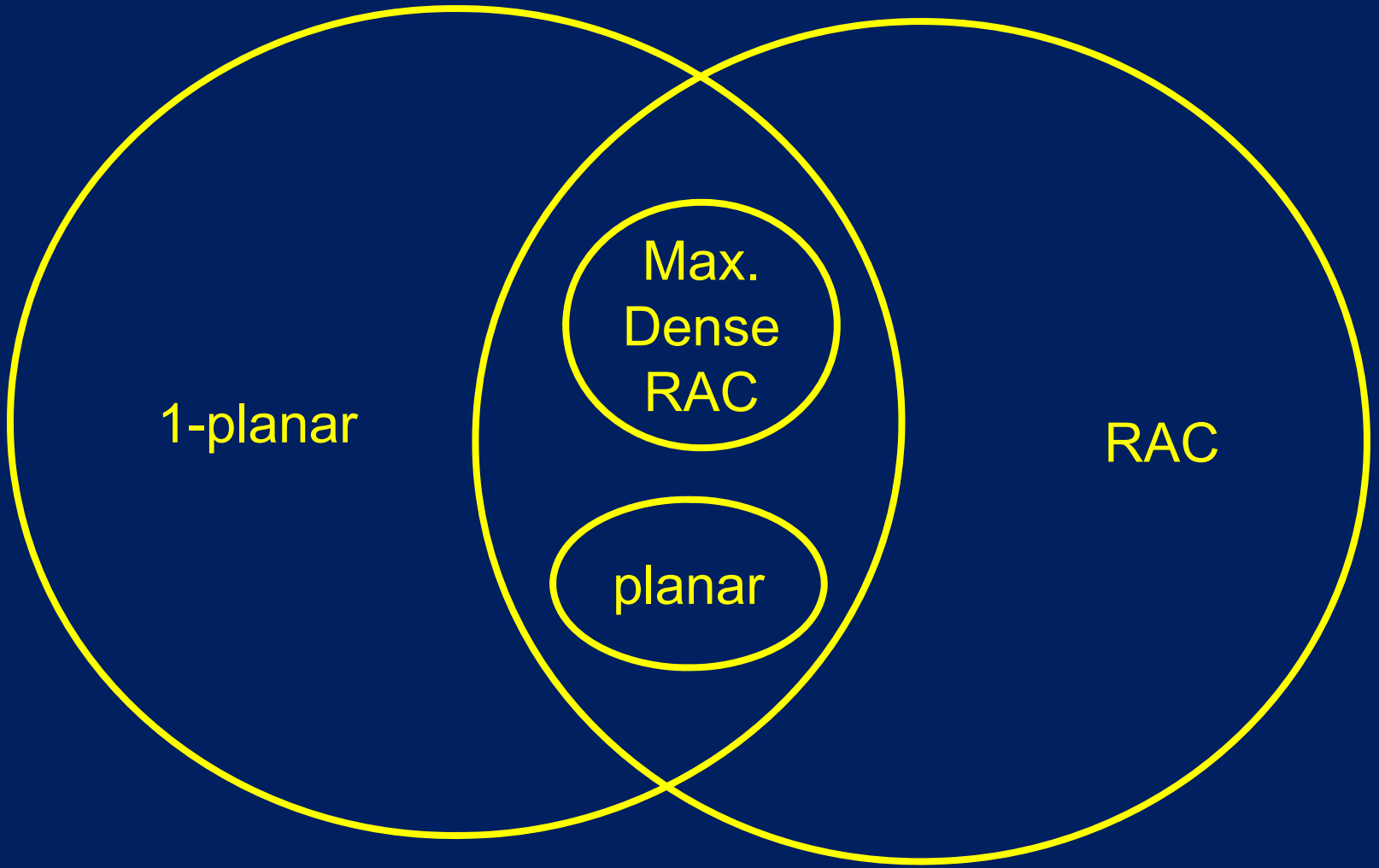


Replace each edge in K_5 with:



Theorem (Liotta, Eades)

If ***G*** is RAC and has ***4n-10*** edges, then ***G*** is 1-planar.



Final remarks

Other classes of *slightly-non-planar* graphs

- *2-planar*: each edge has at most 2 crossings
- *Quasi-planar*: no three edges mutually cross each other
- *Fan-crossing-free*: no pair of incident edges cross another edge
- *Twist-crossing-free*: no path of length 3 crosses itself
- *1-skew*: deletion of one edge makes it planar

Open problems for *curious* people

Class	***Maximum density***	***Computing topology***	***Computing straight-line drawing***
Planar	3n-6	Linear time	Linear time
1-planar	4n-8	NP-hard	Linear time*
Fan-crossing-free	?	?	?
Twist-crossing-free	?	?	?
Quasi-planar	6.5n-20	?	?
Right-angle-crossing (RAC)	4n-10	NP-hard	?
1-skew	3n-5	Quadratic(?)	Linear time*
...	...	...	...

Wild conjecture

Suppose that ***H*** is a topological graph, other than a pair of crossing edges.

Then the following problem is NP-hard.

*Instance: A graph **G***

*Question: Does **G** have a topological embedding in which **H** is not a subgraph?*

Open problems for *practical* people with some mathematical skills

We say that a graph drawing ***D*** has *crossing resolution* ***Θ(D)*** if each crossing angle is at least ***Θ.***

1. Investigate graph drawings ***D*** with ***Θ(D) ≥ π/3.***

2. Consider the following problem:

   Maximum Crossing Resolution
   Input: a graph G
   Output: a drawing D of G with maximum crossing resolution ***Θ(D).***

   Isolate the continuous and discrete parts of this optimisation problem.

Open problems for people with some HCI/Psych skills

Investigate the perceptual and mathematical relationships between crossing resolution and the number of crossings.

Open problems for practical people with lots of mathematical skills:

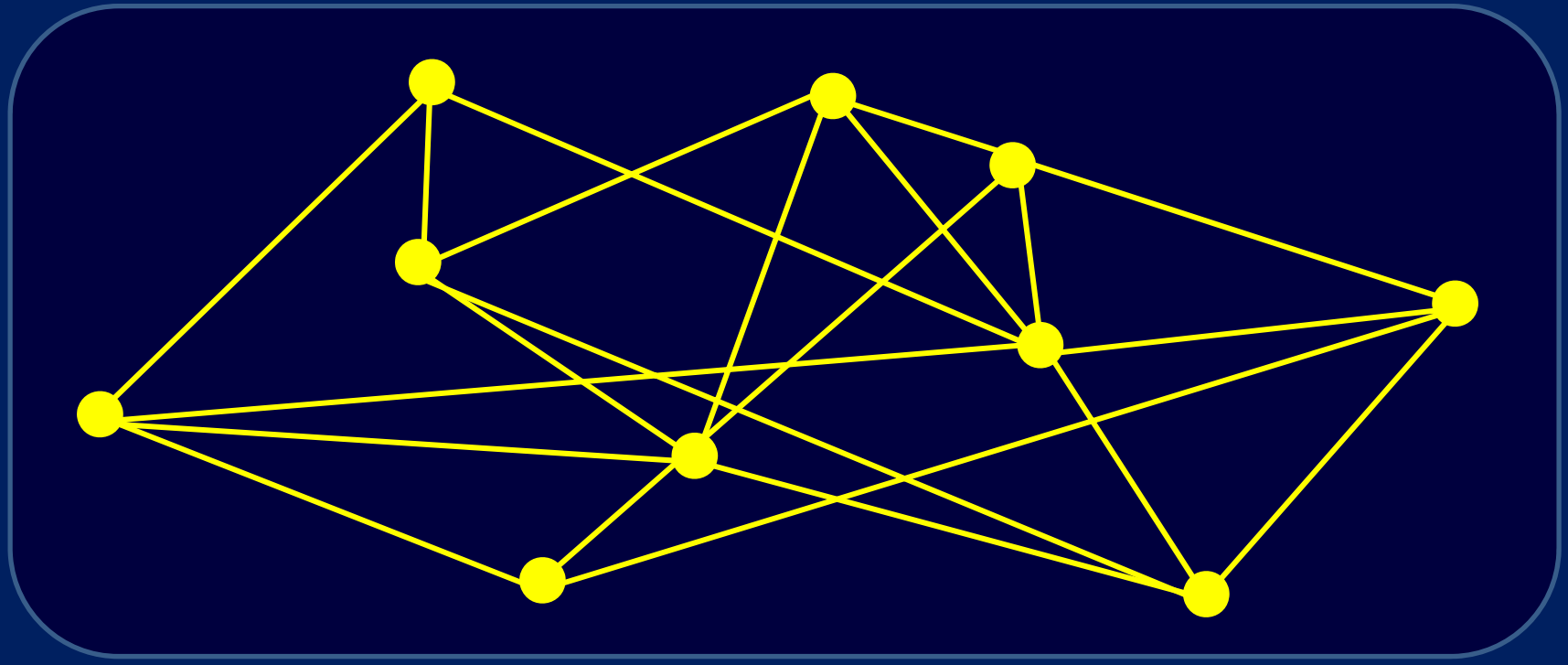
Randomisation

- In a “random” graph drawing with “short” edges:
  - How many edge crossings?
  - What is the crossing resolution?
  - What is the vertex resolution?
  - How do these parameters vary with graph theoretic parameters, such as diameter?

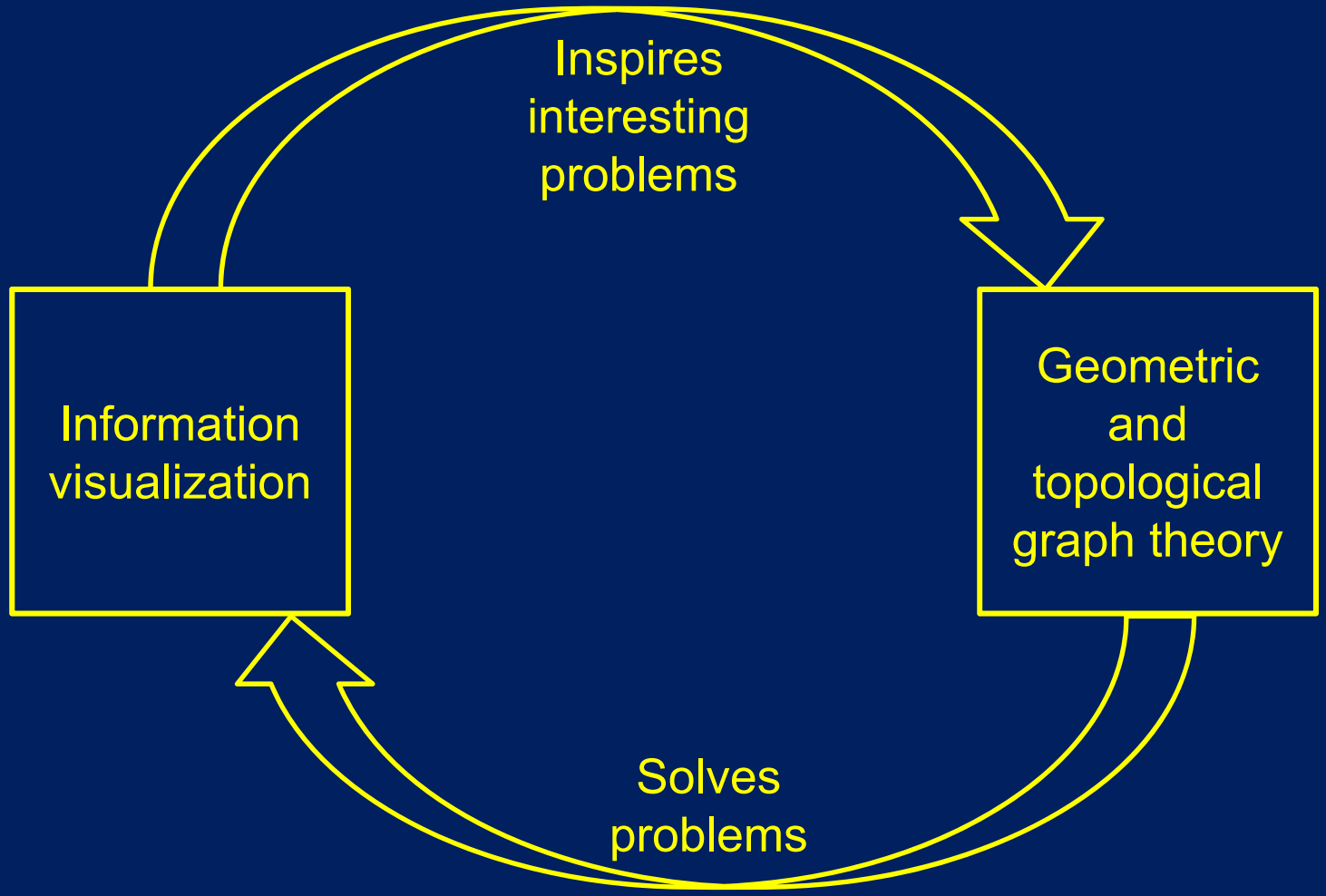
- Implications for graph drawing algorithms
  - Can we design algorithms that exploit randomization?
  - Can we provide stochastic guarantees of performance?

Open problem for computational geometry people:-

- Given a straight-line graph drawing, what is the smallest crossing angle?

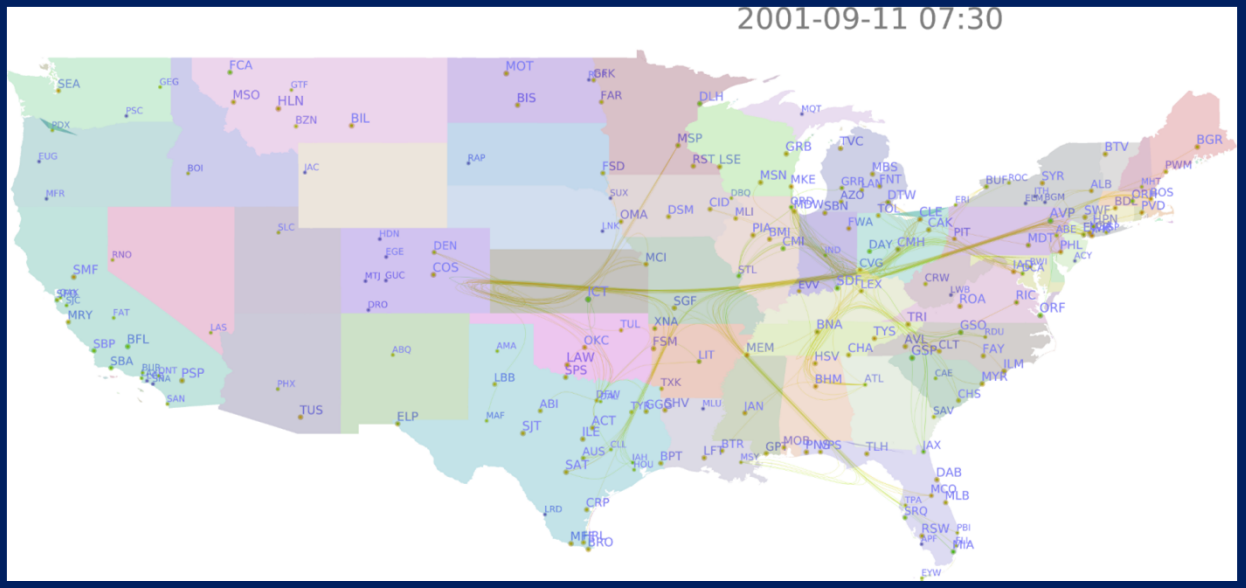


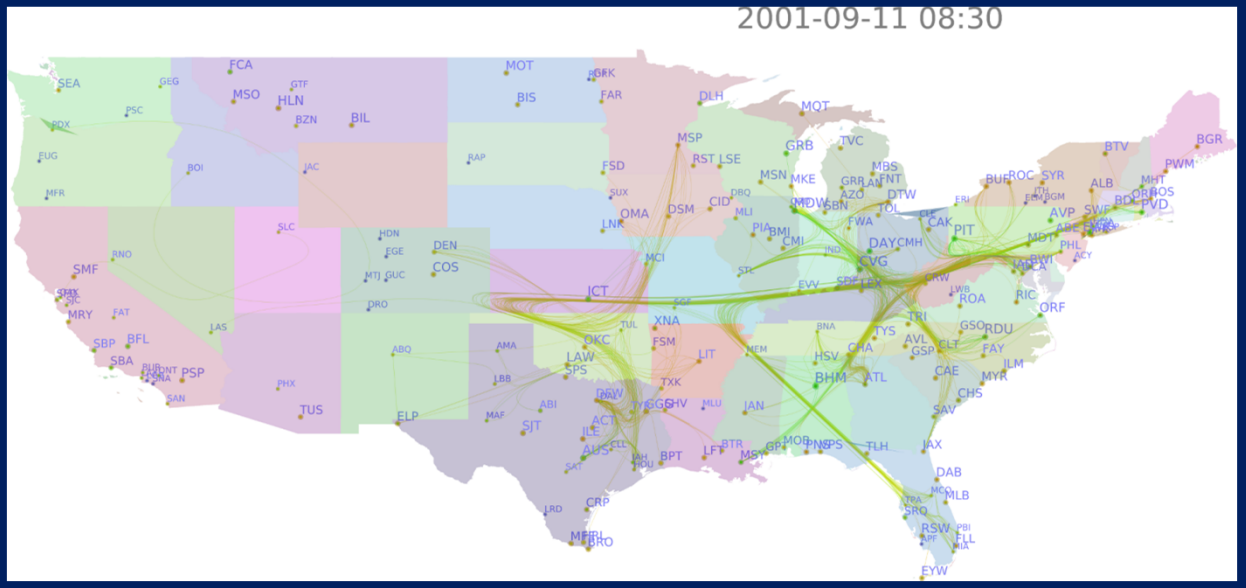
General open problems

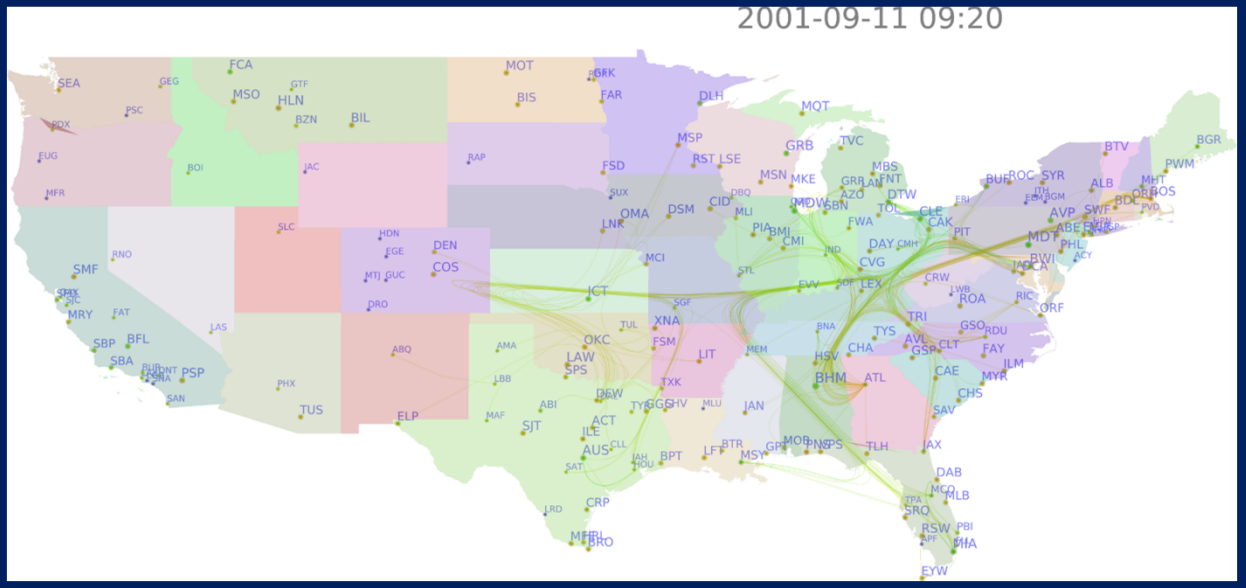


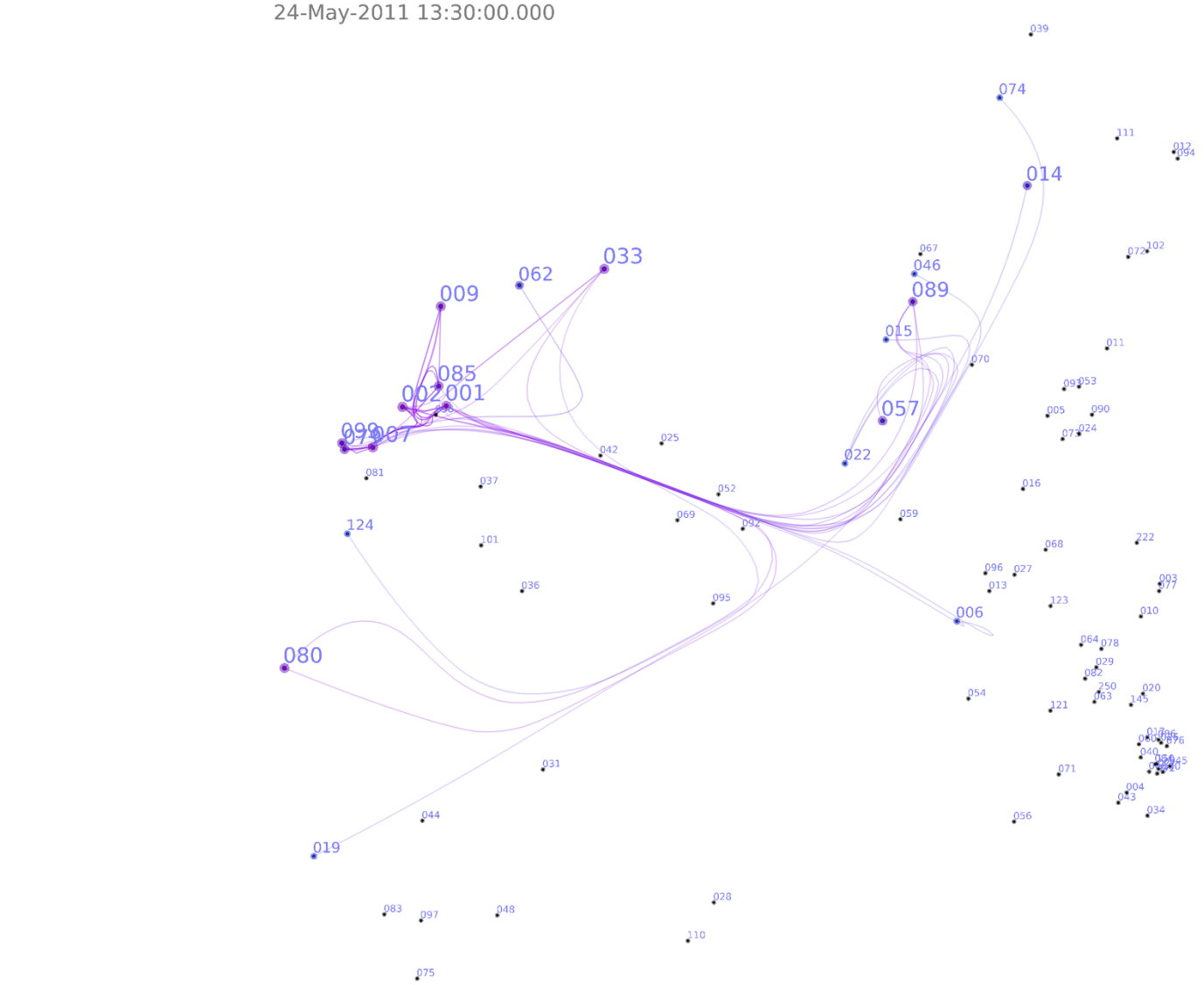
Play video …

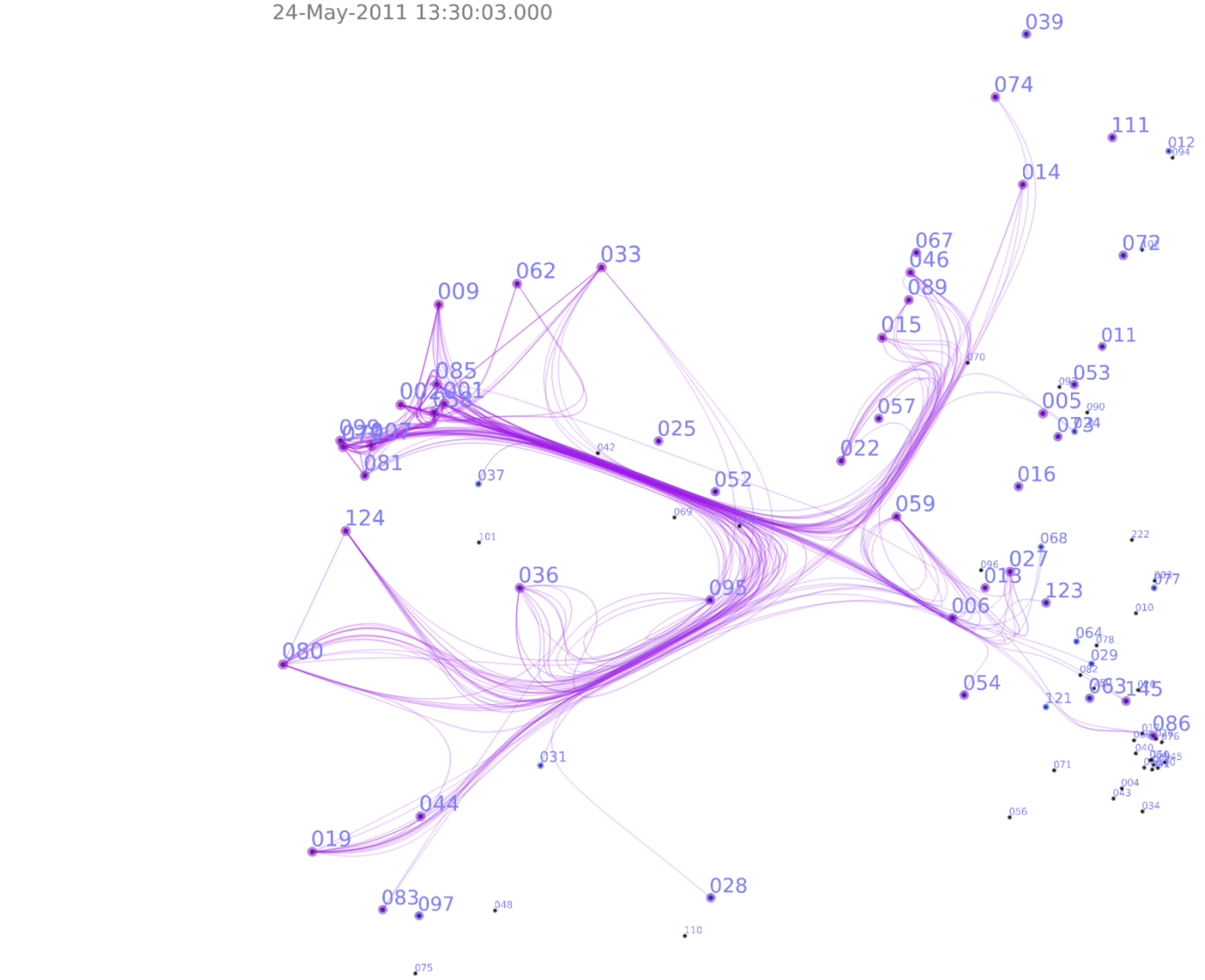
Quan Nguyen: edge bundling problems




2001-09-11 08:30
SEA
GEG
FCA
MSO
HLN
GTF
BZN
BIL
MOT
BIS
FAR
DLH
PSC
PDX
EUG
BOI
JAC
RAP
FSD
MSP
RST
LSE
GRB
TVC
BTV
BGR
PWM
MFR
SUX
MSN
MKE
GRR
AZO
DTW
MBS
FNT
BUF
ROC
SYR
ALB
MHT
BOS
PVD
SLC
HDN
EGE
DEN
COS
MTJ
GUC
DRO
OMA
DSM
CID
DBQ
MLI
LNK
PIA
BMI
CMI
FWA
CAK
PIT
AVP
SWF
ABE
PHL
ACY
MDT
RNO
SMF
MCI
IND
DAY
CMH
CVG
STL
EVV
SPI
LWB
ROA
RIC
ORF
ICT
FOE
TUL
XNA
MRY
FAT
LAS
SBP
BFL
SBA
PSP
OKC
FSM
LIT
HEM
BNA
TYS
GSO
RDU
AVL
CLT
GSP
CHA
FAY
ILM
HSV
CAE
MYR
CHS
ATL
BHM
TXK
LAW
SPS
LBB
ABQ
AMA
PHX
TUS
ELP
MAF
ABI
SJT
ACT
ILE
AUS
CLL
HOU
SAT
BPT
LFT
BTR
MSY
GPT
MOB
PNS
TLH
JAX
SAV
DAB
MCO
MLB
TPA
SRQ
RSW
PBI
FLL
MIA
EYW
CRP
LRD
MFE
BRO


2001-09-11 09:20
SEA
GEG
FCA
MSO
HLN
BZN
BIL
MOT
BIS
FAR
DLH
MQT
MSP
GRB
TVC
BTV
BGR
PWM
RST
LSE
FSD
RAP
MSN
MKE
MBS
FNT
BUF
ROC
SYR
ALB
MHT
BOS
PDX
PSC
EUG
BOI
IAC
MFR
SUX
DBQ
CID
OMA
DSM
MLI
LNK
PIA
BMI
CMI
FWA
AZO
DTW
GRR
CLE
CAK
AVP
SWF
ABE
PIT
MDT
PHL
BWI
DAY
CMH
IND
CVG
STL
MCI
EVV
LEX
JOE
CRW
ROA
RIC
ORF
TRI
GSO
RDU
TYS
AVL
CHA
GSP
CLT
FAY
ILM
HSV
BNA
MEM
LIT
XNA
TUL
OKC
FSM
LAW
SPS
TXK
DFW
ABI
TYR
GGG
HRV
MLU
JAN
ACT
ELP
MAF
SJT
ILE
AUS
CLL
IAH
HOU
BPT
LFT
BTR
MSY
GPT
MOB
PNS
TLH
JAX
BHM
ATL
MYR
CAE
CHS
SAV
DAB
MCO
MLB
TPA
SRO
RSW
PBI
MIA
EYW
CRP
LRD
BRO
SAT
SGF
ICT
DEN
COS
HDN
EGE
MTJ
GUC
DRO
SLC
RNO
SMF
SCK
MRY
FAT
LAS
SBP
BFL
SBA
PSP
PHX
TUS
ABQ
AMA
LBB
SAN
BUR
ONT
LGB
SNA

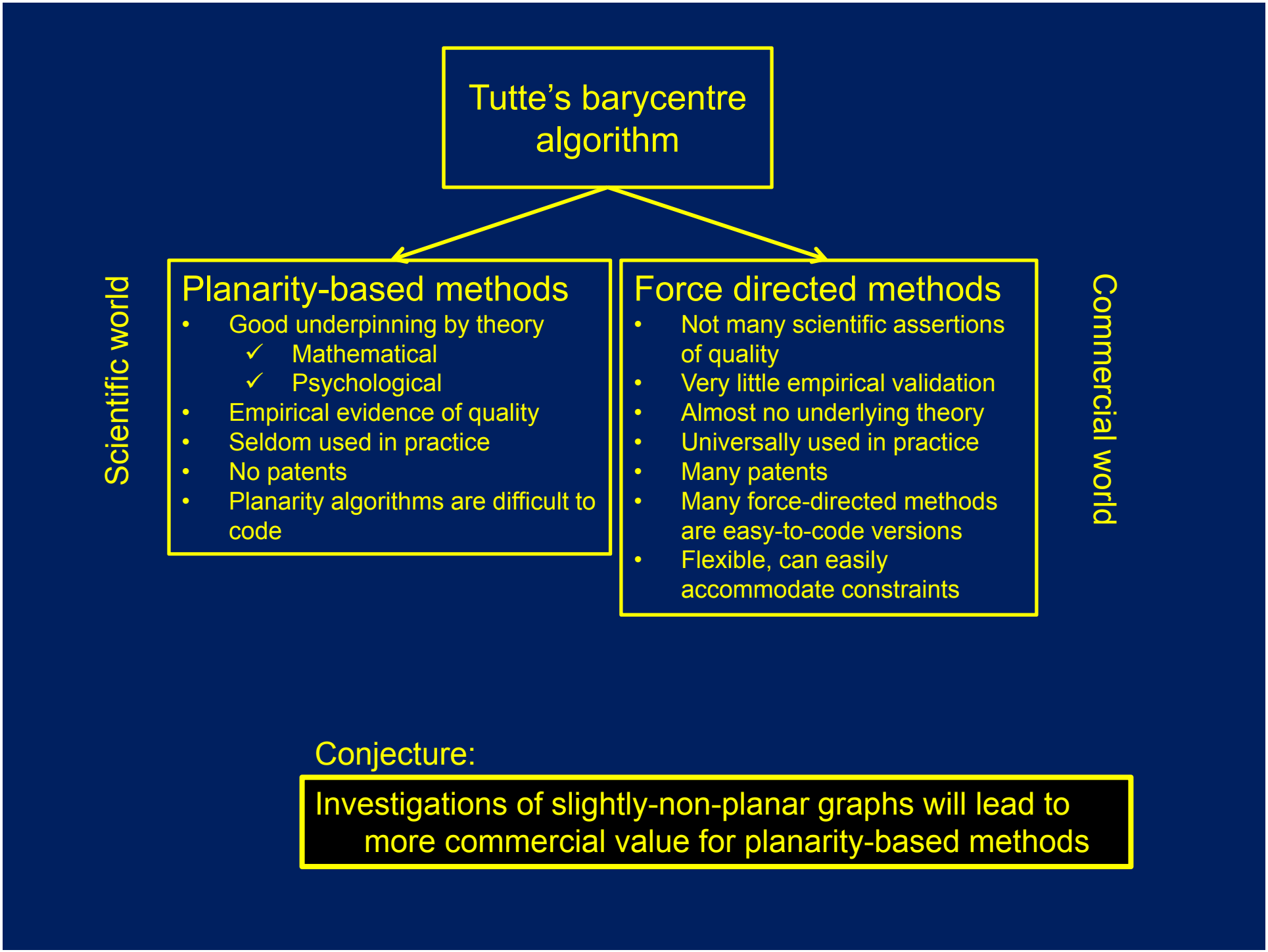

24-May-2011 13:30:00.000
039
074
111
012
094
014
033
062
067
046
072
102
009
089
015
011
085
070
002
001
057
005
090
099
007
025
042
022
081
037
052
016
124
069
092
059
101
068
222
096
027
003
036
013
077
095
123
006
010
064
078
080
029
082
250
020
054
063
145
121
031
071
004
043
044
056
034
019
028
083
097
048
110
075


24-May-2011 13:30:03.000
039
074
111
012
094
014
067
046
089
033
062
009
015
011
070
053
085
005
090
057
025
022
042
081
037
052
016
059
124
069
101
068
222
027
036
095
123
006
064
078
029
080
054
121
145
086
031
071
044
004
043
034
019
056
028
083
097
048
110
075

Problems:

- What mathematically-sound measures of edge bundling pictures define good visualizations?
- What algorithms can be used to optimise these measures?
- What are the mathematical limits on geometric graphs with as a function of these measures?

Final final conjecture ….


Tutte's barycentre algorithm
Scientific world
Planarity-based methods
• Good underpinning by theory
✓ Mathematical
✓ Psychological
• Empirical evidence of quality
• Seldom used in practice
• No patents
• Planarity algorithms are difficult to code
Force directed methods
• Not many scientific assertions of quality
• Very little empirical validation
• Almost no underlying theory
• Universally used in practice
• Many patents
• Many force-directed methods are easy-to-code versions
• Flexible, can easily accommodate constraints
Commercial world
Conjecture:
Investigations of slightly-non-planar graphs will lead to more commercial value for planarity-based methods