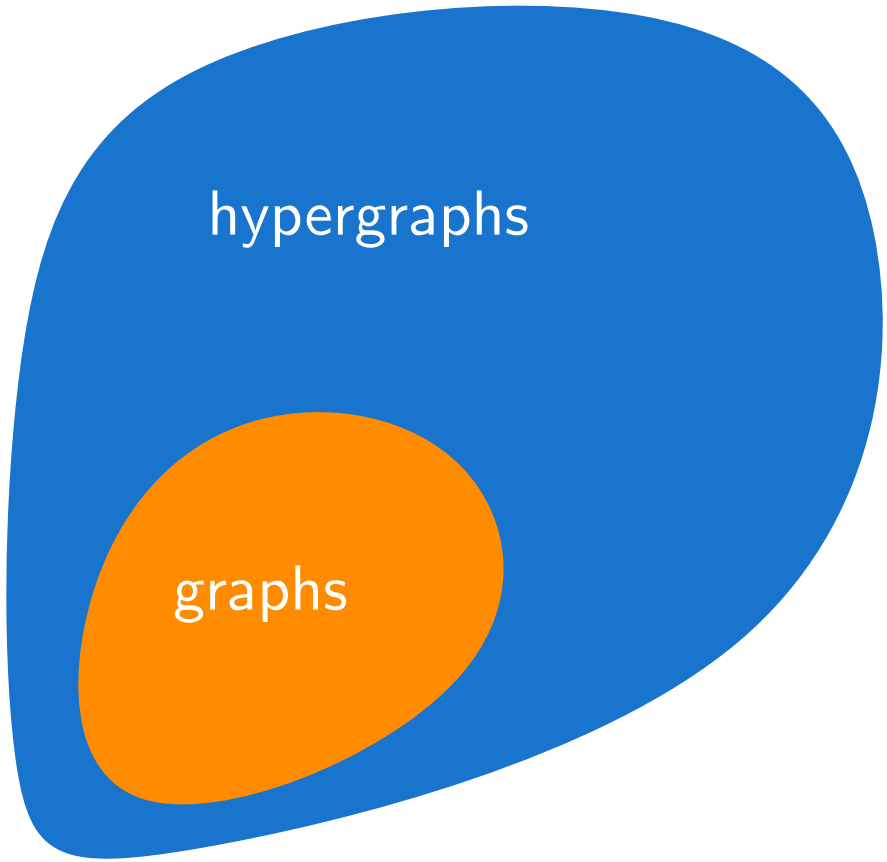
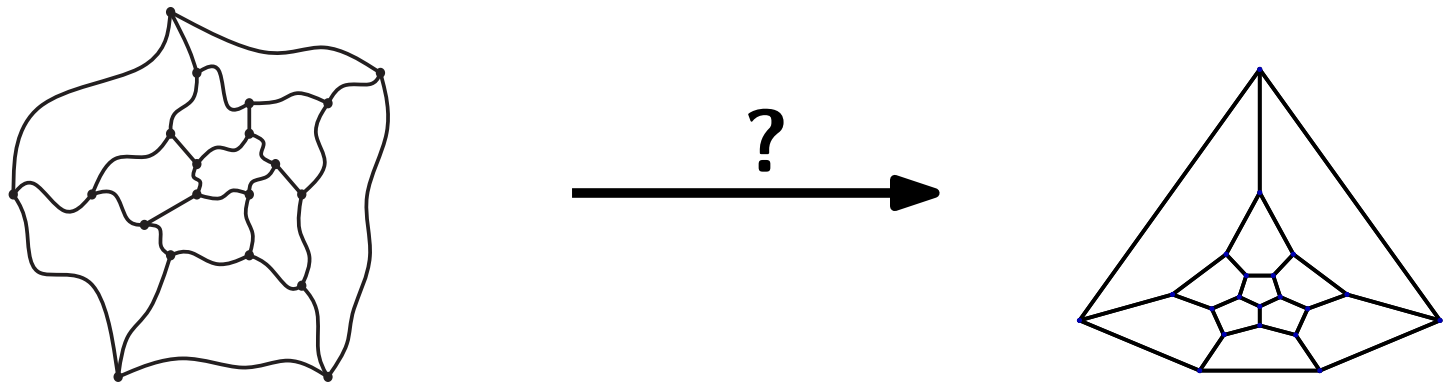


Survey on graph and hypergraph drawing

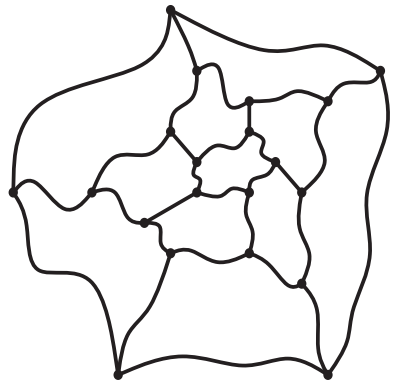
Dagstuhl Seminar on
Low-Dimensional Topology
André Schulz and Alexander Wolff
August 2019

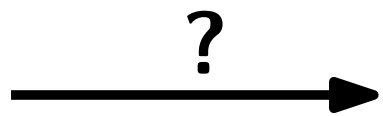
What is Graph Drawing?

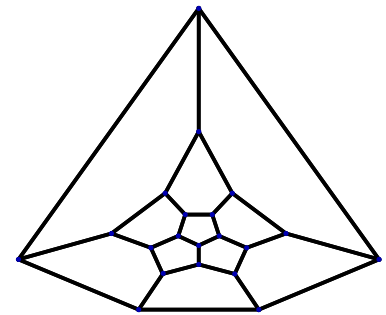
What is Graph Drawing?



What is Graph Drawing?



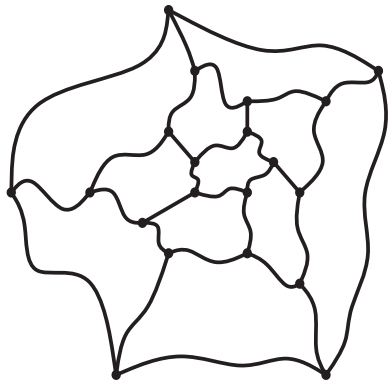




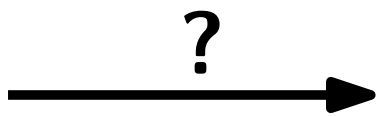
abstract (combinatorial) graph

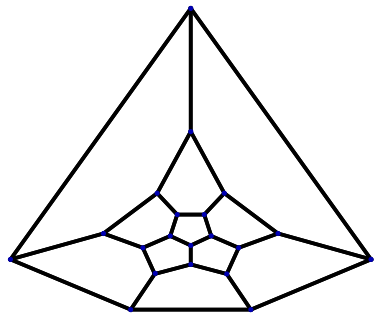
drawing (e.g. node-link diagram)

What is Graph Drawing?



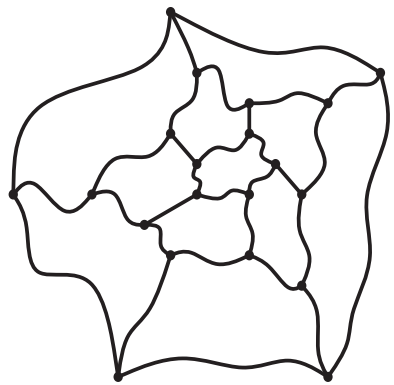
abstract (combinatorial)
graph



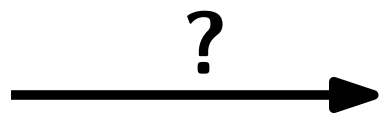


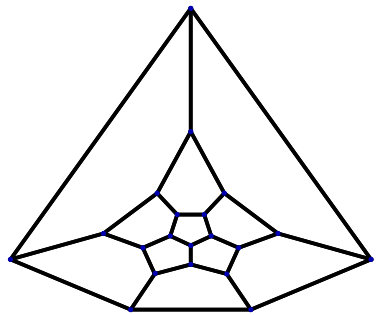
drawing
(e.g. node-link diagram)

What is Graph Drawing?



abstract (combinatorial) graph

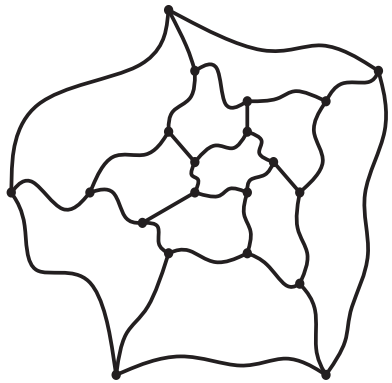




drawing (e.g. node-link diagram)

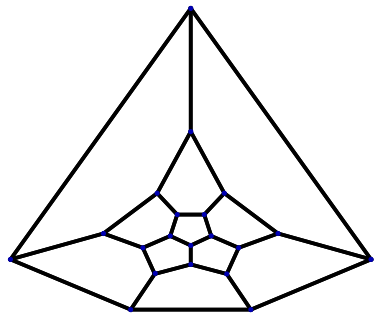
Goal: Algorithm guarantees a (provable) geometric quality measure in the worst case.

What is Graph Drawing?



?

ALGORITHM



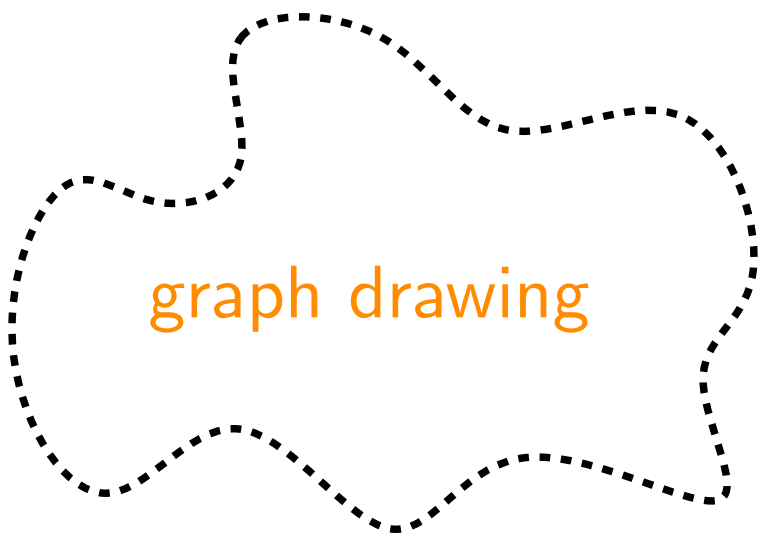
abstract (combinatorial) graph

drawing (e.g. node-link diagram)

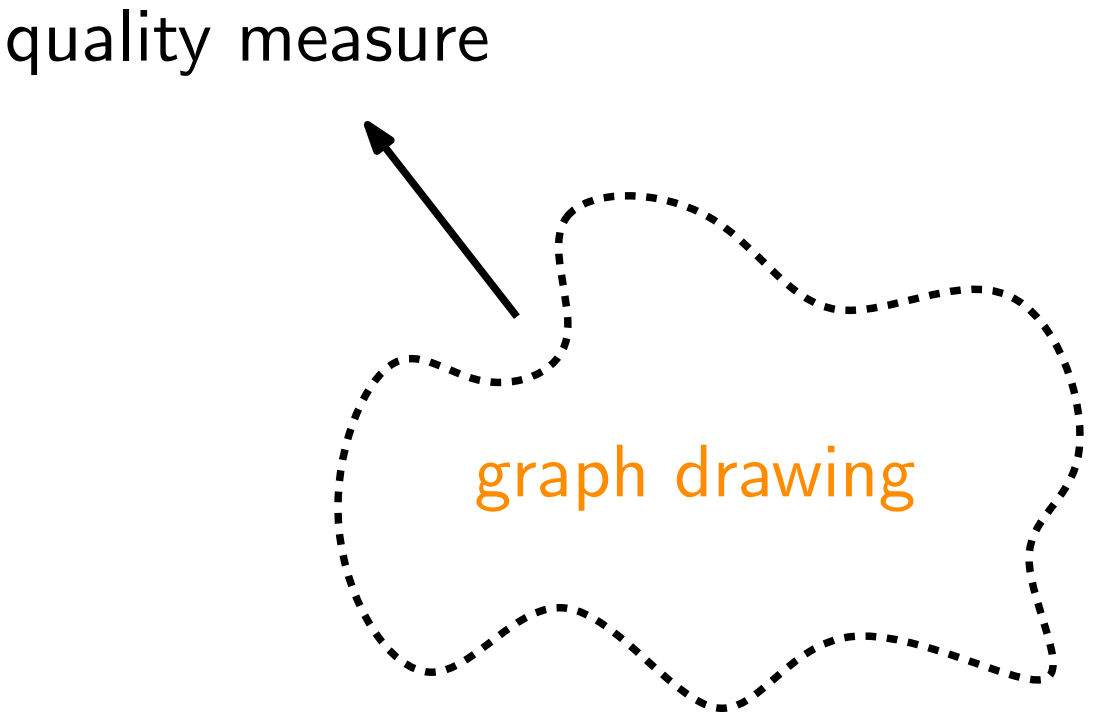
Goal: Algorithm guarantees a (provable) geometric quality measure in the worst case.

Evaluation is (usually) not task-driven.

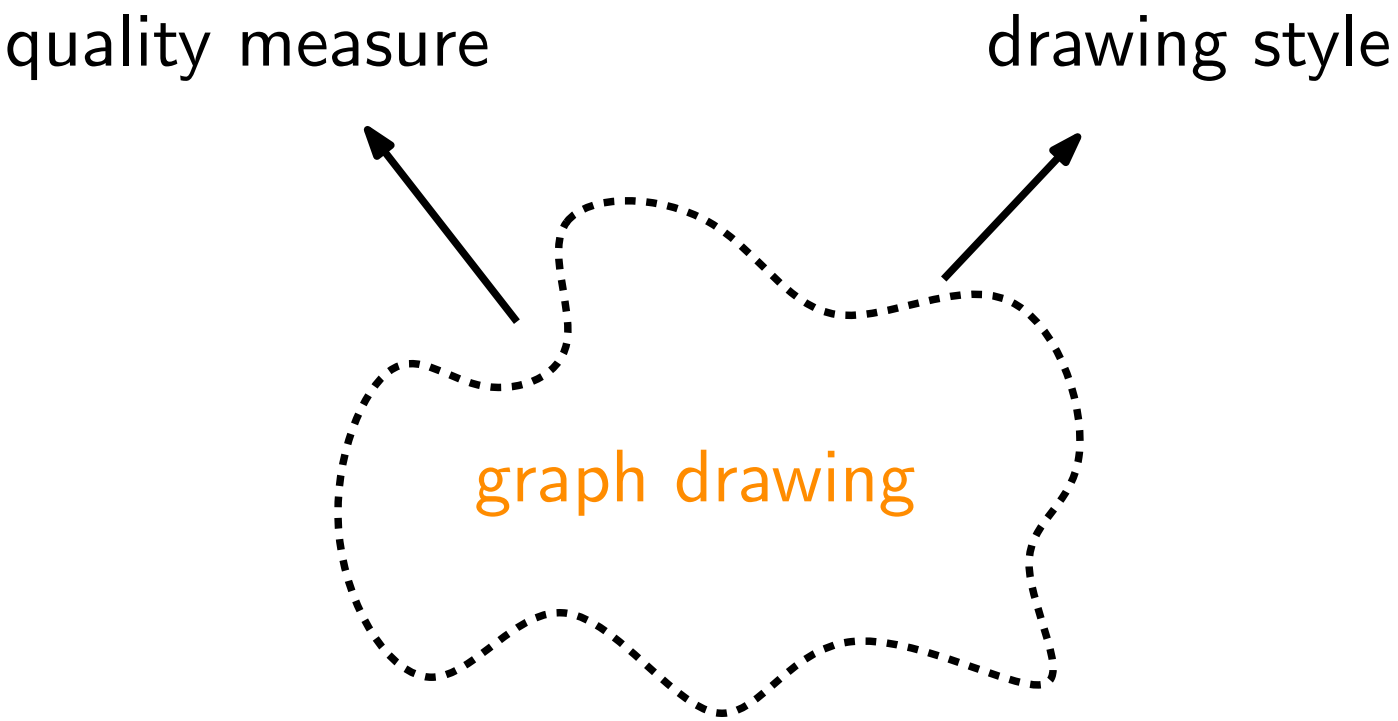
The Many Dimensions of Graph Drawing



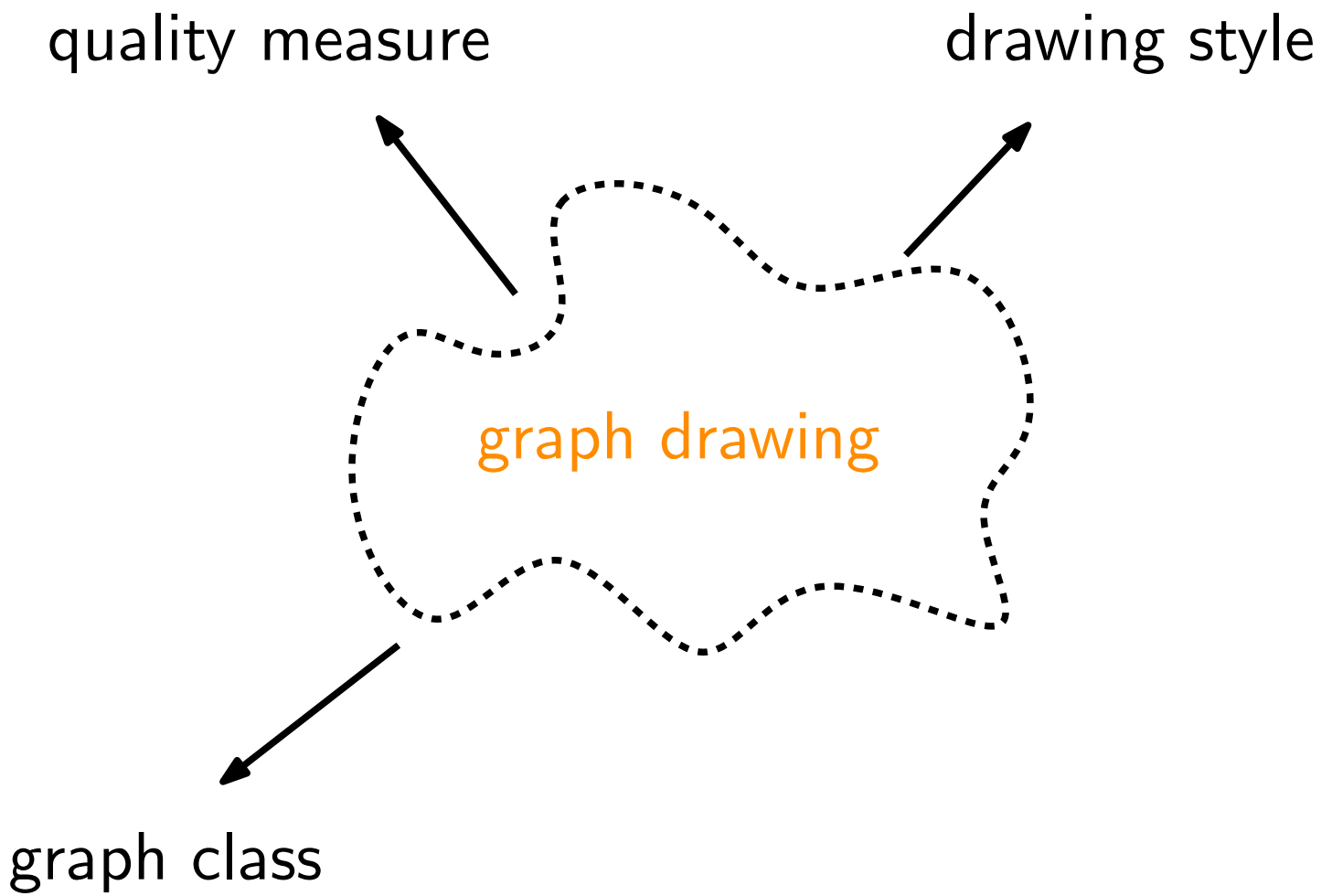
The Many Dimensions of Graph Drawing



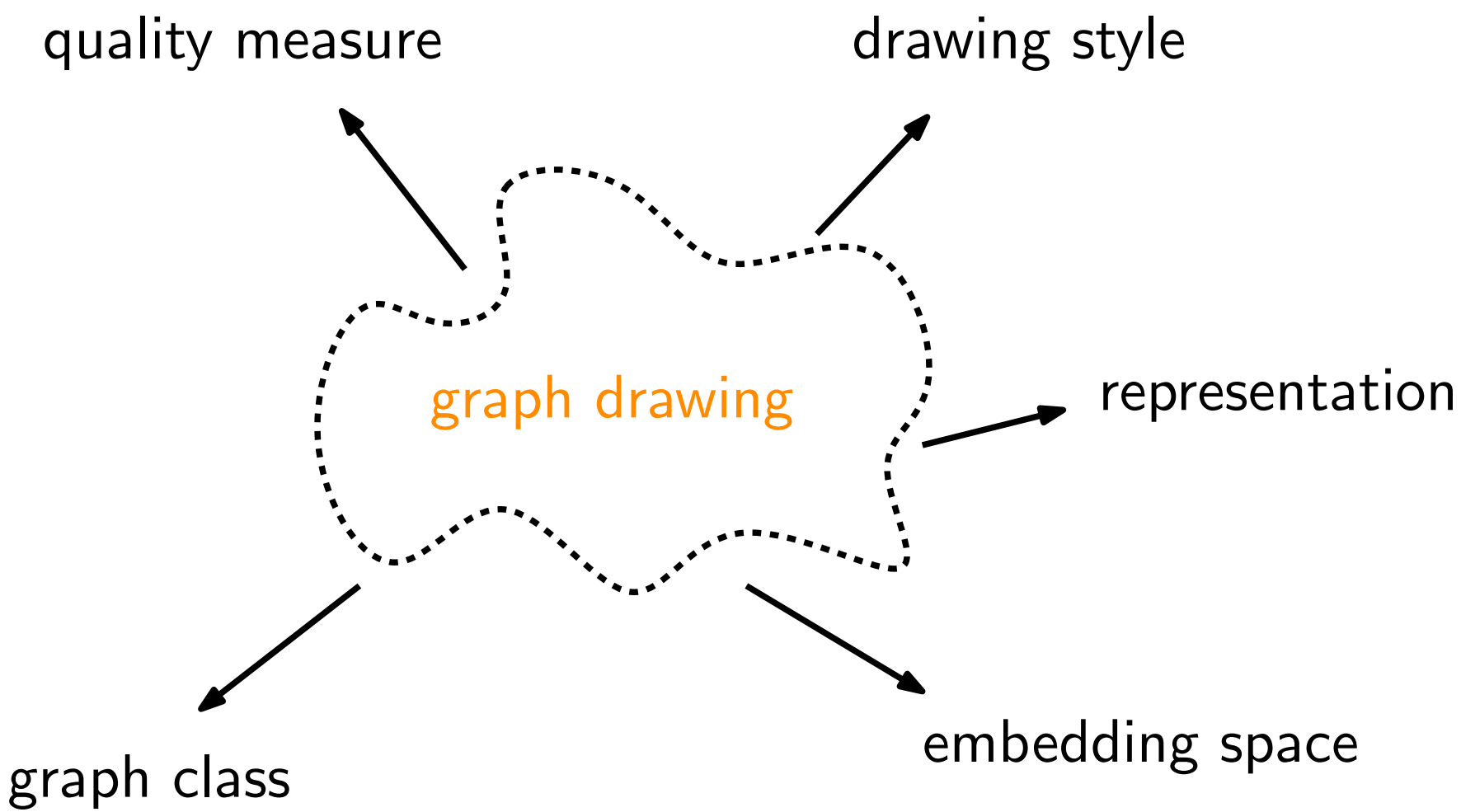
The Many Dimensions of Graph Drawing



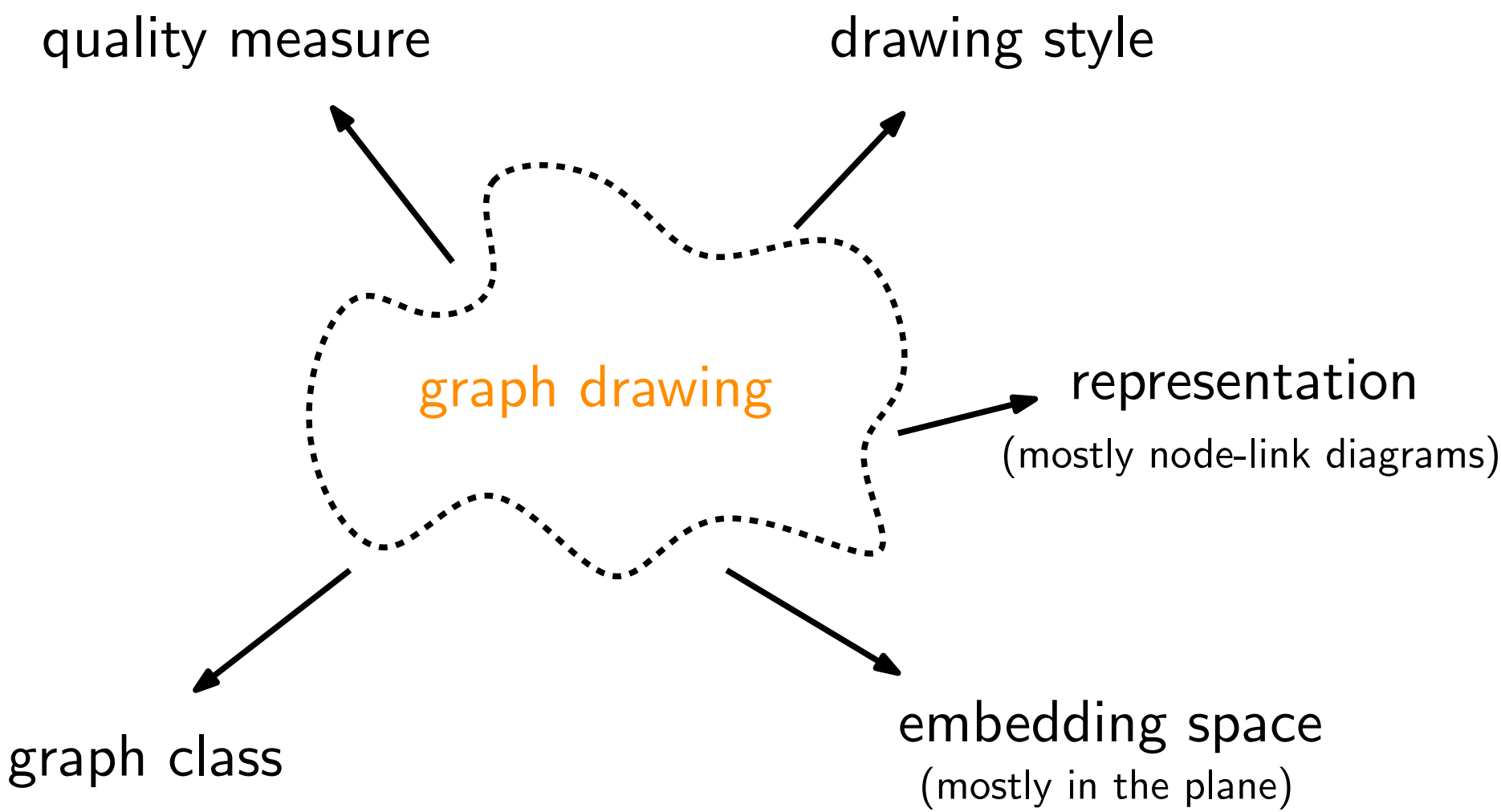
The Many Dimensions of Graph Drawing



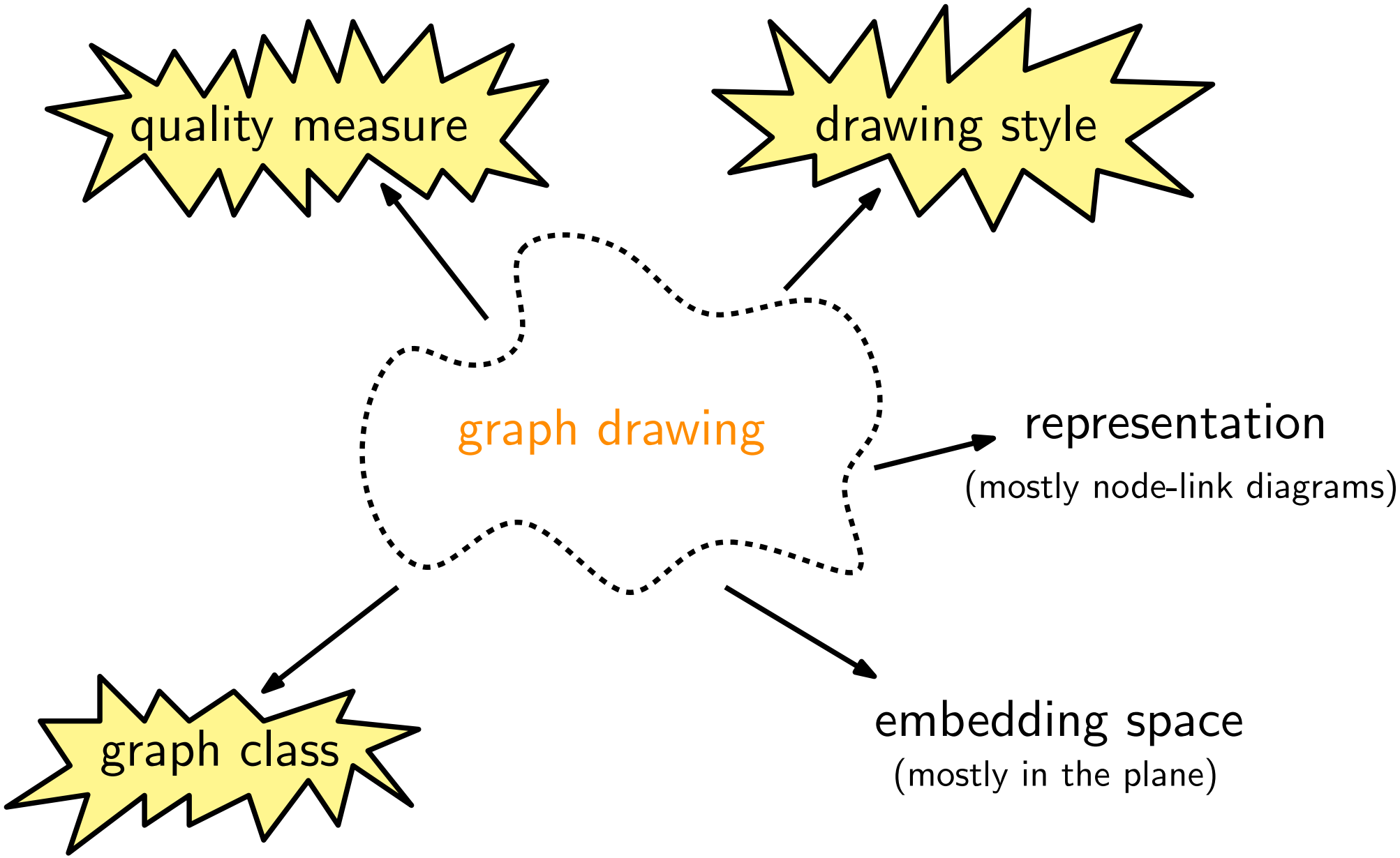
The Many Dimensions of Graph Drawing



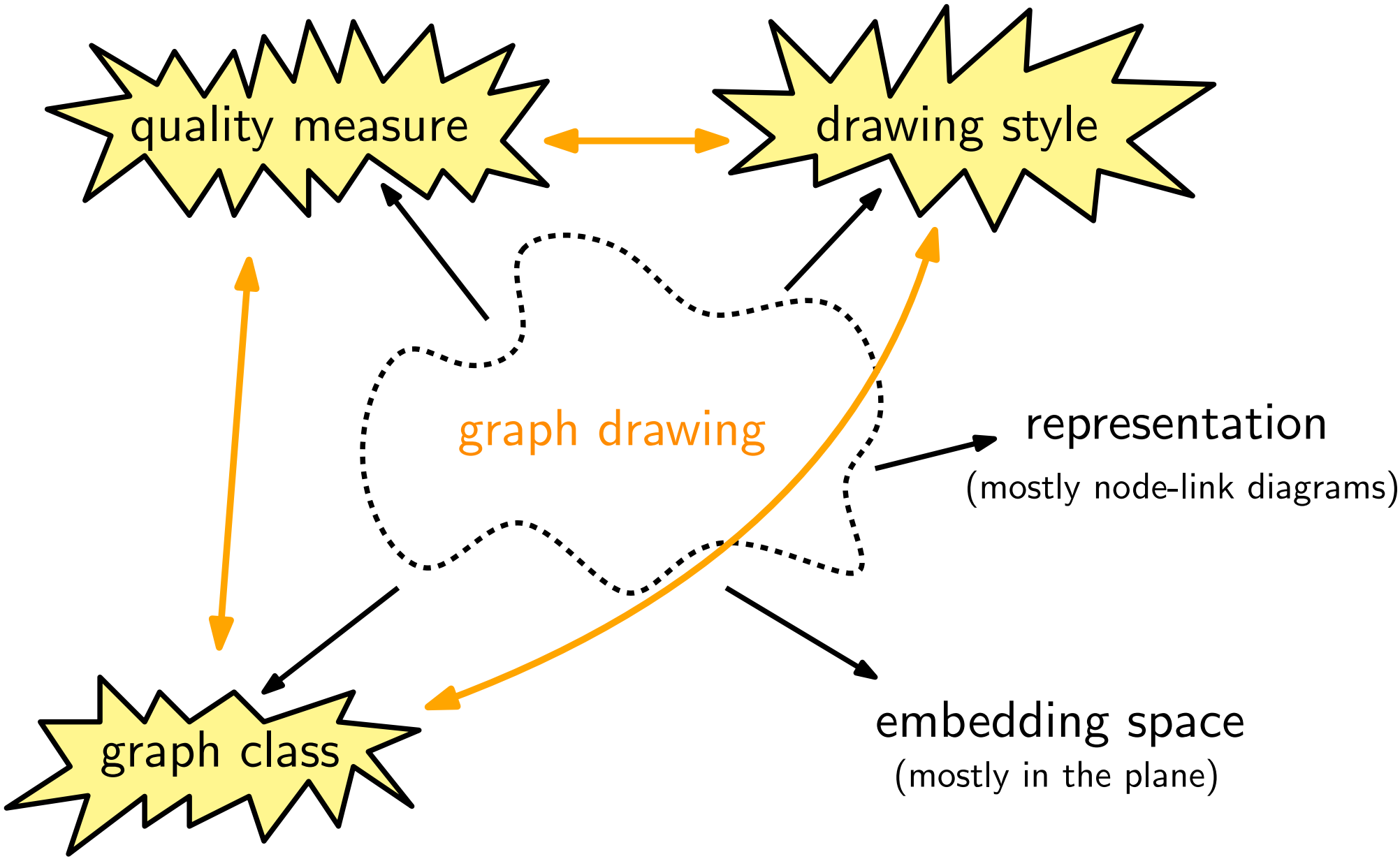
The Many Dimensions of Graph Drawing



The Many Dimensions of Graph Drawing

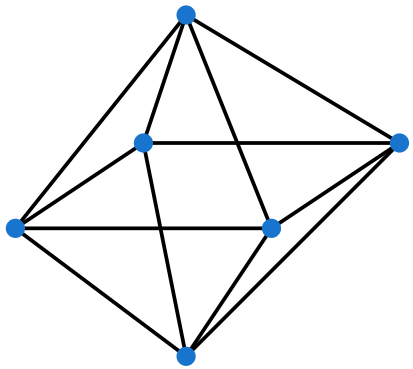


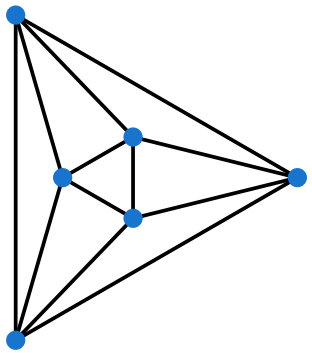
The Many Dimensions of Graph Drawing

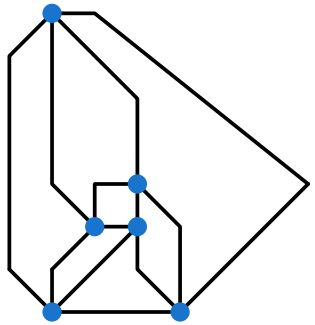


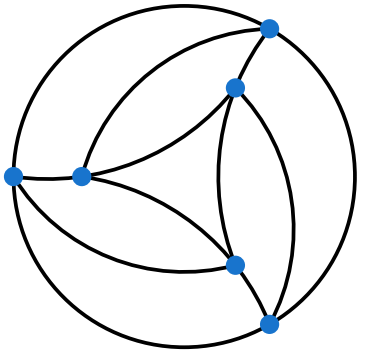
Drawing Styles

Drawing Styles

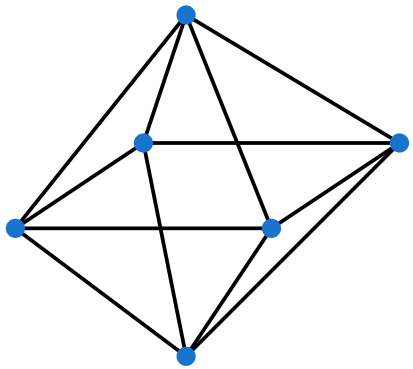
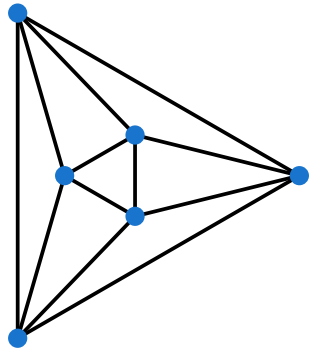
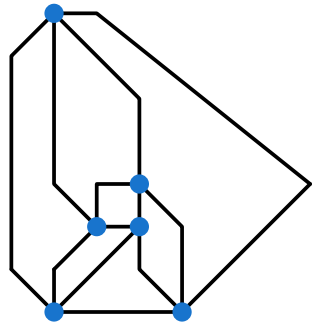
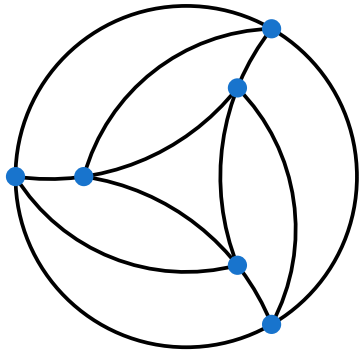






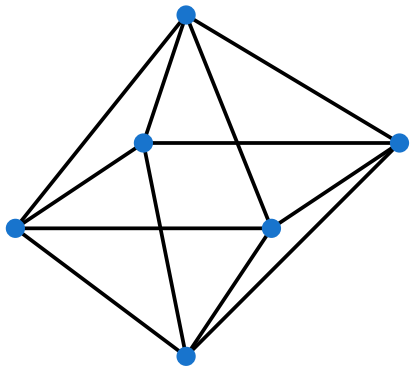
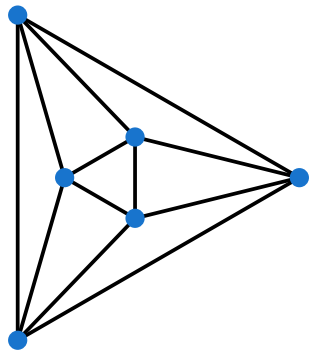
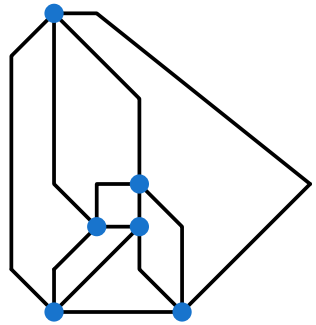
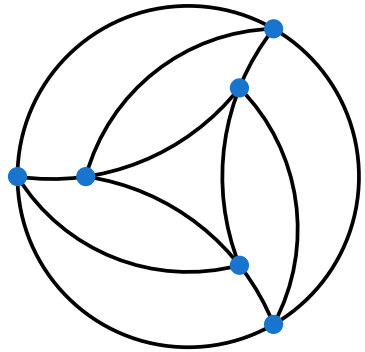


Drawing Styles

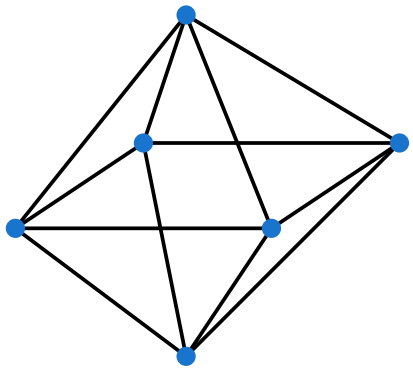
- straight-line vs. curved

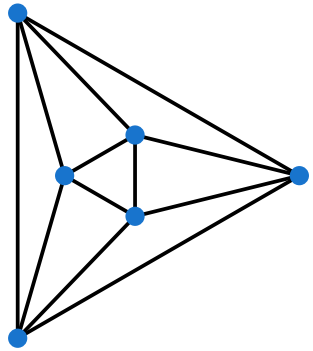
Drawing Styles

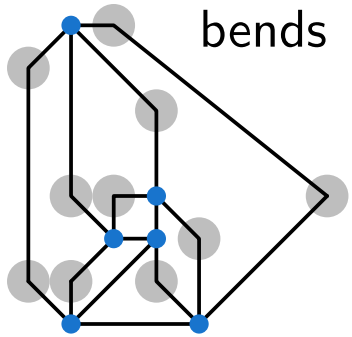





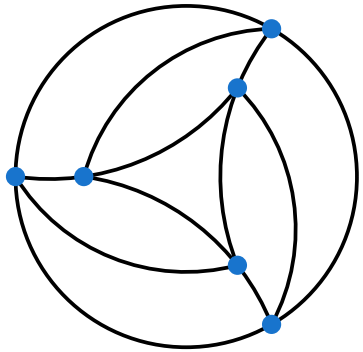
- straight-line vs. curved
- straight-line vs. polyline

Drawing Styles



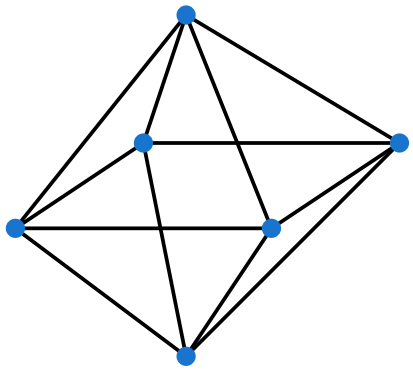
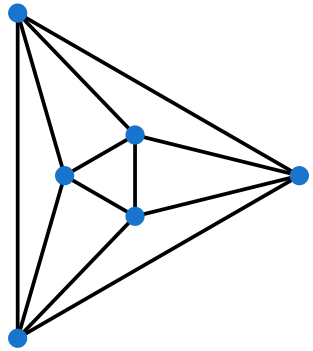
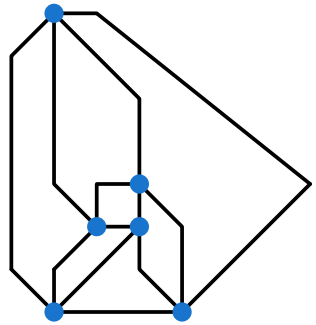
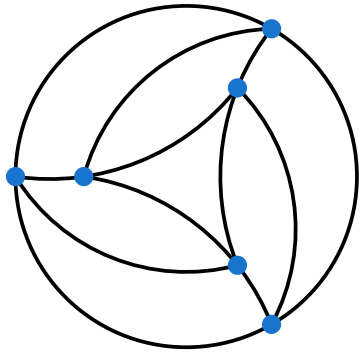






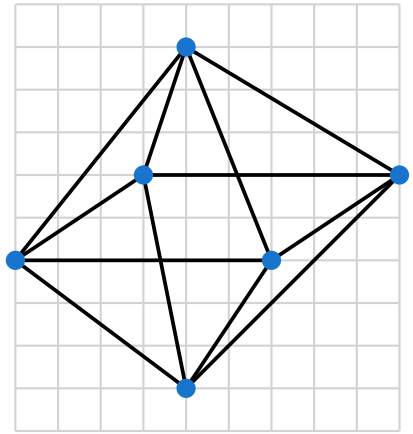
- straight-line vs. curved
- straight-line vs. polyline

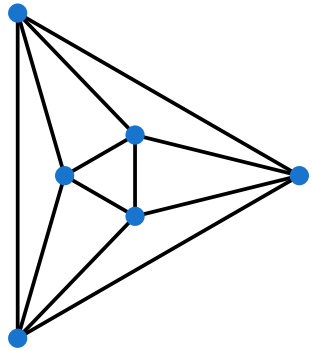
Drawing Styles

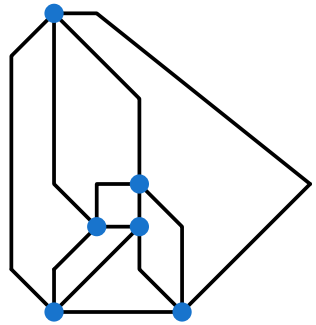





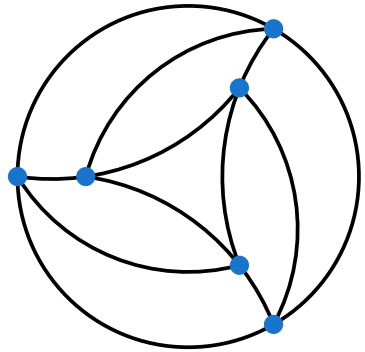
- straight-line vs. curved
- straight-line vs. polyline
- restricted slopes

Drawing Styles



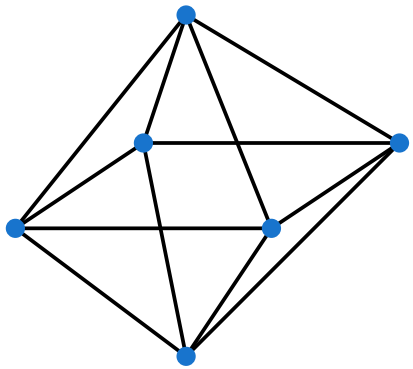
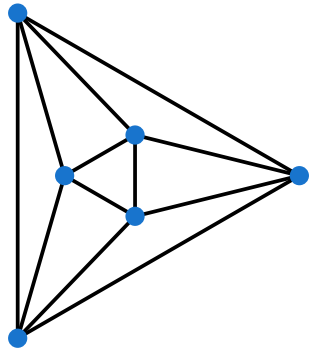
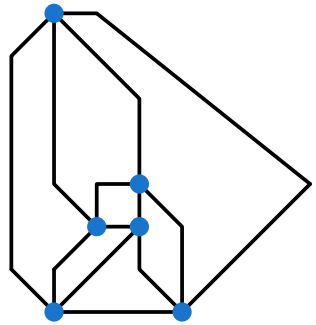
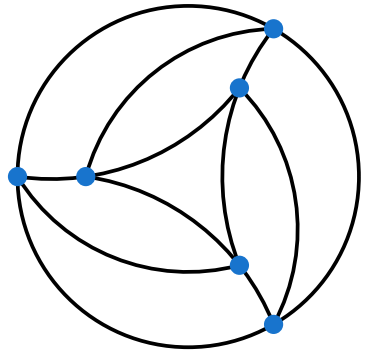






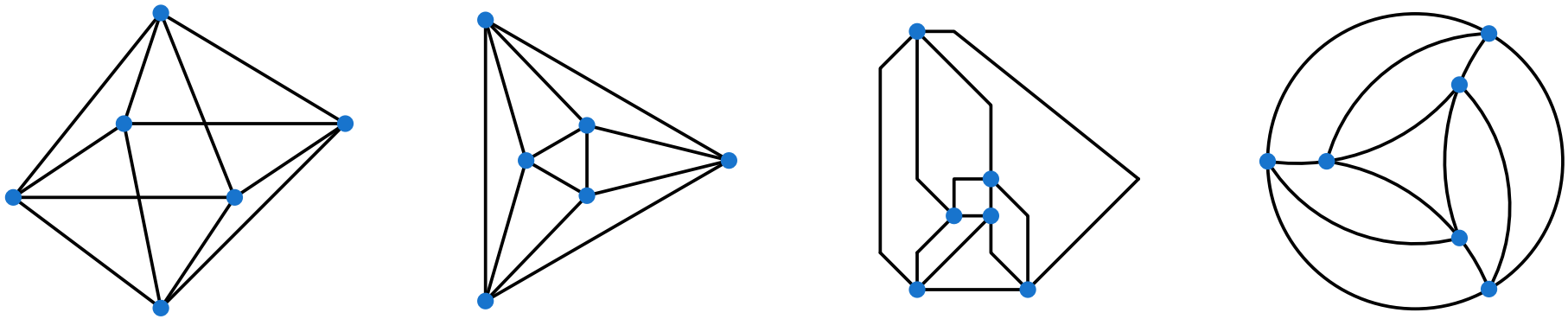
- straight-line vs. curved
- straight-line vs. polyline
- restricted slopes
- restricted to grid points

Drawing Styles

- straight-line vs. curved
- straight-line vs. polyline
- restricted slopes
- restricted to grid points
- directed drawings

Drawing Styles



- straight-line vs. curved
- straight-line vs. polyline
- restricted slopes
- restricted to grid points
- directed drawings
- monotone drawings, confluent drawings, partial edge drawing, radial drawings, thick drawings, Lombardi drawings, ...

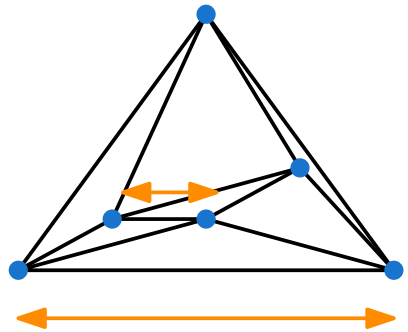
Classical Measures

Classical Measures

- vertex resolution

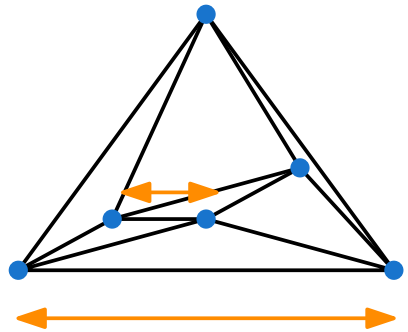
Classical Measures

- vertex resolution $= \dfrac{\text{maximal distance between two vertices}}{\text{minimal distance between two vertices}}$



Classical Measures

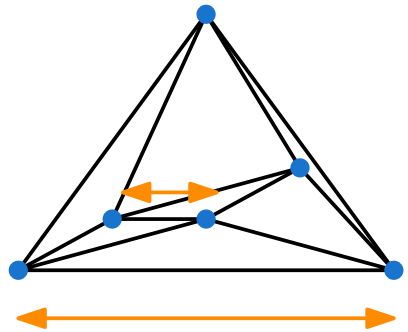
- vertex resolution $= \dfrac{\text{maximal distance between two vertices}}{\text{minimal distance between two vertices}}$



goal: small vertex resolution

Classical Measures

- vertex resolution $= \dfrac{\text{maximal distance between two vertices}}{\text{minimal distance between two vertices}}$

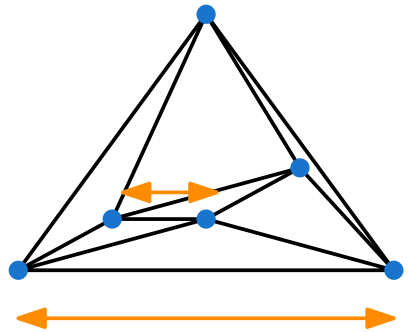


goal: small vertex resolution

- angular resolution

Classical Measures

- vertex resolution $= \dfrac{\text{maximal distance between two vertices}}{\text{minimal distance between two vertices}}$

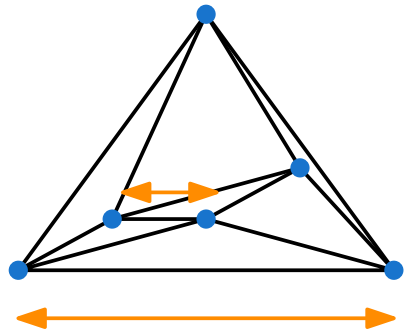


goal: small vertex resolution

- angular resolution = size of the smallest angle

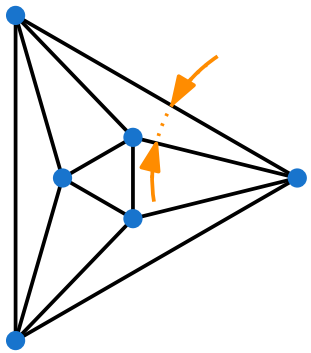
Classical Measures

- vertex resolution $= \dfrac{\text{maximal distance between two vertices}}{\text{minimal distance between two vertices}}$



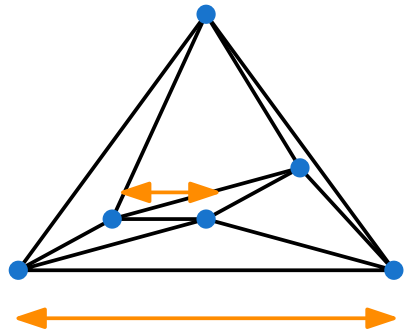
goal: small vertex resolution

- angular resolution = size of the smallest angle



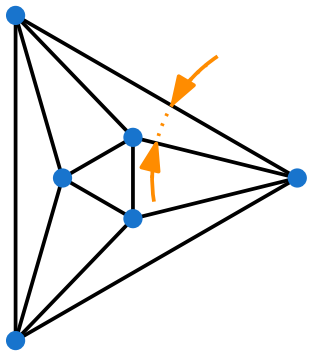
Classical Measures

- vertex resolution $= \dfrac{\text{maximal distance between two vertices}}{\text{minimal distance between two vertices}}$



goal: small vertex resolution

- angular resolution = size of the smallest angle



goal: large angular resolution

More Measures

More Measures

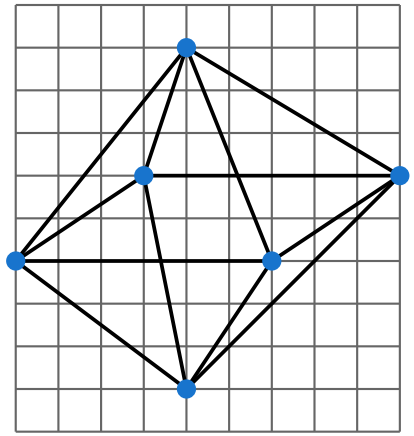
- grid size

More Measures

- grid size $=$ area of the drawing using integer grid points

More Measures

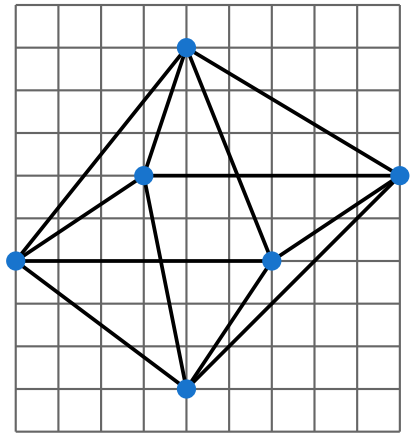
- grid size = area of the drawing using integer grid points



goal: small grid size

More Measures

- grid size = area of the drawing using integer grid points

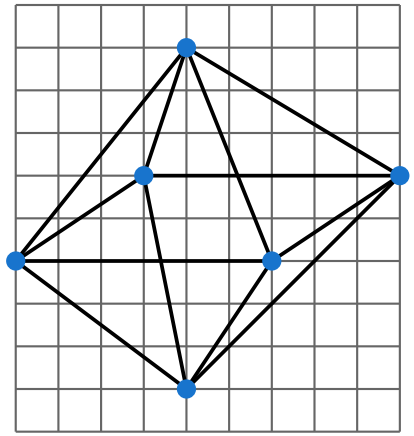


goal: small grid size

$\rightarrow$ implies good vertex and angular resolution

More Measures

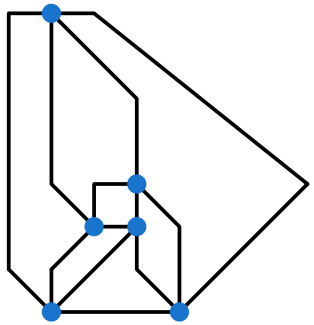
- grid size = area of the drawing using integer grid points



goal: small grid size

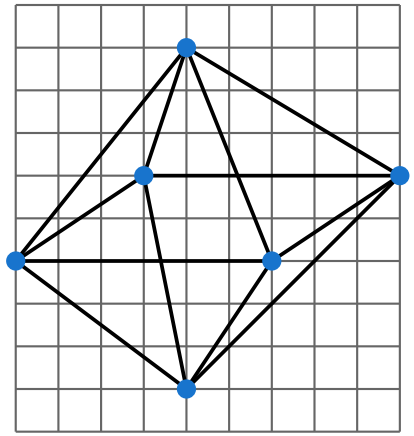
$\rightarrow$ implies good vertex and angular resolution

- number of bends



More Measures

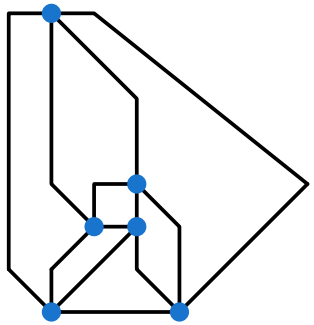
- grid size = area of the drawing using integer grid points



goal: small grid size

$\rightarrow$ implies good vertex and angular resolution

- number of bends

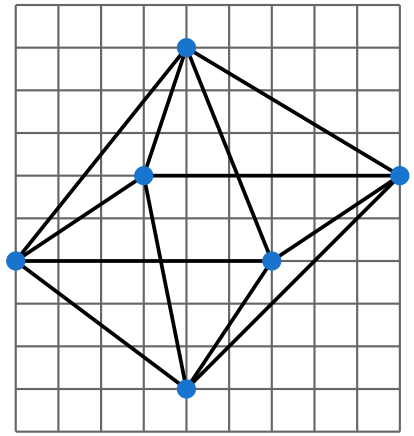


goal: minimize the number of total bends

goal: minimize the maximum number of bends per edge

More Measures

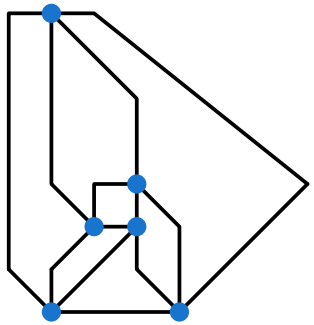
- grid size = area of the drawing using integer grid points



goal: small grid size

$\rightarrow$ implies good vertex and angular resolution

- number of bends



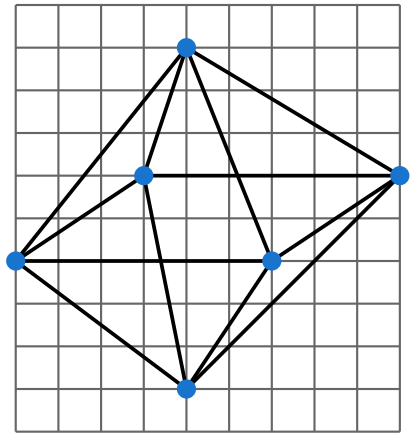
goal: minimize the number of total bends

goal: minimize the maximum number of bends per edge

- number of edge crossings

More Measures

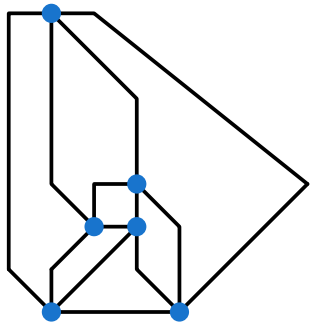
- grid size = area of the drawing using integer grid points



goal: small grid size

$\rightarrow$ implies good vertex and angular resolution

- number of bends



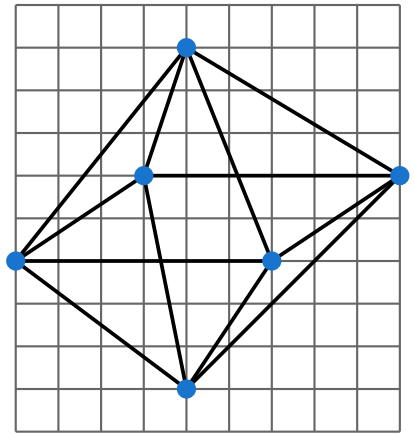
goal: minimize the number of total bends

goal: minimize the maximum number of bends per edge

- number of edge crossings
- and many more .....

More Measures

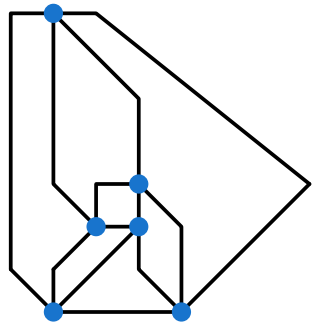
- grid size = area of the drawing using integer grid points



goal: small grid size

$\rightarrow$ implies good vertex and angular resolution

- number of bends



goal: minimize the number of total bends

goal: minimize the maximum number of bends per edge

- number of edge crossings
- and many more .....

Improving on one measure often decreases another measure!

Graph Classes

Graph Classes

Many problems become feasible or meaningful only when the graph class is restricted:

Graph Classes

Many problems become feasible or meaningful
only when the graph class is restricted:

- planar graphs (can be drawn without crossings)

Graph Classes

Many problems become feasible or meaningful
only when the graph class is restricted:

- planar graphs (can be drawn without crossings)
- trees (connected, no cycles)

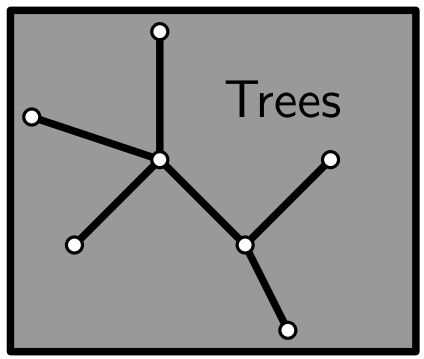
Graph Classes

Many problems become feasible or meaningful
only when the graph class is restricted:

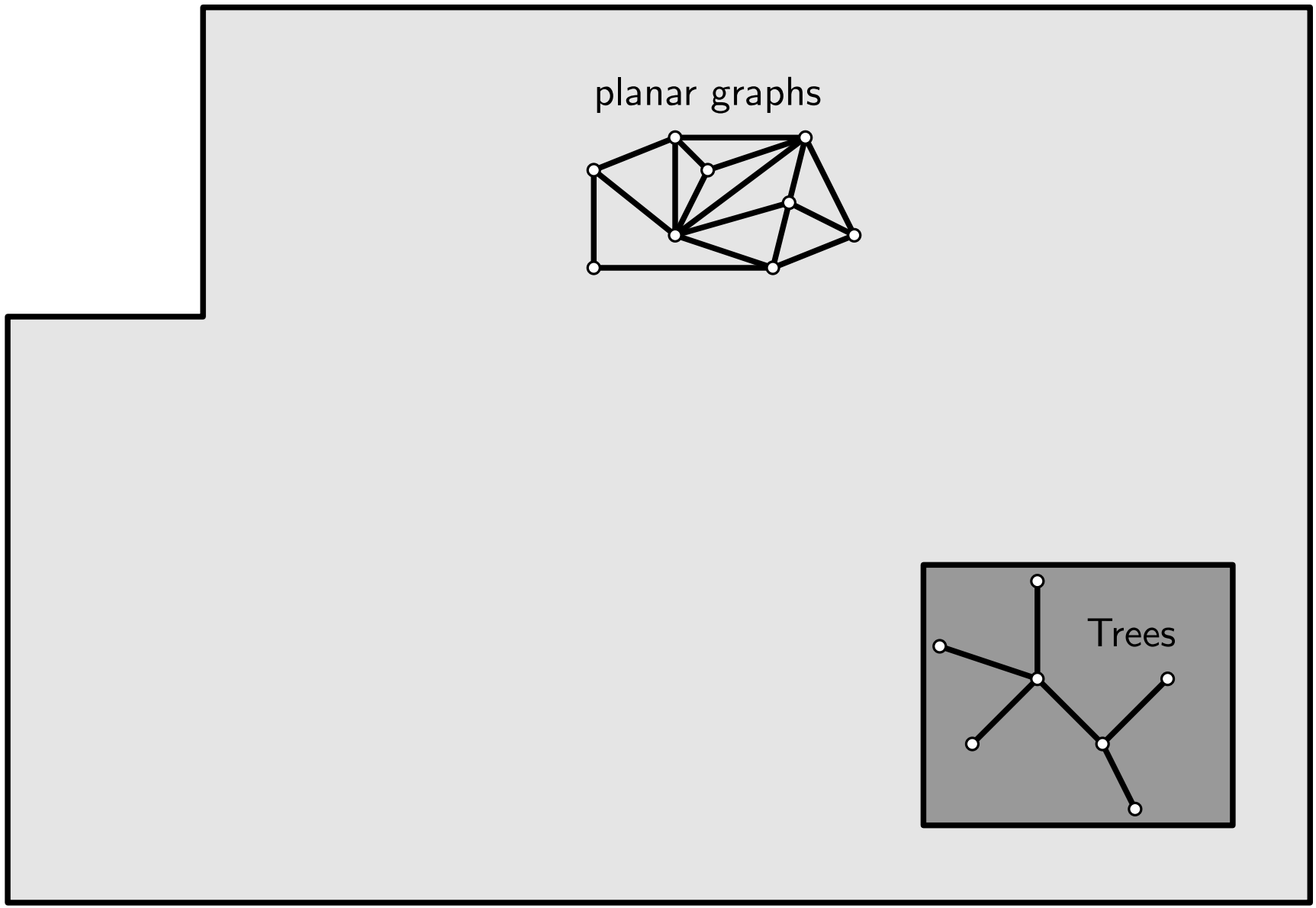
- planar graphs (can be drawn without crossings)
- trees (connected, no cycles)
- triangulations (maximal planar)
- planar 3-trees
- outerplanar graphs
- serial-parallel graphs
- k-connected graphs
- ...

Prominent Graph Classes by Example

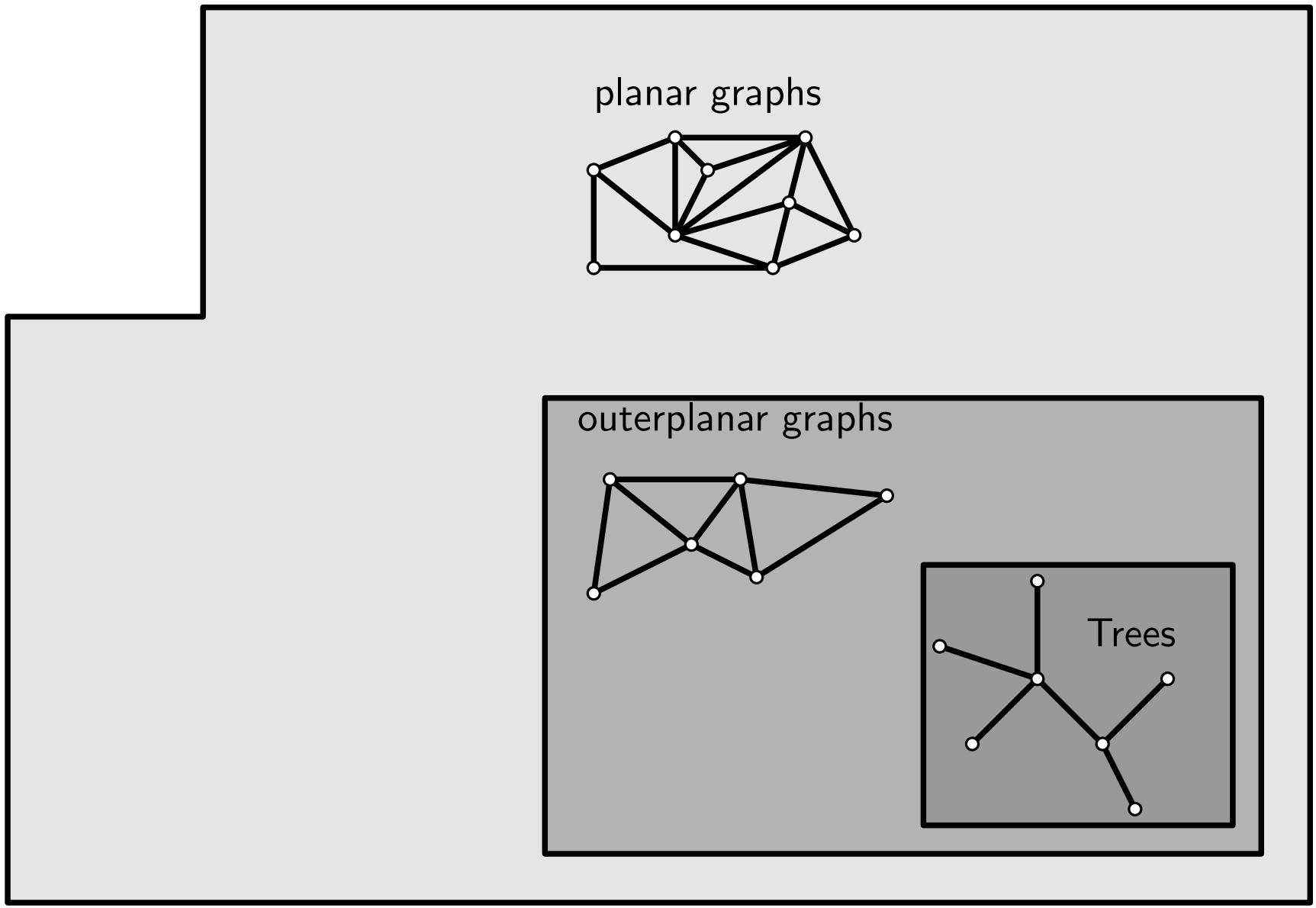
Prominent Graph Classes by Example



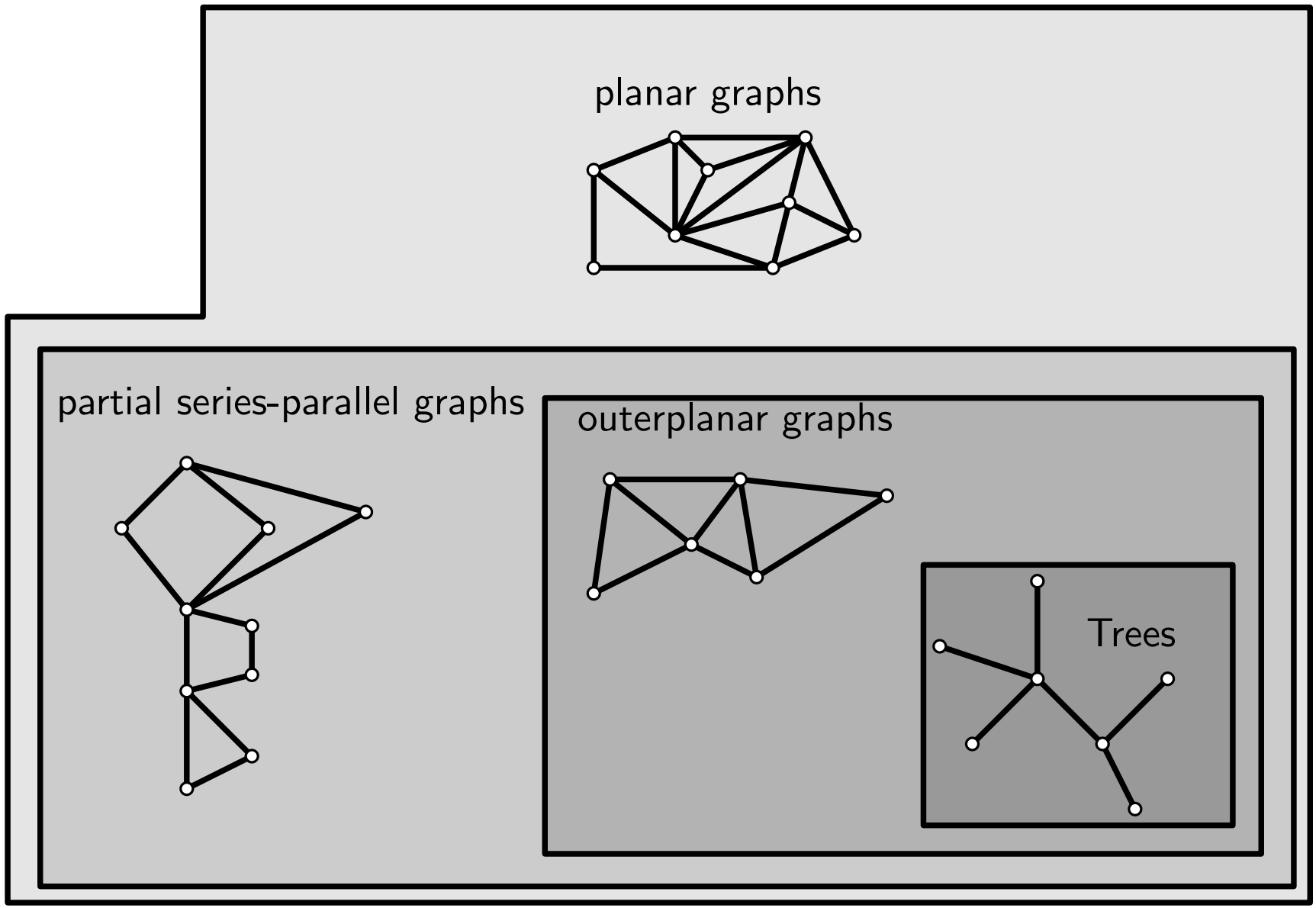
Prominent Graph Classes by Example



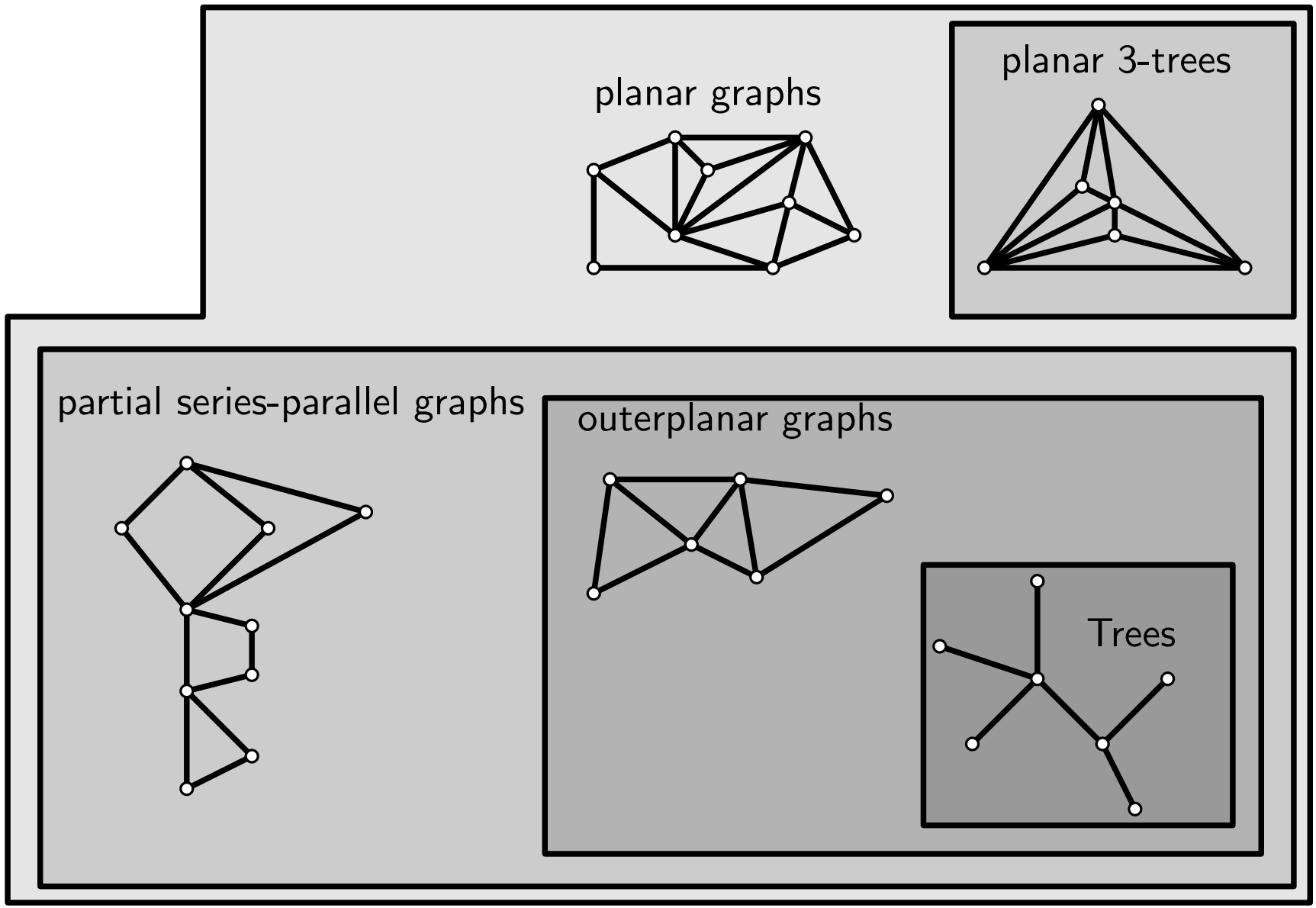
Prominent Graph Classes by Example



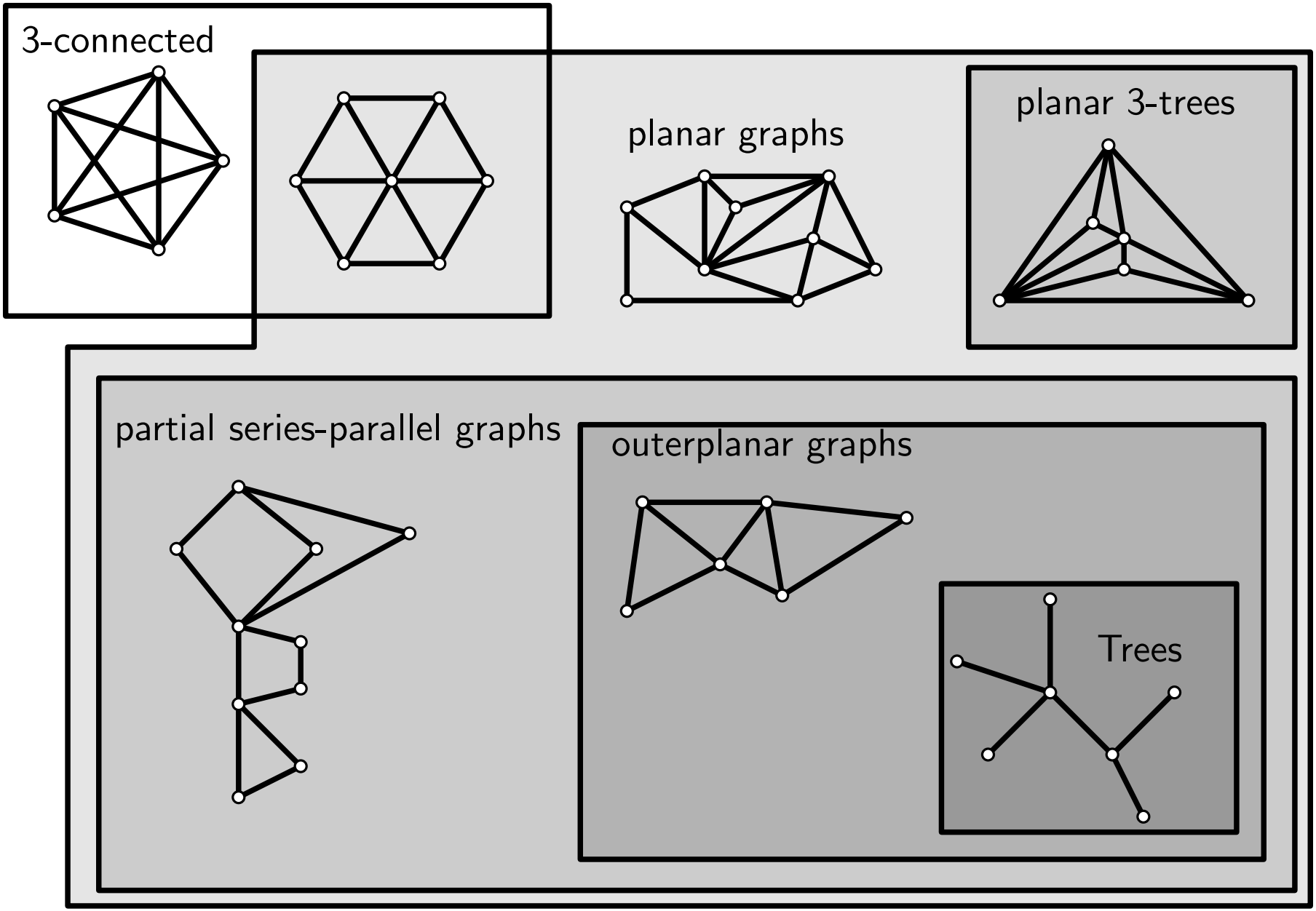
Prominent Graph Classes by Example



Prominent Graph Classes by Example



Prominent Graph Classes by Example



Standard Techniques I

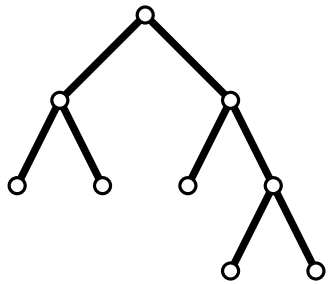
Standard Techniques I

- If your graph class has a recursive description, construct the graph drawing recursively.

Standard Techniques I

- If your graph class has a recursive description, construct the graph drawing recursively.

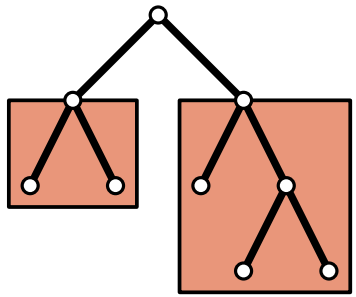
Binary trees:



Standard Techniques I

- If your graph class has a recursive description, construct the graph drawing recursively.

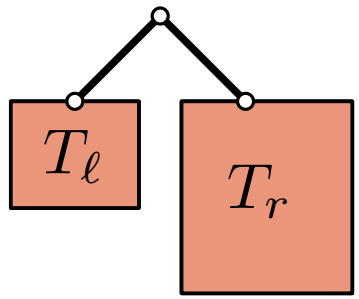
Binary trees:



Standard Techniques I

- If your graph class has a recursive description, construct the graph drawing recursively.

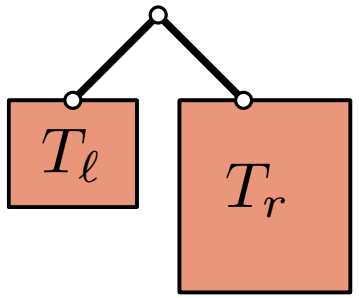
Binary trees:

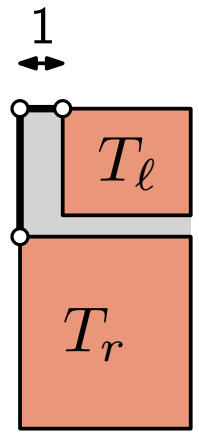


Standard Techniques I

- If your graph class has a recursive description, construct the graph drawing recursively.

Binary trees:



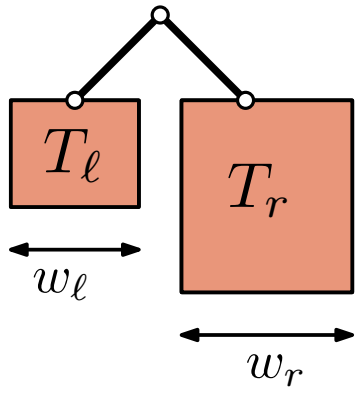


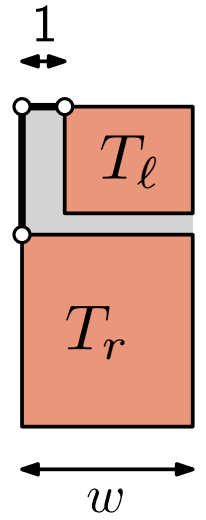
w.l.o.g. $|T_\ell| \leq |T_r|$

Standard Techniques I

- If your graph class has a recursive description, construct the graph drawing recursively.

Binary trees:





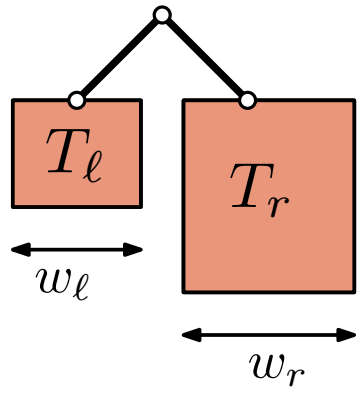
w.l.o.g. $|T_\ell| \leq |T_r|$

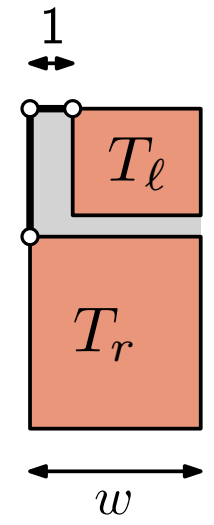
$w = \max\{w_l + 1, w_r\}$

Standard Techniques I

- If your graph class has a recursive description, construct the graph drawing recursively.

Binary trees:





w.l.o.g. $|T_\ell| \leq |T_r|$

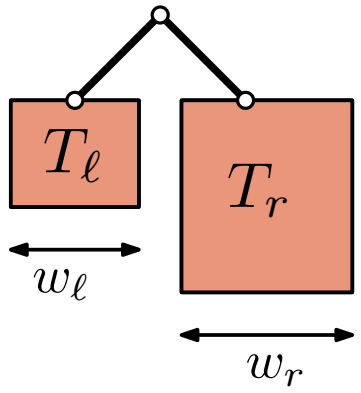
$w = \max\{w_l + 1, w_r\}$

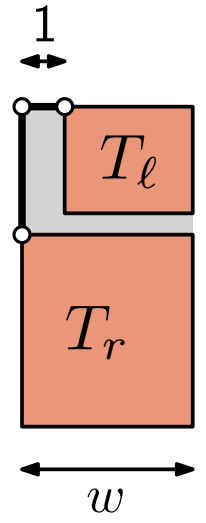
$$W(n) \leq W(n/2) + 1$$

Standard Techniques I

- If your graph class has a recursive description, construct the graph drawing recursively.

Binary trees:





w.l.o.g. $|T_\ell| \leq |T_r|$

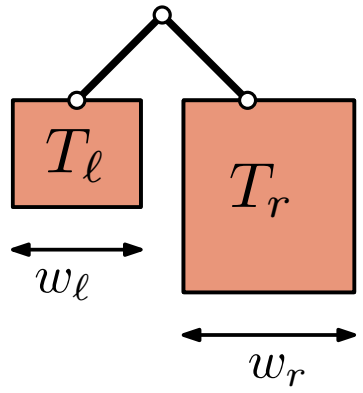
$w = \max\{w_l + 1, w_r\}$

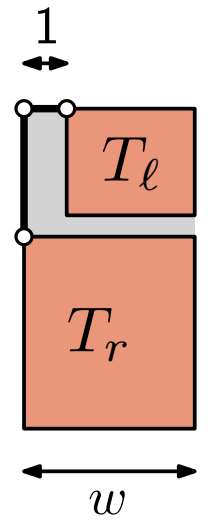
$$W(n) \leq W(n/2) + 1$$
$$W(n) = O(\log n)$$

Standard Techniques I

- If your graph class has a recursive description, construct the graph drawing recursively.

Binary trees:





w.l.o.g. $|T_\ell| \leq |T_r|$

$w = \max\{w_l + 1, w_r\}$

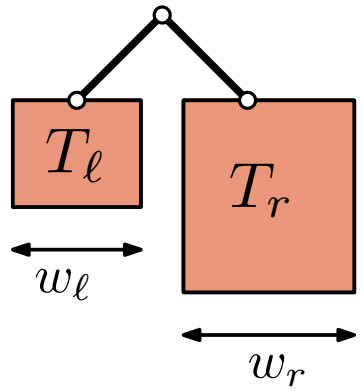
$W(n) \leq W(n/2) + 1$
$W(n) = O(\log n)$

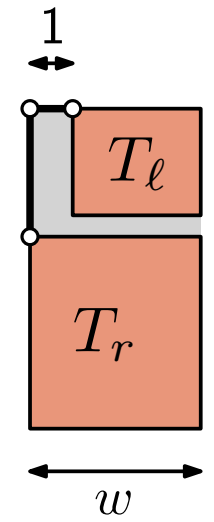
No row without vertex: $H(n) = O(n)$

Standard Techniques I

- If your graph class has a recursive description, construct the graph drawing recursively.

Binary trees:





w.l.o.g. $|T_\ell| \leq |T_r|$

$w = \max\{w_l + 1, w_r\}$

$W(n) \leq W(n/2) + 1$
$W(n) = O(\log n)$

No row without vertex: $H(n) = O(n)$

$\Rightarrow$ Area $O(n \log n)$ for upward grid drawing.

[Crescenzi, Di Battista, Piperno '92]

Standard Techniques II

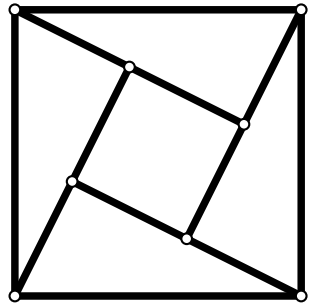
Standard Techniques II

- If your graph class has an inductive construction, build the graph drawing inductively.

Standard Techniques II

- If your graph class has an inductive construction, build the graph drawing inductively.

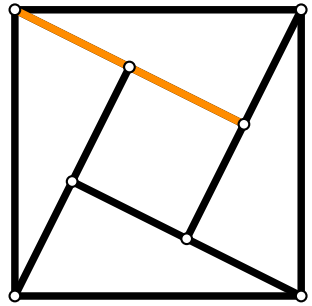
  Goal: Draw a graph with few segments.



Standard Techniques II

- If your graph class has an inductive construction, build the graph drawing inductively.

  Goal: Draw a graph with few segments.

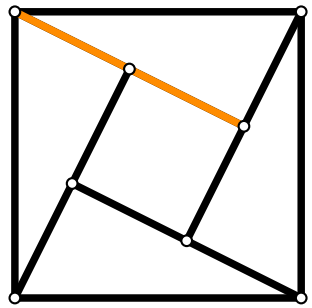


8 vertices
12 edges
8 segments

Standard Techniques II

- If your graph class has an inductive construction, build the graph drawing inductively.

  Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

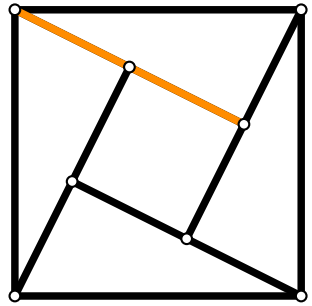
Planar 3-trees can be drawn with $2n - 4$ segments.

[Dujmović et al. '05]

Standard Techniques II

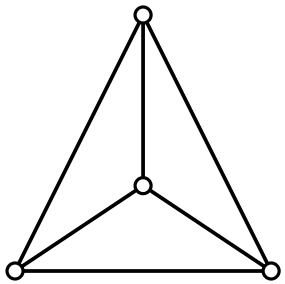
- If your graph class has an inductive construction, build the graph drawing inductively.

  Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

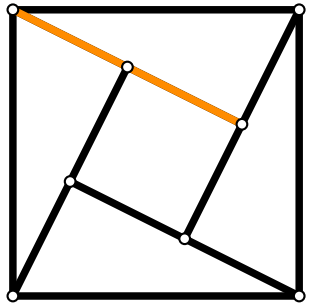
Planar 3-trees can be drawn with $2n - 4$ segments.
[Dujmović et al. '05]



Standard Techniques II

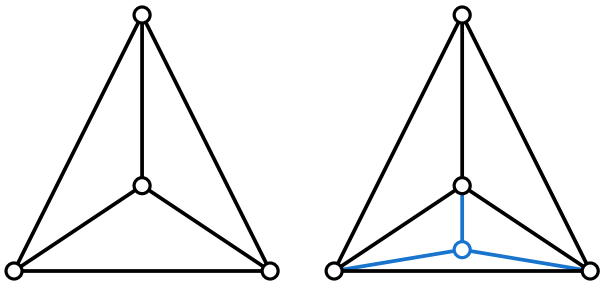
- If your graph class has an inductive construction, build the graph drawing inductively.

  Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

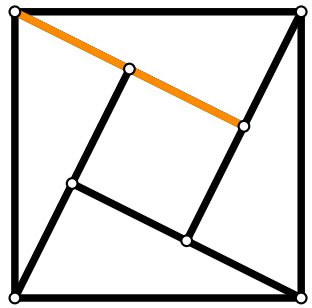
Planar 3-trees can be drawn with $2n - 4$ segments.
[Dujmović et al. '05]



Standard Techniques II

- If your graph class has an inductive construction, build the graph drawing inductively.

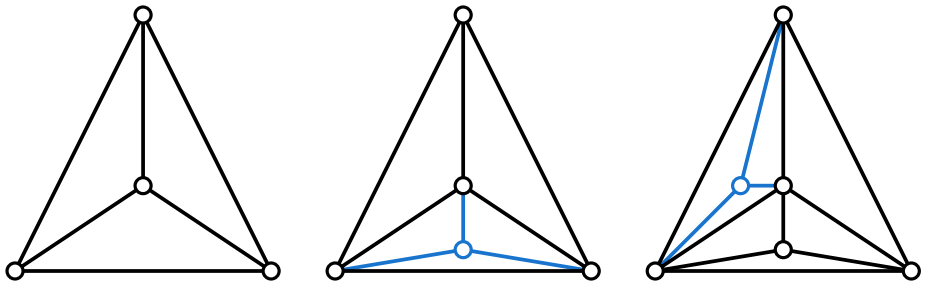
  Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

Planar 3-trees can be drawn with $2n - 4$ segments.

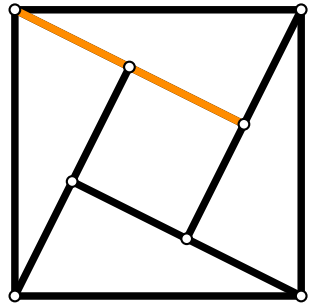
[Dujmović et al. '05]



Standard Techniques II

- If your graph class has an inductive construction, build the graph drawing inductively.

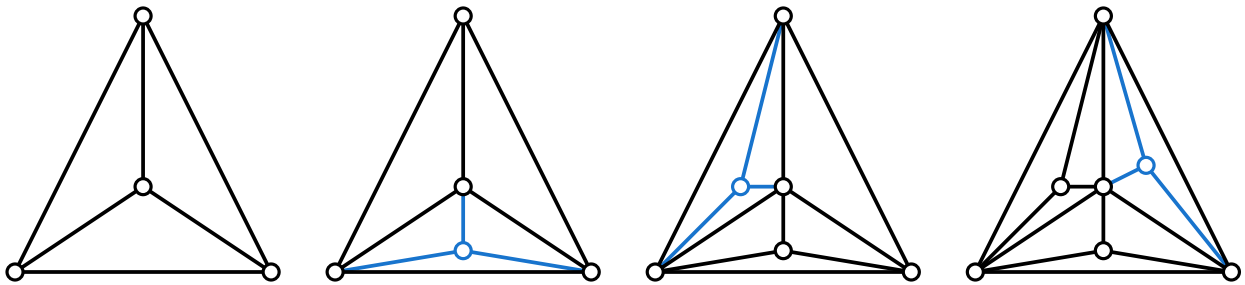
  Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

Planar 3-trees can be drawn with $2n - 4$ segments.

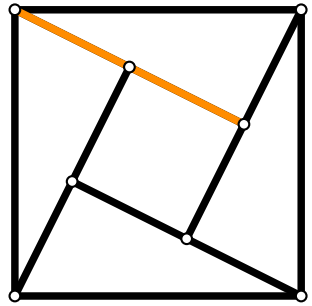
[Dujmović et al. '05]



Standard Techniques II

- If your graph class has an inductive construction, build the graph drawing inductively.

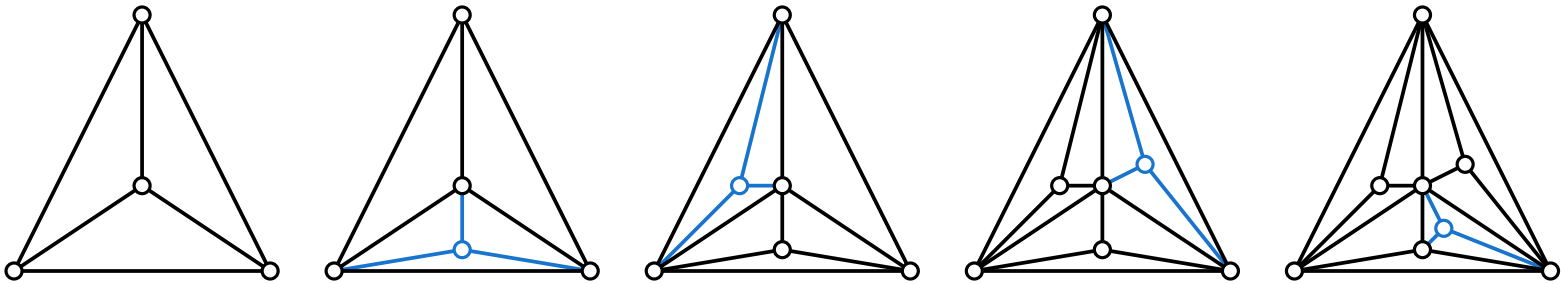
  Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

Planar 3-trees can be drawn with $2n - 4$ segments.

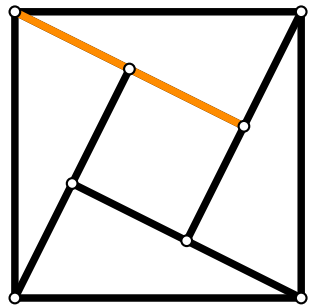
[Dujmović et al. '05]



Standard Techniques II

- If your graph class has an inductive construction, build the graph drawing inductively.

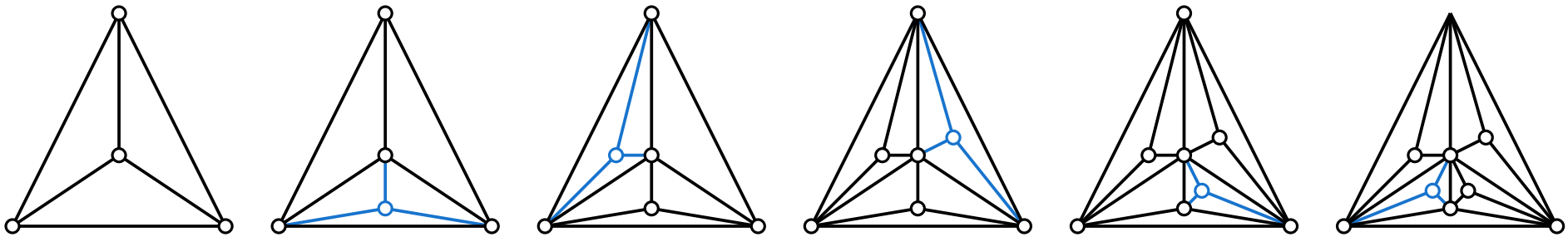
  Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

Planar 3-trees can be drawn with $2n - 4$ segments.

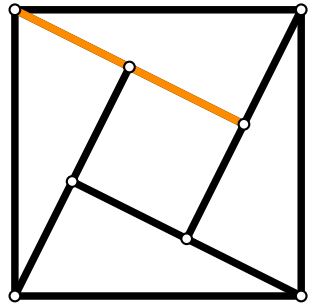
[Dujmović et al. '05]



Standard Techniques II

- If your graph class has an inductive construction, build the graph drawing inductively.

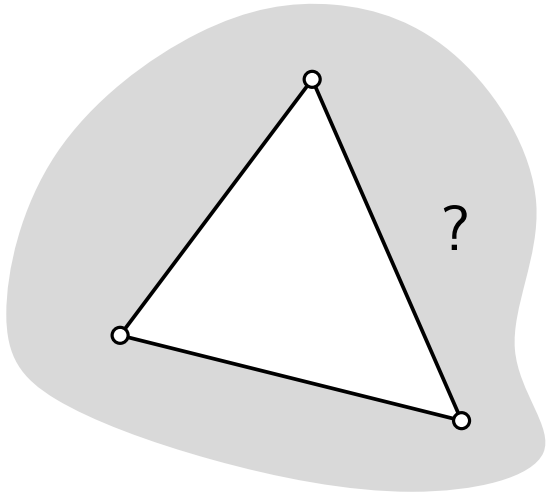
Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

Planar 3-trees can be drawn with $2n - 4$ segments.

[Dujmović et al. '05]

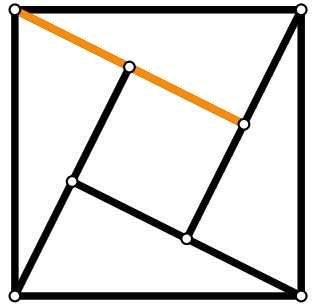


3-tree before last addition
$\leq 2(n-1) - 4$ segments

Standard Techniques II

- If your graph class has an inductive construction, build the graph drawing inductively.

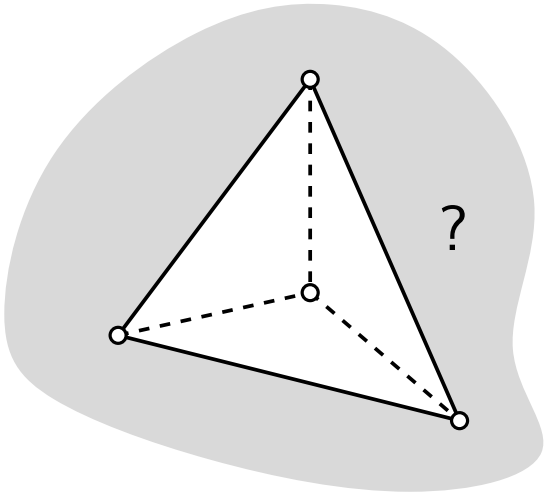
Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

Planar 3-trees can be drawn with $2n - 4$ segments.

[Dujmović et al. '05]

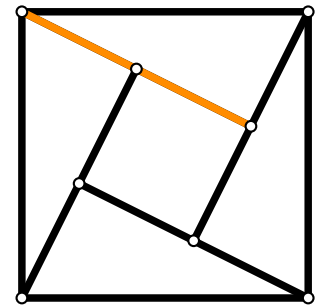


3-tree before last addition
$\leq 2(n-1) - 4$ segments

Standard Techniques II

- If your graph class has an inductive construction, build the graph drawing inductively.

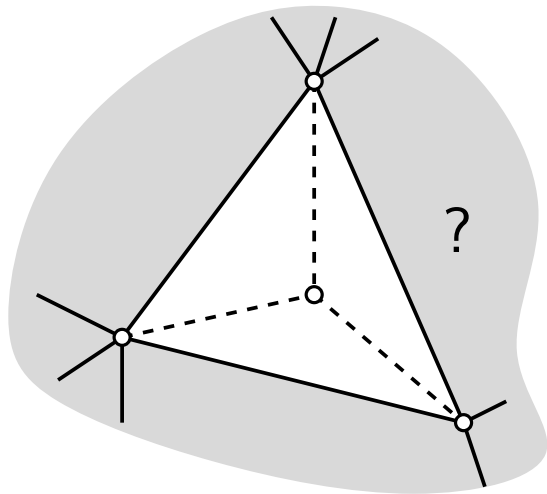
Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

Planar 3-trees can be drawn with $2n - 4$ segments.

[Dujmović et al. '05]

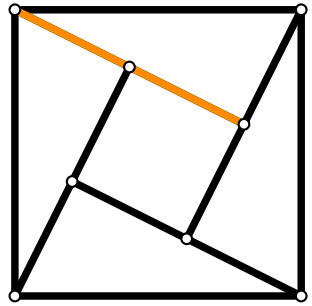


3-tree before last addition
$\leq 2(n-1) - 4$ segments

Standard Techniques II

- If your graph class has an inductive construction, build the graph drawing inductively.

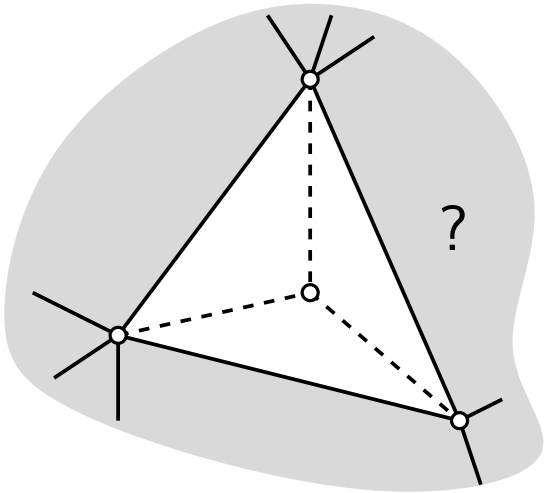
Goal: Draw a graph with few segments.



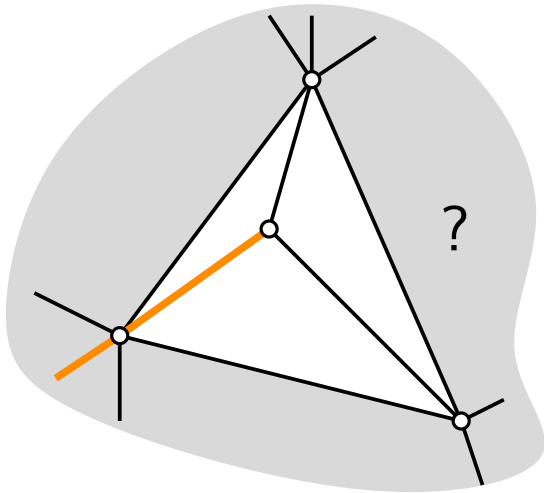
8 vertices
12 edges
8 segments

Planar 3-trees can be drawn with $2n - 4$ segments.

[Dujmović et al. '05]



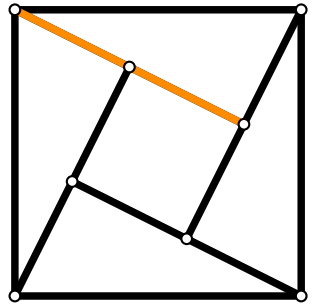
3-tree before last addition
$\leq 2(n-1) - 4$ segments



Standard Techniques II

- If your graph class has an inductive construction, build the graph drawing inductively.

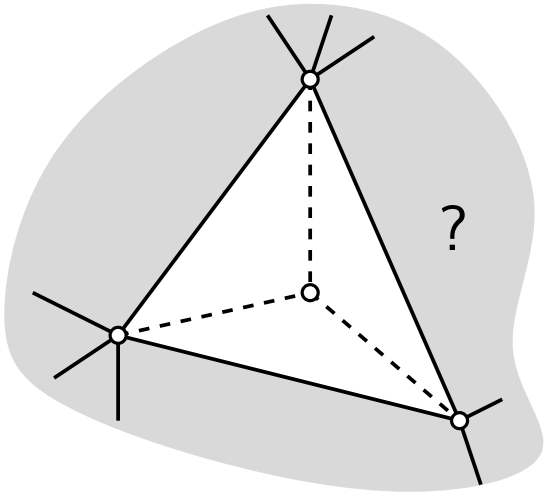
Goal: Draw a graph with few segments.



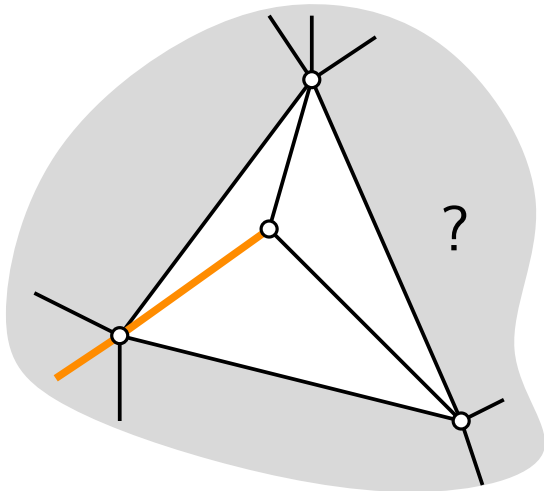
8 vertices
12 edges
8 segments

Planar 3-trees can be drawn with $2n-4$ segments.

[Dujmović et al. '05]



3-tree before last addition
$\leq 2(n-1)-4$ segments

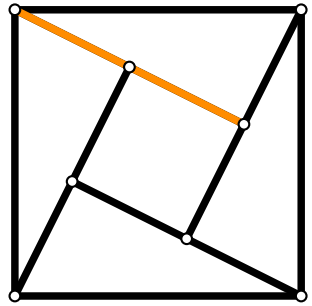


$\leq 2(n-1)-4+2 = 2n-4$ segments

Standard Techniques II

- If your graph class has an inductive construction, build the graph drawing inductively.

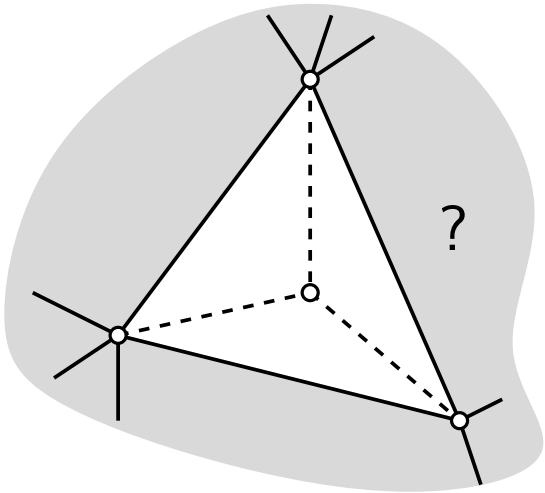
Goal: Draw a graph with few segments.



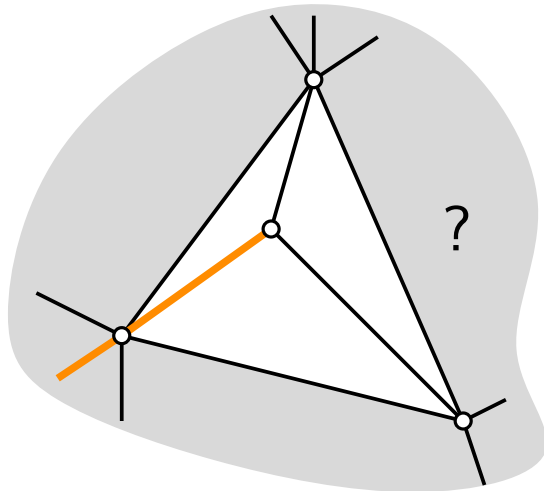
8 vertices
12 edges
8 segments

Planar 3-trees can be drawn with $2n-4$ segments.

[Dujmović et al. '05]



3-tree before last addition
$\leq 2(n-1)-4$ segments



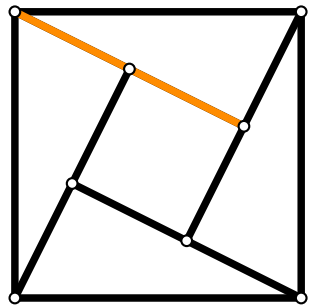
$\leq 2(n-1)-4+2 = 2n-4$ segments

3-connected planar graphs have an inductive construction sequence:

Standard Techniques II

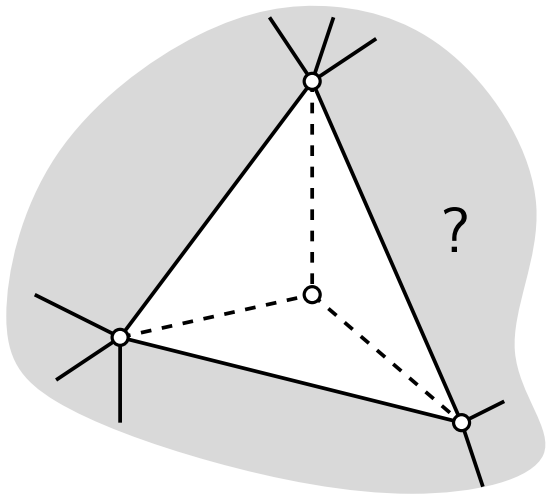
- If your graph class has an inductive construction, build the graph drawing inductively.

Goal: Draw a graph with few segments.

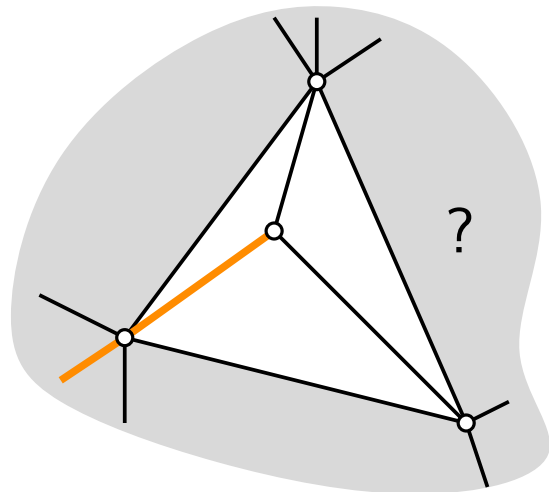


8 vertices
12 edges
8 segments

Planar 3-trees can be drawn with $2n-4$ segments.
[Dujmović et al. '05]



3-tree before last addition
$\leq 2(n-1)-4$ segments



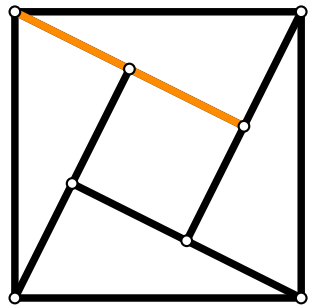
$\leq 2(n-1)-4+2 = 2n-4$ segments

3-connected planar graphs have an inductive construction sequence:
canonical ordering [De Fraysseix, Pach, Pollack '90]

Standard Techniques II

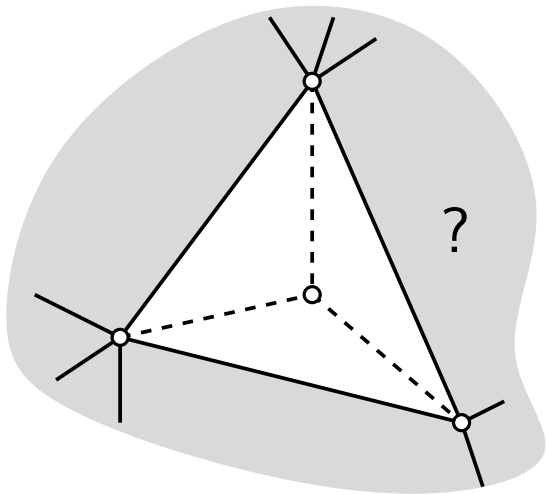
- If your graph class has an inductive construction, build the graph drawing inductively.

Goal: Draw a graph with few segments.

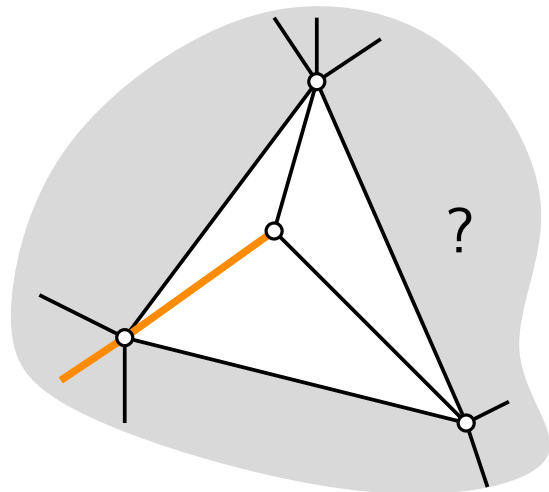


8 vertices
12 edges
8 segments

Planar 3-trees can be drawn with $2n-4$ segments.
[Dujmović et al. '05]



3-tree before last addition
$\leq 2(n-1)-4$ segments



$\leq 2(n-1)-4+2 = 2n-4$ segments

3-connected planar graphs have an inductive construction sequence:
canonical ordering [De Fraysseix, Pach, Pollack '90] @ boost C++ lib

More Ideas

More Ideas

- The approach that (arguably) works best in practice is the spring embedder.

More Ideas

- The approach that (arguably) works best in practice is the spring embedder.
- Model the graph as a physical system:
  1. all vertices repel
  2. adjacent vertices attract

More Ideas

- The approach that (arguably) works best in practice is the spring embedder.
- Model the graph as a physical system:

1. all vertices repel

$F_{ij} = 0$

(but pin a face)

2. adjacent vertices attract

$F_{ij} = \omega_{ij}(p_i - p_j)$

(like a spring)

More Ideas

- The approach that (arguably) works best in practice is the spring embedder.
- Model the graph as a physical system:

1. all vertices repel

$F_{ij} = 0$

(but pin a face)

2. adjacent vertices attract

$F_{ij} = \omega_{ij}(p_i - p_j)$

(like a spring)

Tutte '60

More Ideas

- The approach that (arguably) works best in practice is the spring embedder.
- Model the graph as a physical system:

1. all vertices repel

$F_{ij} = 0$

(but pin a face)

2. adjacent vertices attract

$F_{ij} = \omega_{ij}(p_i - p_j)$

(like a spring)

Tutte '60

$$F_{ij} = \frac{c_1}{\|p_i - p_j\|^{1/2}}(p_j - p_i)$$

$$F_{ij} = c_2 \log\left(\frac{\|p_i - p_j\|}{c_3}\right)(p_i - p_j)$$

Eades '84

More Ideas

- The approach that (arguably) works best in practice is the spring embedder.
- Model the graph as a physical system:

1. all vertices repel

$F_{ij} = 0$

(but pin a face)

2. adjacent vertices attract

$F_{ij} = \omega_{ij}(p_i - p_j)$

(like a spring)

Tutte '60

$$F_{ij} = \frac{c_1}{\|p_i - p_j\|^{1/2}}(p_j - p_i)$$

$$F_{ij} = c_2 \log\left(\frac{\|p_i - p_j\|}{c_3}\right)(p_i - p_j)$$

Eades '84

$$F_{ij} = \frac{k^2}{\|p_i - p_j\|}(p_j - p_i)$$

$$F_{ij} = \frac{\|p_i - p_j\|}{k}(p_i - p_j)$$

Fruchterman Reingold '92

Simultaneous Embedding

Simultaneous Embedding

Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$

Question: Can we place V such that each of these graphs has a planar (straight-line) drawing?

Simultaneous Embedding

Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$

Question: Can we place V such that each of these graphs has a planar (straight-line) drawing?

Two paths: YES

[Brass et al. '07]

Simultaneous Embedding

Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$

Question: Can we place V such that each of these graphs has a planar (straight-line) drawing?

Two paths: YES

[Brass et al. '07]

$\pi_1 = (v_2, v_1, v_5, v_3, v_6, v_4)$

$\pi_2 = (v_1, v_5, v_2, v_6, v_4, v_3)$

Simultaneous Embedding

Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$

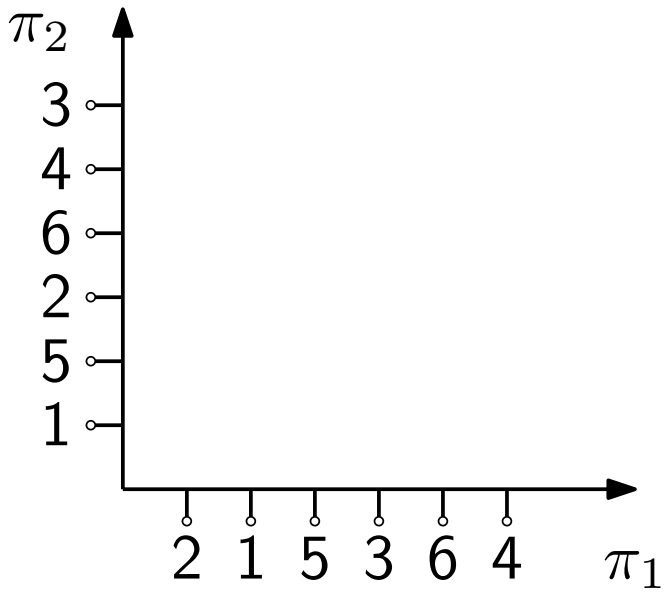
Question: Can we place V such that each of these graphs has a planar (straight-line) drawing?

Two paths: YES

[Brass et al. '07]

$\pi_1 = (v_2, v_1, v_5, v_3, v_6, v_4)$

$\pi_2 = (v_1, v_5, v_2, v_6, v_4, v_3)$



Simultaneous Embedding

Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$

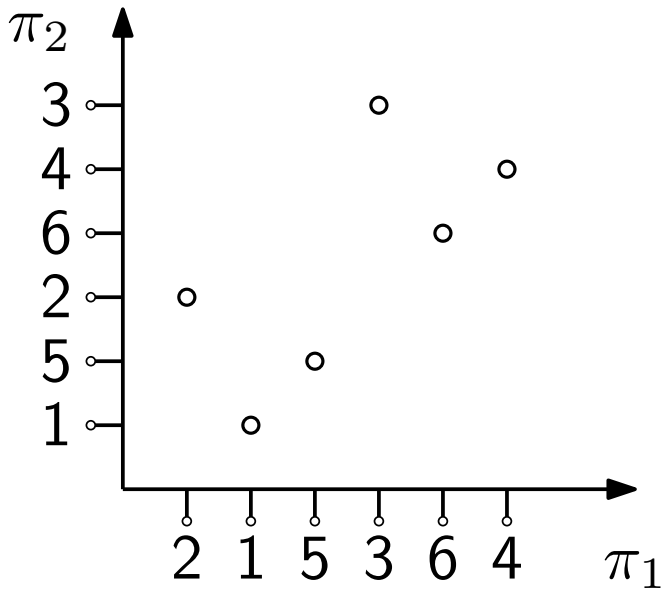
Question: Can we place V such that each of these graphs has a planar (straight-line) drawing?

Two paths: YES

[Brass et al. '07]

$\pi_1 = (v_2, v_1, v_5, v_3, v_6, v_4)$

$\pi_2 = (v_1, v_5, v_2, v_6, v_4, v_3)$



Simultaneous Embedding

Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$

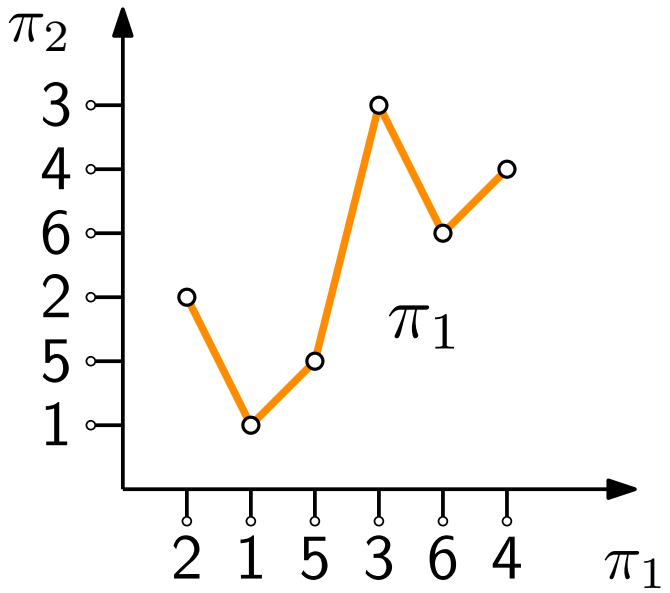
Question: Can we place V such that each of these graphs has a planar (straight-line) drawing?

Two paths: YES

[Brass et al. '07]

$\pi_1 = (v_2, v_1, v_5, v_3, v_6, v_4)$

$\pi_2 = (v_1, v_5, v_2, v_6, v_4, v_3)$



Simultaneous Embedding

Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$

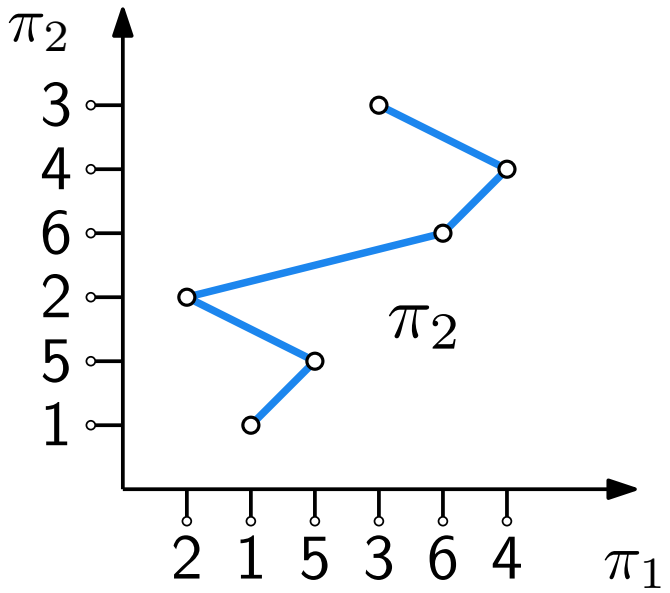
Question: Can we place V such that each of these graphs has a planar (straight-line) drawing?

Two paths: YES

[Brass et al. '07]

$\pi_1 = (v_2, v_1, v_5, v_3, v_6, v_4)$

$\pi_2 = (v_1, v_5, v_2, v_6, v_4, v_3)$



Simultaneous Embedding

Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$

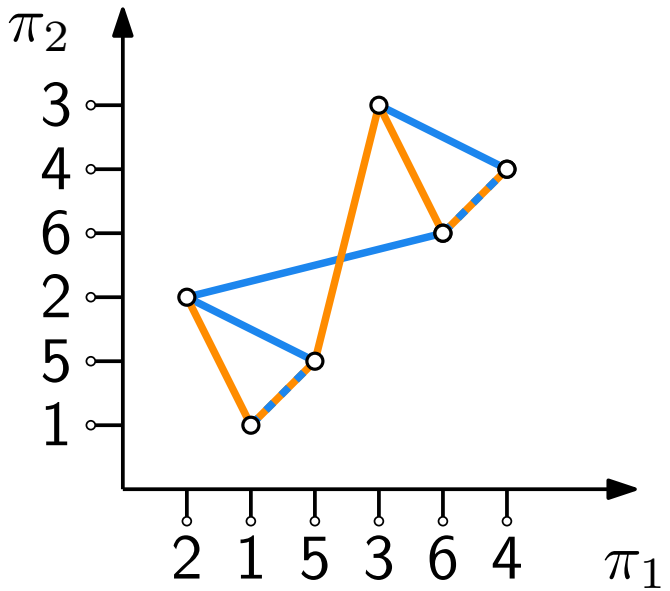
Question: Can we place V such that each of these graphs has a planar (straight-line) drawing?

Two paths: YES

[Brass et al. '07]

$\pi_1 = (v_2, v_1, v_5, v_3, v_6, v_4)$

$\pi_2 = (v_1, v_5, v_2, v_6, v_4, v_3)$



Simultaneous Embedding

Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$

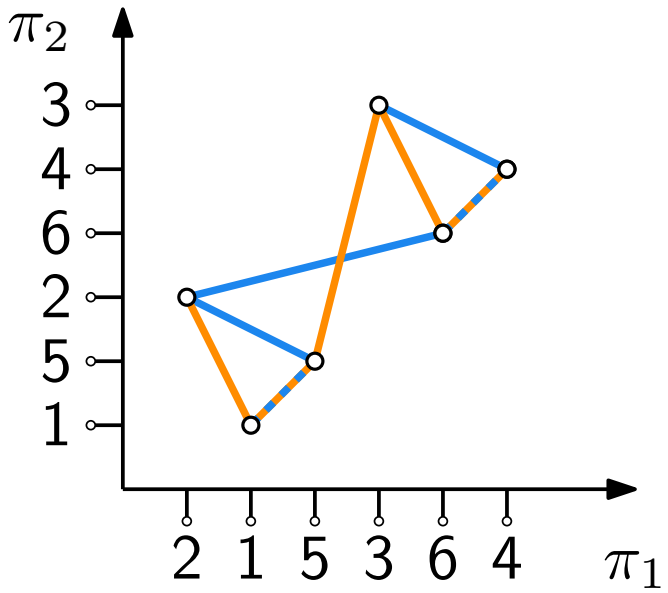
Question: Can we place V such that each of these graphs has a planar (straight-line) drawing?

Two paths: YES

[Brass et al. '07]

$\pi_1 = (v_2, v_1, v_5, v_3, v_6, v_4)$

$\pi_2 = (v_1, v_5, v_2, v_6, v_4, v_3)$



Six Matchings: NO

[Cabello et al. '07]

Simultaneous Embedding

Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$

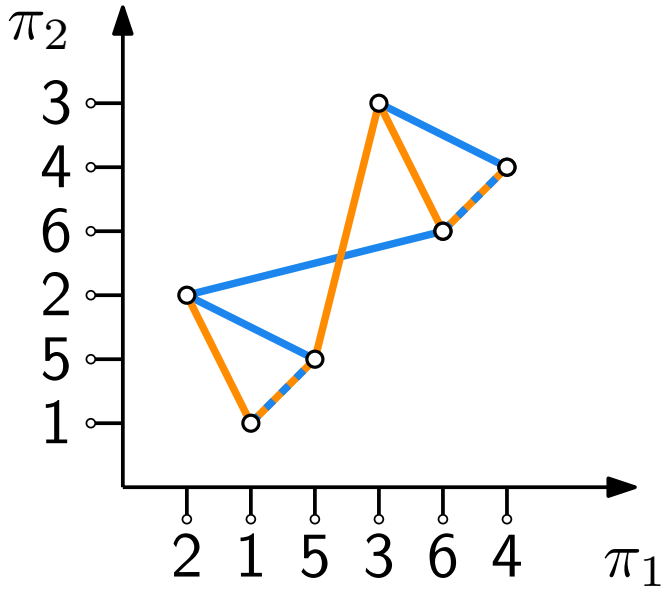
Question: Can we place V such that each of these graphs has a planar (straight-line) drawing?

Two paths: YES

[Brass et al. '07]

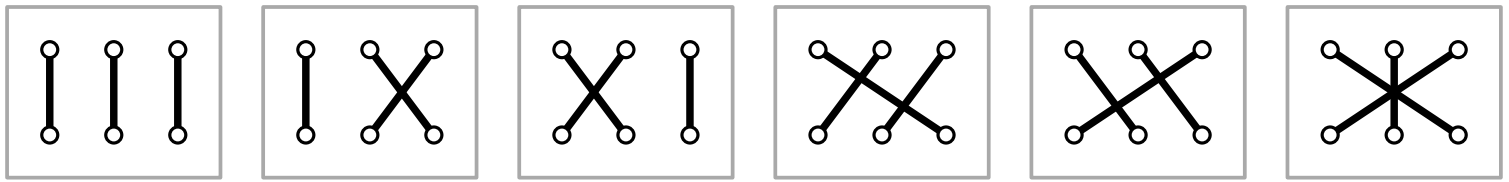
$\pi_1 = (v_2, v_1, v_5, v_3, v_6, v_4)$

$\pi_2 = (v_1, v_5, v_2, v_6, v_4, v_3)$



Six Matchings: NO

[Cabello et al. '07]



Simultaneous Embedding

Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$

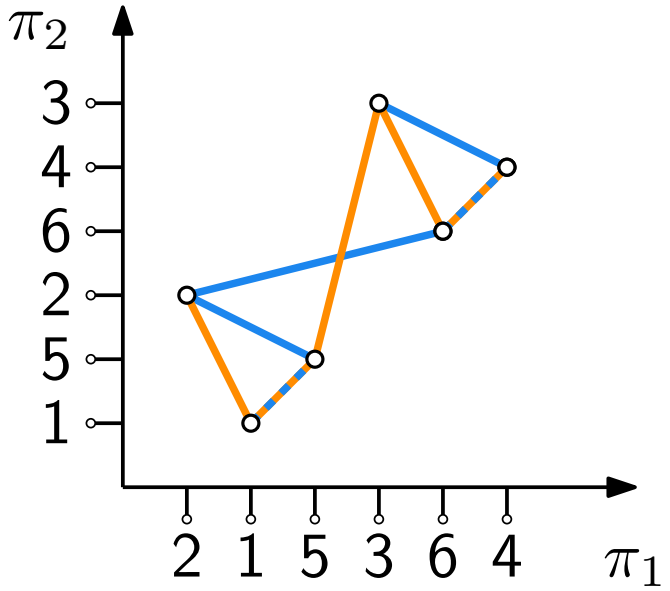
Question: Can we place V such that each of these graphs has a planar (straight-line) drawing?

Two paths: YES

[Brass et al. '07]

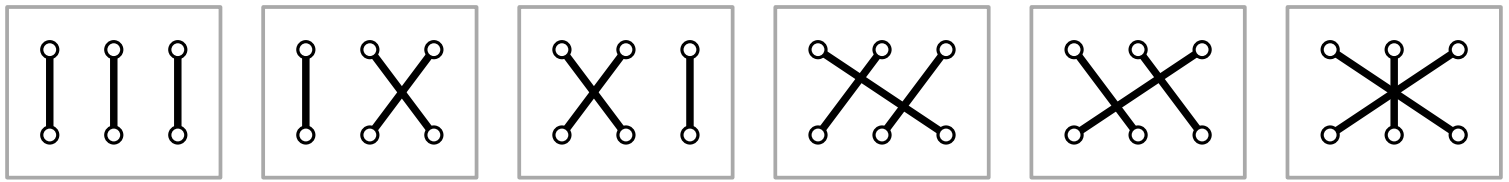
$\pi_1 = (v_2, v_1, v_5, v_3, v_6, v_4)$

$\pi_2 = (v_1, v_5, v_2, v_6, v_4, v_3)$



Six Matchings: NO

[Cabello et al. '07]



In a $K_{3,3}$-drawing at least two edges cross. For every pair of edges one matching contains these.

Simultaneous Embedding

Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$

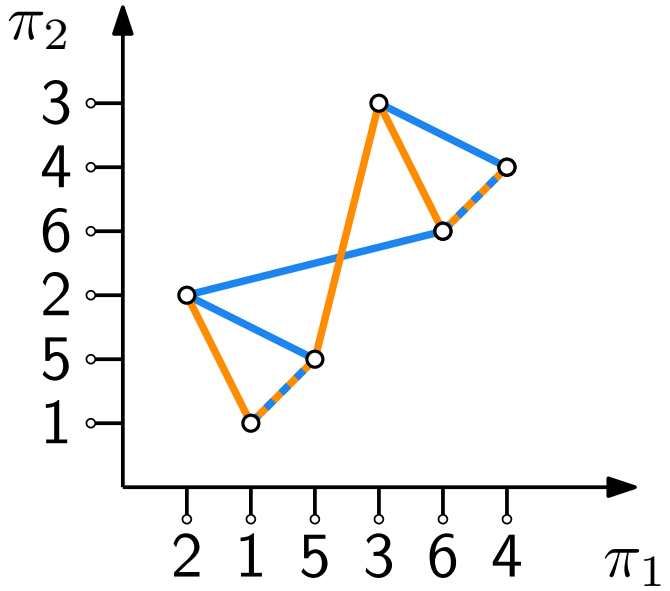
Question: Can we place V such that each of these graphs has a planar (straight-line) drawing?

Two paths: YES

[Brass et al. '07]

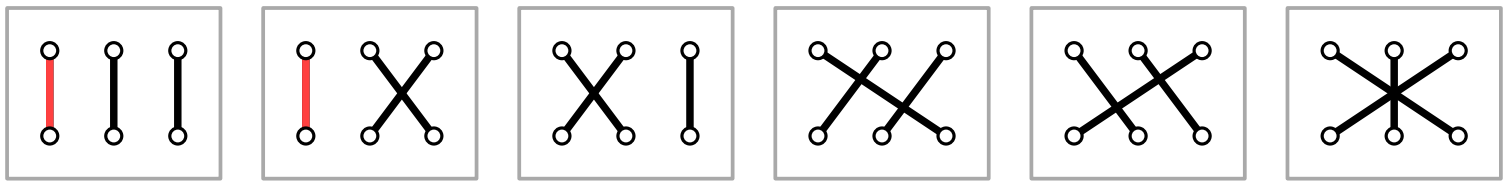
$\pi_1 = (v_2, v_1, v_5, v_3, v_6, v_4)$

$\pi_2 = (v_1, v_5, v_2, v_6, v_4, v_3)$



Six Matchings: NO

[Cabello et al. '07]



In a $K_{3,3}$-drawing at least two edges cross. For every pair of edges one matching contains these.

Morphing

Morphing

Given: Two plane drawings δ_1 and δ_2 of a planar graph $G = (V, E)$.

Question: Can we continuously deform δ_1 to δ_2 without introducing crossings?

Morphing

Given: Two plane drawings δ_1 and δ_2 of a planar graph $G = (V, E)$.

Question: Can we continuously deform δ_1 to δ_2 without introducing crossings?

Solution: [Floater & Gotsman '99]

- Compute (asymmetric) spring weights for the two drawings of G.
- Interpolate between weights and compute spring embedding.

Morphing

Given: Two plane drawings δ_1 and δ_2 of a planar graph $G = (V, E)$.

Question: Can we continuously deform δ_1 to δ_2 without introducing crossings?

Solution: [Floater & Gotsman '99]

- Compute (asymmetric) spring weights for the two drawings of G.
- Interpolate between weights and compute spring embedding.

$\rightarrow$ Works well in practice but gives complicated trajectories.

Morphing

Given: Two plane drawings δ_1 and δ_2 of a planar graph $G = (V, E)$.

Question: Can we continuously deform δ_1 to δ_2 without introducing crossings?

Solution: [Floater & Gotsman '99]

- Compute (asymmetric) spring weights for the two drawings of G.
- Interpolate between weights and compute spring embedding.

$\rightarrow$ Works well in practice but gives complicated trajectories.

Alternative Solution: [Angelini et al. '14]

- linear number of linear moves per vertex (worst-case opt.)
- complicated

3D

Given a graph G, find a set of planes in 3-space
such that there is a *crossing-free straight-line drawing* of G
with all vertices and edges drawn on these planes.

3D

Given a graph G, find a set of planes in 3-space
such that there is a *crossing-free straight-line drawing* of G
with all vertices and edges drawn on these planes.

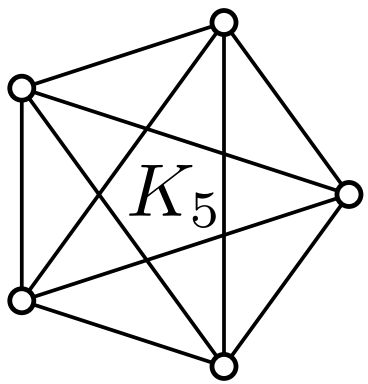
Call the minimum number of planes needed $\rho_3^2(G)$.

3D

Given a graph G, find a set of planes in 3-space
such that there is a *crossing-free straight-line drawing* of G
with all vertices and edges drawn on these planes.

Call the minimum number of planes needed $\rho_3^2(G)$.

$\rho_3^2(K_5) = ?$

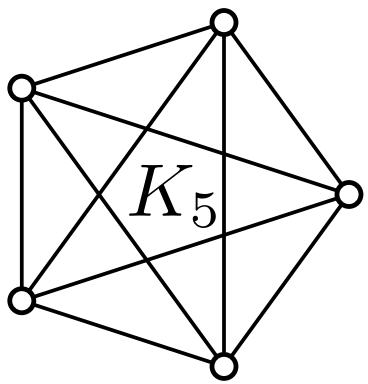


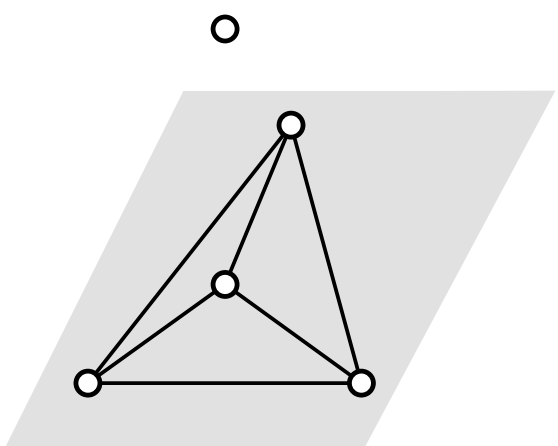
3D

Given a graph G, find a set of planes in 3-space
such that there is a *crossing-free straight-line drawing* of G
with all vertices and edges drawn on these planes.

Call the minimum number of planes needed $\rho_3^2(G)$.

$\rho_3^2(K_5) = ?$



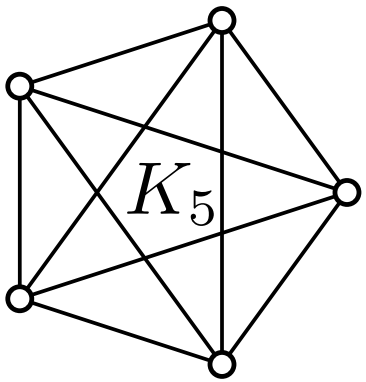


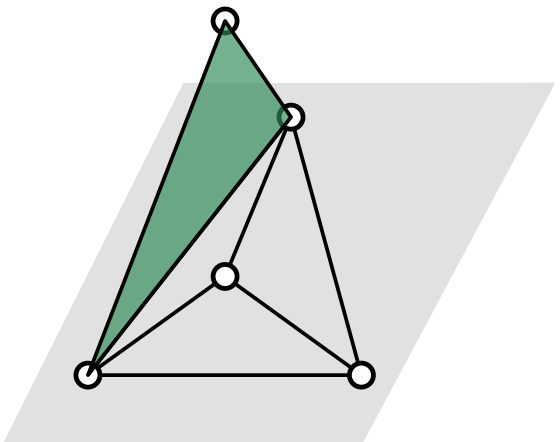
3D

Given a graph G, find a set of planes in 3-space
such that there is a *crossing-free straight-line drawing* of G
with all vertices and edges drawn on these planes.

Call the minimum number of planes needed $\rho_3^2(G)$.

$\rho_3^2(K_5) = ?$



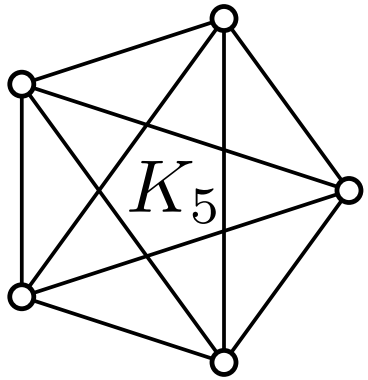


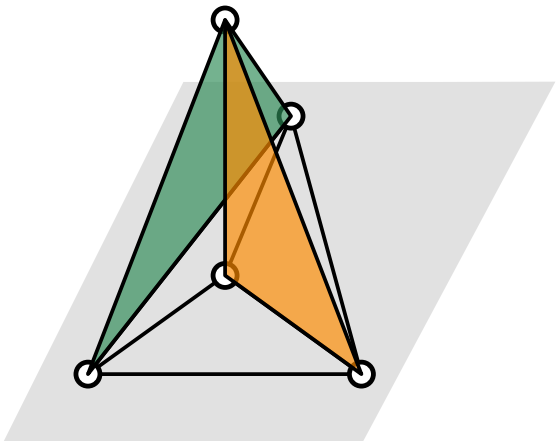
3D

Given a graph G, find a set of planes in 3-space
such that there is a *crossing-free straight-line drawing* of G
with all vertices and edges drawn on these planes.

Call the minimum number of planes needed $\rho_3^2(G)$.

$\rho_3^2(K_5) = ?$



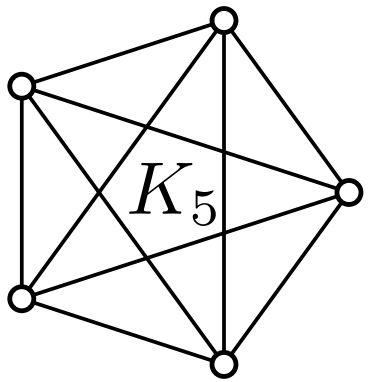


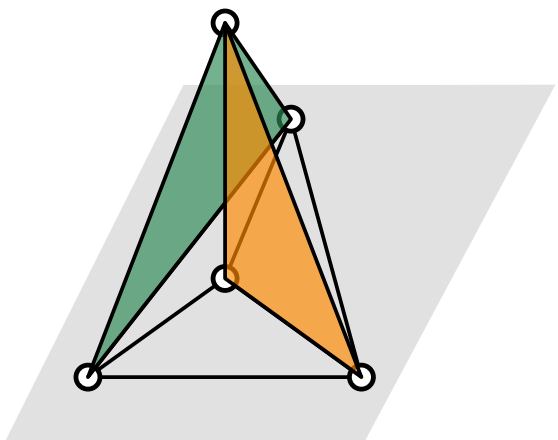
3D

Given a graph G, find a set of planes in 3-space
such that there is a *crossing-free straight-line drawing* of G
with all vertices and edges drawn on these planes.

Call the minimum number of planes needed $\rho_3^2(G)$.

$\rho_3^2(K_5) = 3$





3D

Given a graph G, find a set of planes in 3-space
such that there is a *crossing-free straight-line drawing* of G
with all vertices and edges drawn on these planes.

Call the minimum number of planes needed $\rho_3^2(G)$.

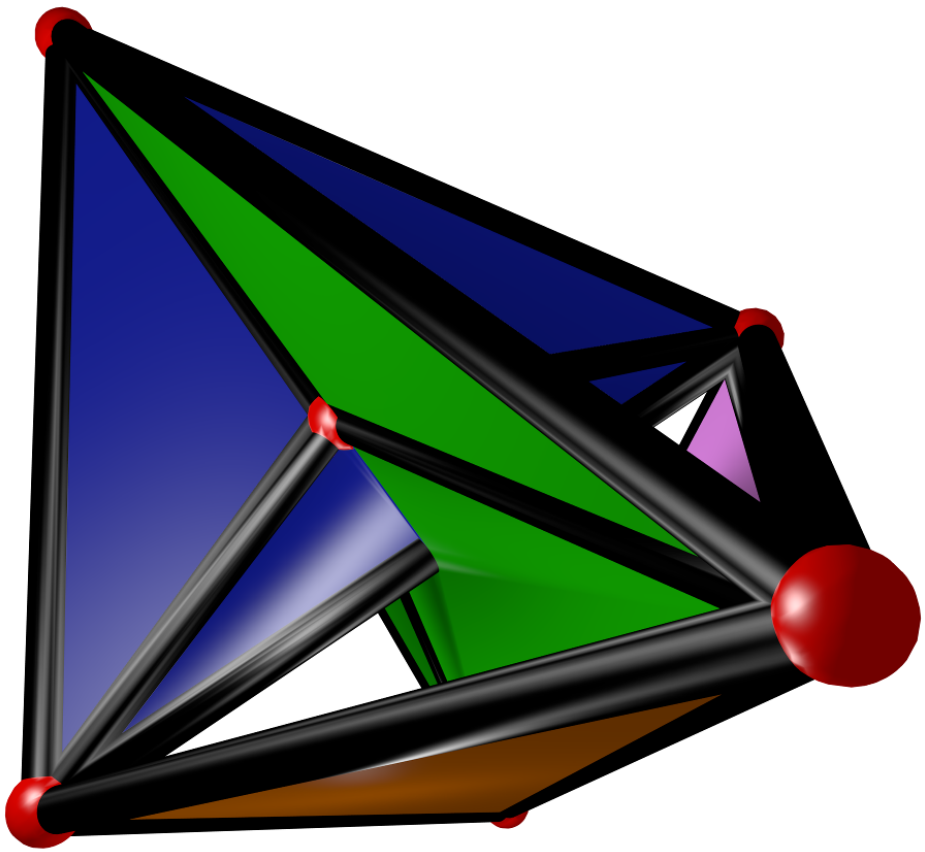
$\rho_3^2(K_6) = ?$

3D

Given a graph G, find a set of planes in 3-space
such that there is a *crossing-free straight-line drawing* of G
with all vertices and edges drawn on these planes.

Call the minimum number of planes needed $\rho_3^2(G)$.

$\rho_3^2(K_6) = ?$

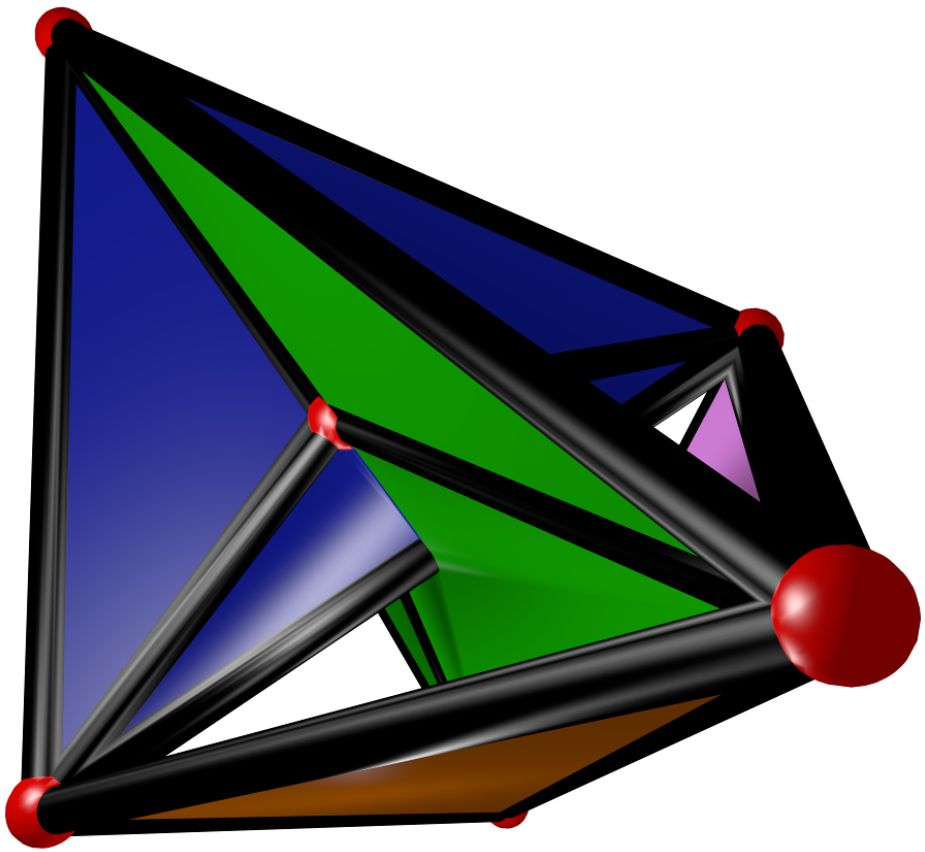


3D

Given a graph G, find a set of planes in 3-space
such that there is a *crossing-free straight-line drawing* of G
with all vertices and edges drawn on these planes.

Call the minimum number of planes needed $\rho_3^2(G)$.

$\rho_3^2(K_6) = 4$



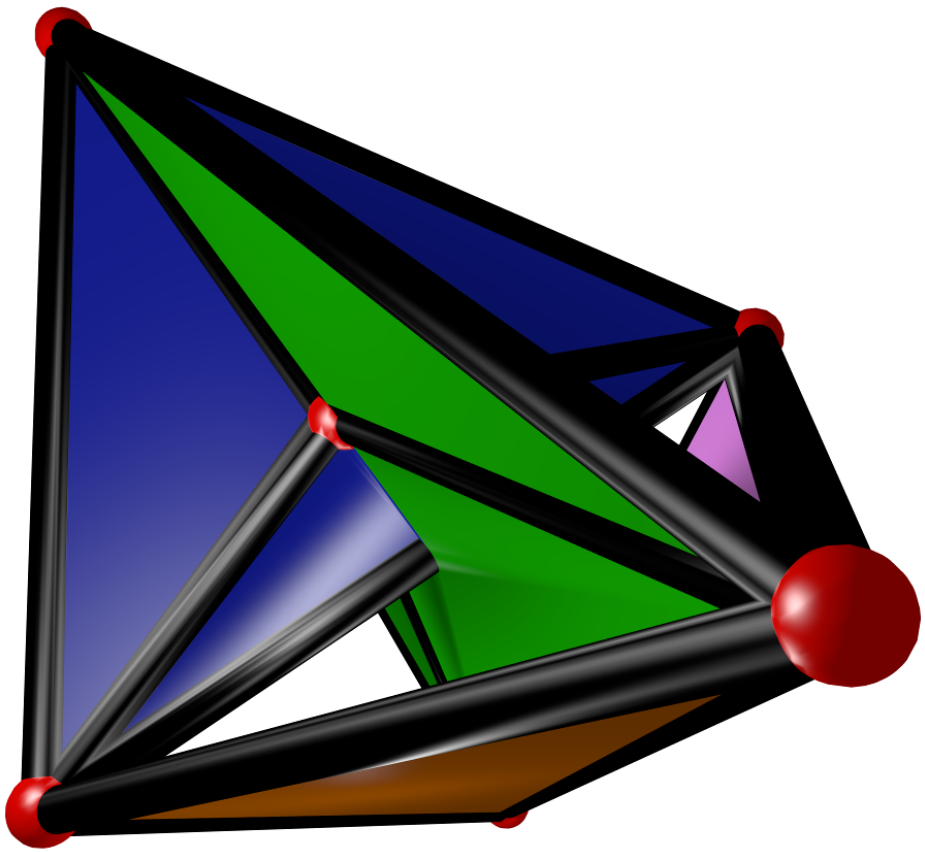
3D

Given a graph G, find a set of planes in 3-space
such that there is a *crossing-free straight-line drawing* of G
with all vertices and edges drawn on these planes.

Call the minimum number of planes needed $\rho_3^2(G)$.

$$\rho_3^2(K_6) = 4$$

For any planar graph G,
clearly $\rho_3^2(G) = 1$.

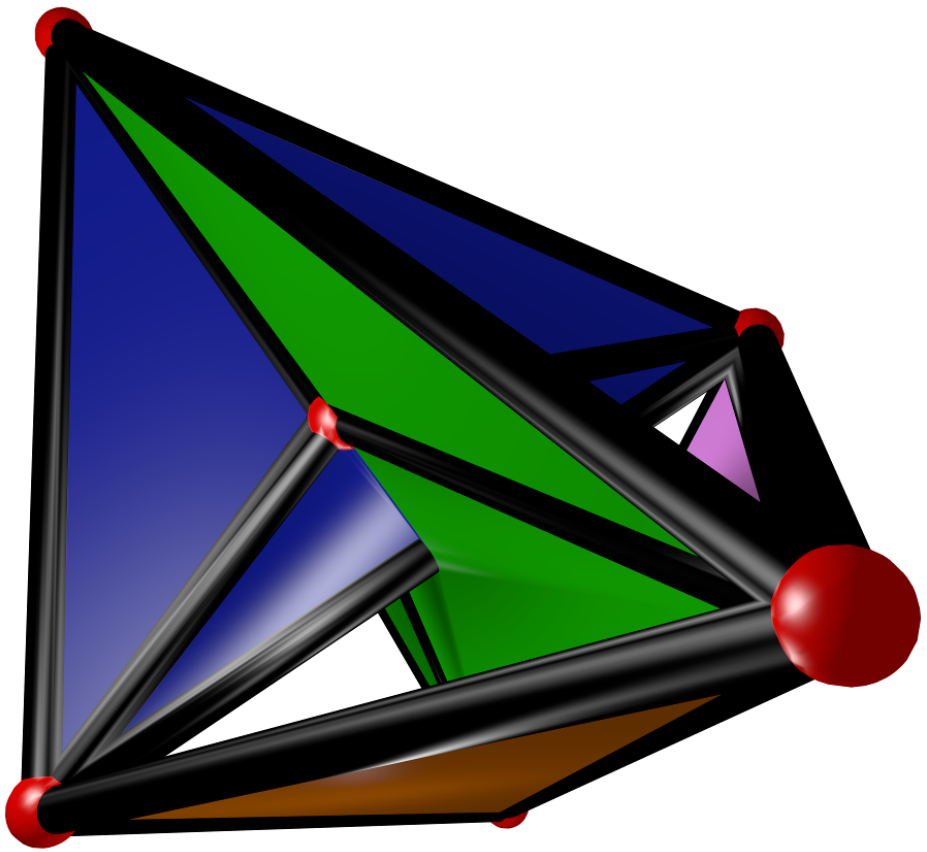


3D

Given a graph G, find a set of planes in 3-space such that there is a *crossing-free straight-line drawing* of G with all vertices and edges drawn on these planes.

Call the minimum number of planes needed $\rho_3^2(G)$.

$$\rho_3^2(K_6) = 4$$



For any planar graph G, clearly $\rho_3^2(G) = 1$.

Note: $\rho_3^2(K_n) \in \Theta(n^2)$.

$$\binom{n}{2}/6 \lessapprox \rho_3^2(K_n) \lessapprox \binom{n}{2}/3$$

Affine Cover Numbers

Let G be a graph and $1 \leq m < d$.

Affine cover number $\rho_d^m(G)$:

minimum number of m-dimensional hyperplanes in $\mathbb{R}^d$ s.t. G has a crossing-free straight-line drawing that is contained in these planes.

Affine Cover Numbers

Let G be a graph and $1 \leq m < d$.

Affine cover number $\rho_d^m(G)$:

minimum number of m-dimensional hyperplanes in $\mathbb{R}^d$ s.t. G has a crossing-free straight-line drawing that is contained in these planes.

Weak affine cover number $\pi_d^m(G)$:

requires only *vertices* to be contained in the planes.

Affine Cover Numbers

Let G be a graph and $1 \leq m < d$.

Affine cover number $\rho_d^m(G)$:

minimum number of m-dimensional hyperplanes in $\mathbb{R}^d$ s.t. G has a crossing-free straight-line drawing that is contained in these planes.

Weak affine cover number $\pi_d^m(G)$:

requires only *vertices* to be contained in the planes.

Observations

$\rho_d^m = \pi_d^m = 1$ for $m \geq 3$

Affine Cover Numbers

Let G be a graph and $1 \leq m < d$.

Affine cover number $\rho_d^m(G)$:

minimum number of m-dimensional hyperplanes in $\mathbb{R}^d$ s.t. G has a crossing-free straight-line drawing that is contained in these planes.

Weak affine cover number $\pi_d^m(G)$:

requires only *vertices* to be contained in the planes.

Observations

$\rho_d^m = \pi_d^m = 1$ for $m \geq 3$ $\qquad$ $\rho_d^m = \rho_3^m$ and $\pi_d^m = \pi_3^m$ for $d \geq 3$

Affine Cover Numbers

Let G be a graph and $1 \leq m < d$.

Affine cover number $\rho_d^m(G)$:

minimum number of m-dimensional hyperplanes in $\mathbb{R}^d$ s.t. G has a crossing-free straight-line drawing that is contained in these planes.

Weak affine cover number $\pi_d^m(G)$:

requires only *vertices* to be contained in the planes.

Observations

$\rho_d^m = \pi_d^m = 1$ for $m \geq 3$

"Collapse of the Affine Hierarchy"

$\rho_d^m = \rho_3^m$ and $\pi_d^m = \pi_3^m$ for $d \geq 3$

Affine Cover Numbers

Let G be a graph and $1 \leq m < d$.

Affine cover number $\rho_d^m(G)$:

minimum number of m-dimensional hyperplanes in $\mathbb{R}^d$ s.t. G has a crossing-free straight-line drawing that is contained in these planes.

Weak affine cover number $\pi_d^m(G)$:

requires only *vertices* to be contained in the planes.

Observations

"Collapse of the Affine Hierarchy"

$\rho_d^m = \pi_d^m = 1$ for $m \geq 3$

$\rho_d^m = \rho_3^m$ and $\pi_d^m = \pi_3^m$ for $d \geq 3$

$\pi_d^m \leq \rho_d^m$

Affine Cover Numbers

Let G be a graph and $1 \leq m < d$.

Affine cover number $\rho_d^m(G)$:

minimum number of m-dimensional hyperplanes in $\mathbb{R}^d$ s.t. G has a crossing-free straight-line drawing that is contained in these planes.

Weak affine cover number $\pi_d^m(G)$:

requires only *vertices* to be contained in the planes.

Observations

"Collapse of the Affine Hierarchy"

$\rho_d^m = \pi_d^m = 1$ for $m \geq 3$

$\rho_d^m = \rho_3^m$ and $\pi_d^m = \pi_3^m$ for $d \geq 3$

$\pi_d^m \leq \rho_d^m$

$\rho_3^2 \leq \rho_3^1 \leq \rho_2^1$

$\pi_3^2 \leq \pi_3^1 \leq \pi_2^1$

Affine Cover Numbers

Let G be a graph and $1 \leq m < d$.

Affine cover number $\rho_d^m(G)$:

minimum number of m-dimensional hyperplanes in $\mathbb{R}^d$ s.t. G has a crossing-free straight-line drawing that is contained in these planes.

Weak affine cover number $\pi_d^m(G)$:

requires only *vertices* to be contained in the planes.

Observations

"Collapse of the Affine Hierarchy"

$\rho_d^m = \pi_d^m = 1$ for $m \geq 3$ $\quad$ $\rho_d^m = \rho_3^m$ and $\pi_d^m = \pi_3^m$ for $d \geq 3$

$\pi_d^m \leq \rho_d^m$ $\quad$ $\rho_3^2 \leq \rho_3^1 \leq \rho_2^1$ $\quad$ $\pi_3^2 \leq \pi_3^1 \leq \pi_2^1$

Interesting cases

- Line cover numbers in 2D and 3D: ρ_2^1, ρ_3^1, π_2^1, π_3^1

Affine Cover Numbers

Let G be a graph and $1 \leq m < d$.

Affine cover number $\rho_d^m(G)$:

minimum number of m-dimensional hyperplanes in $\mathbb{R}^d$ s.t. G has a crossing-free straight-line drawing that is contained in these planes.

Weak affine cover number $\pi_d^m(G)$:

requires only *vertices* to be contained in the planes.

Observations

"Collapse of the Affine Hierarchy"

$\rho_d^m = \pi_d^m = 1$ for $m \geq 3$ $\quad$ $\rho_d^m = \rho_3^m$ and $\pi_d^m = \pi_3^m$ for $d \geq 3$

$\pi_d^m \leq \rho_d^m$ $\quad$ $\rho_3^2 \leq \rho_3^1 \leq \rho_2^1$ $\quad$ $\pi_3^2 \leq \pi_3^1 \leq \pi_2^1$

Interesting cases

- Line cover numbers in 2D and 3D: ρ_2^1, ρ_3^1, π_2^1, π_3^1
- Plane cover numbers in 3D: ρ_3^2, π_3^2

Affine Cover Numbers

Let G be a graph and $1 \leq m < d$.

Affine cover number $\rho_d^m(G)$:

minimum number of m-dimensional hyperplanes in $\mathbb{R}^d$ s.t. G has a crossing-free straight-line drawing that is contained in these planes.

Weak affine cover number $\pi_d^m(G)$:

requires only *vertices* to be contained in the planes.

Observations

"Collapse of the Affine Hierarchy"

$\rho_d^m = \pi_d^m = 1$ for $m \geq 3$ $\quad$ $\rho_d^m = \rho_3^m$ and $\pi_d^m = \pi_3^m$ for $d \geq 3$

$\pi_d^m \leq \rho_d^m$ $\quad$ $\rho_3^2 \leq \rho_3^1 \leq \rho_2^1$ $\quad$ $\pi_3^2 \leq \pi_3^1 \leq \pi_2^1$

Interesting cases

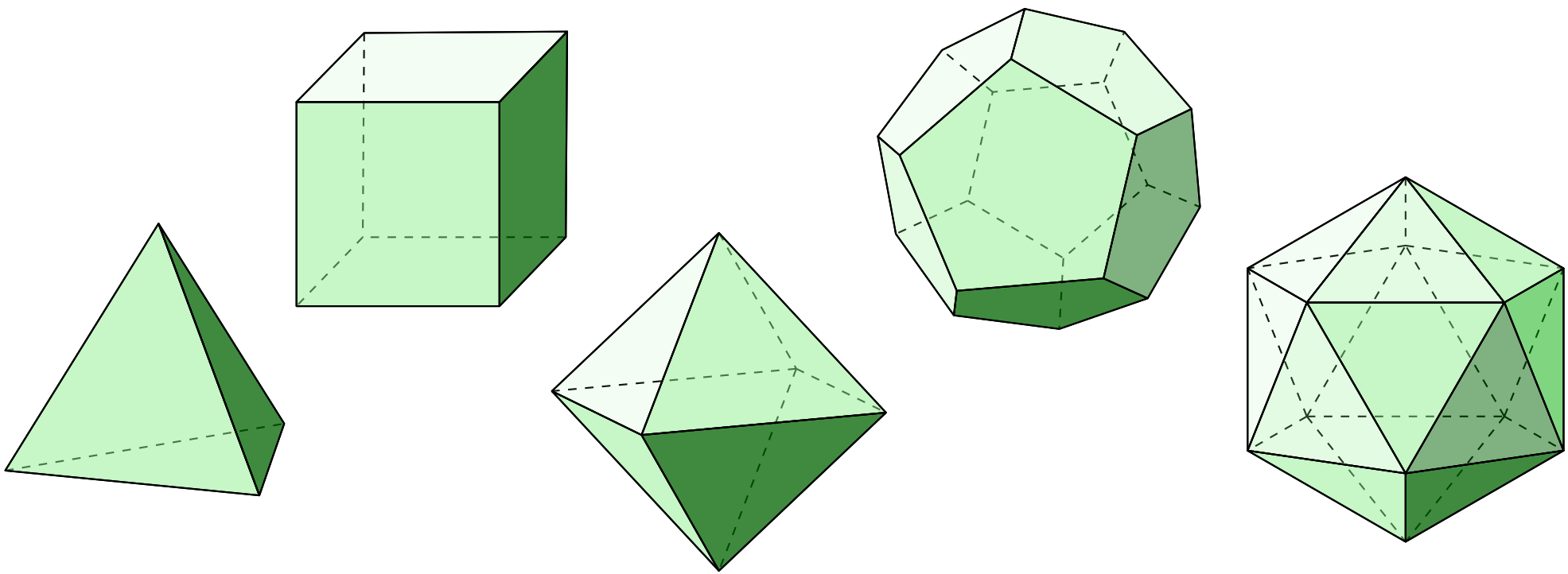
- Line cover numbers in 2D and 3D: ρ_2^1, ρ_3^1, π_2^1, π_3^1
- Plane cover numbers in 3D: ρ_3^2, π_3^2

[WADS'17 & GD'19]

Unfortunately, *each* of these numbers is NP-hard to compute :-(

Line Cover Numbers of the Platonic Solids

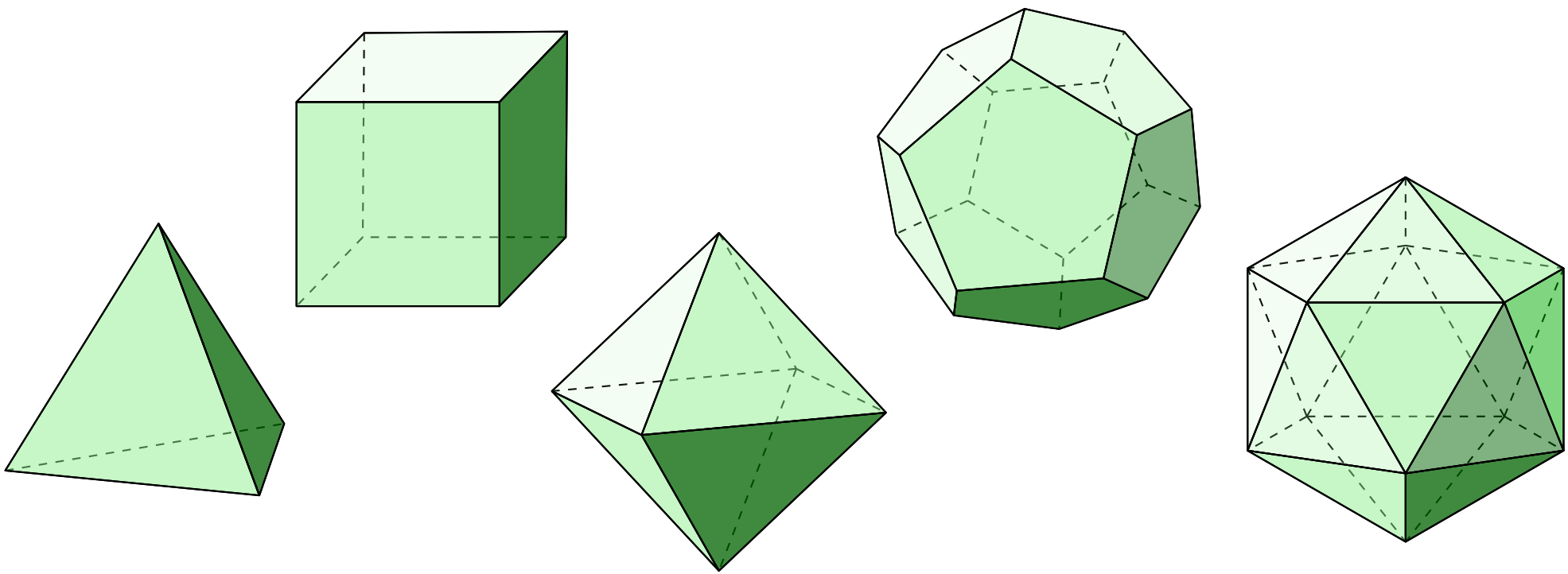
$G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4				
cube	8	12	6				
octahedron	6	12	8				
dodecahedron	20	30	12				
icosahedron	12	30	20				



Line Cover Numbers of the Platonic Solids

[Kryven et al., CALDAM'18]

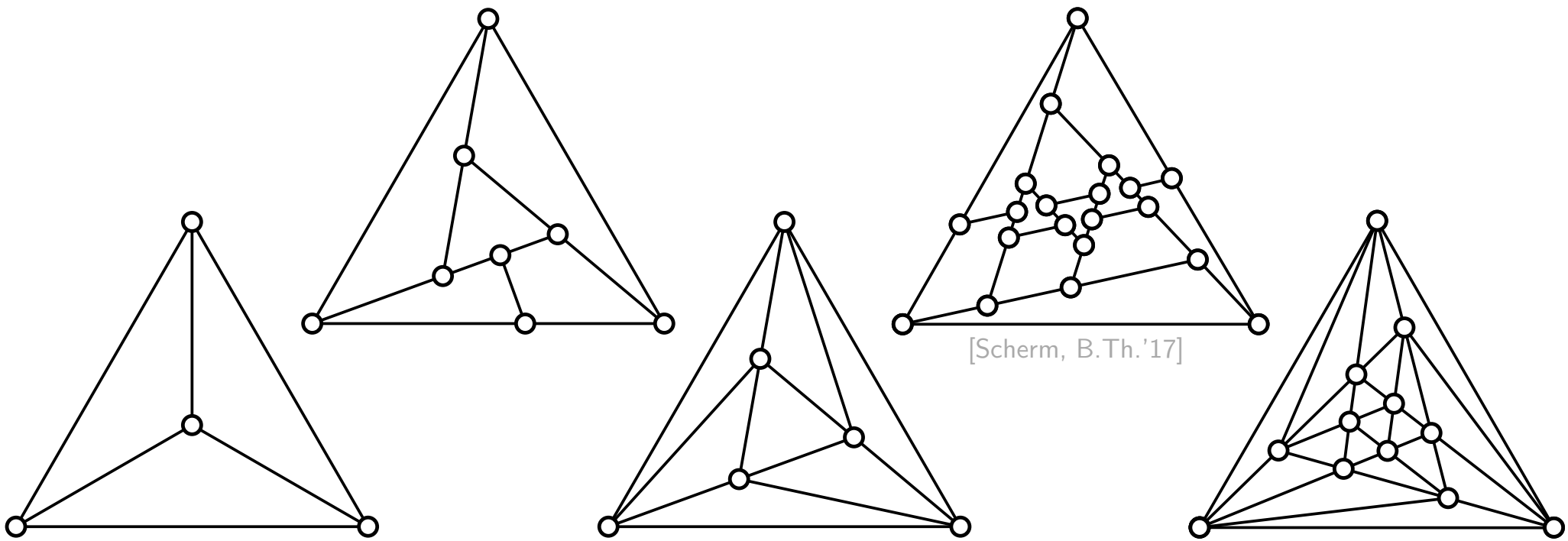
$G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4				
cube	8	12	6				
octahedron	6	12	8				
dodecahedron	20	30	12				
icosahedron	12	30	20				



Line Cover Numbers of the Platonic Solids

[Kryven et al., CALDAM'18]

$G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4	6	6		
cube	8	12	6	7	7		
octahedron	6	12	8	9	9		
dodecahedron	20	30	12	9...10	9...10		
icosahedron	12	30	20	13...15	13...15		

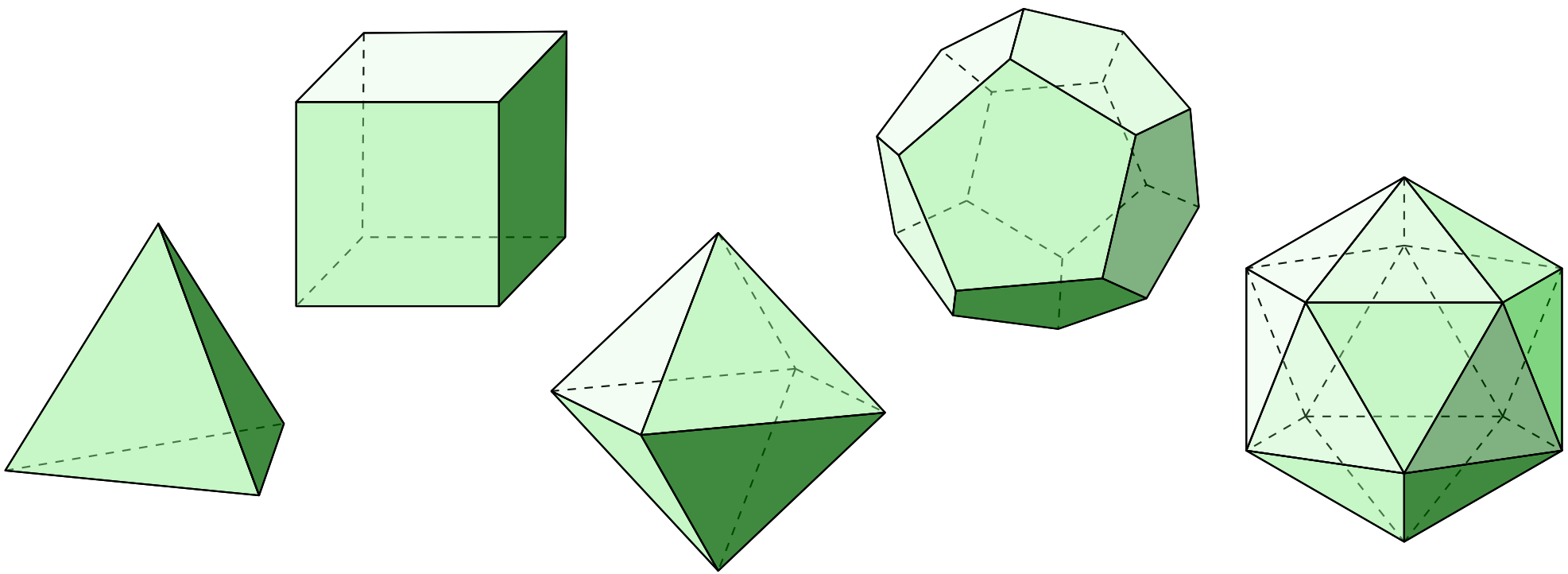


[Scherm, B.Th.'17]

Line Cover Numbers of the Platonic Solids

[Kryven et al., CALDAM'18] [Firman, MTh. '17]

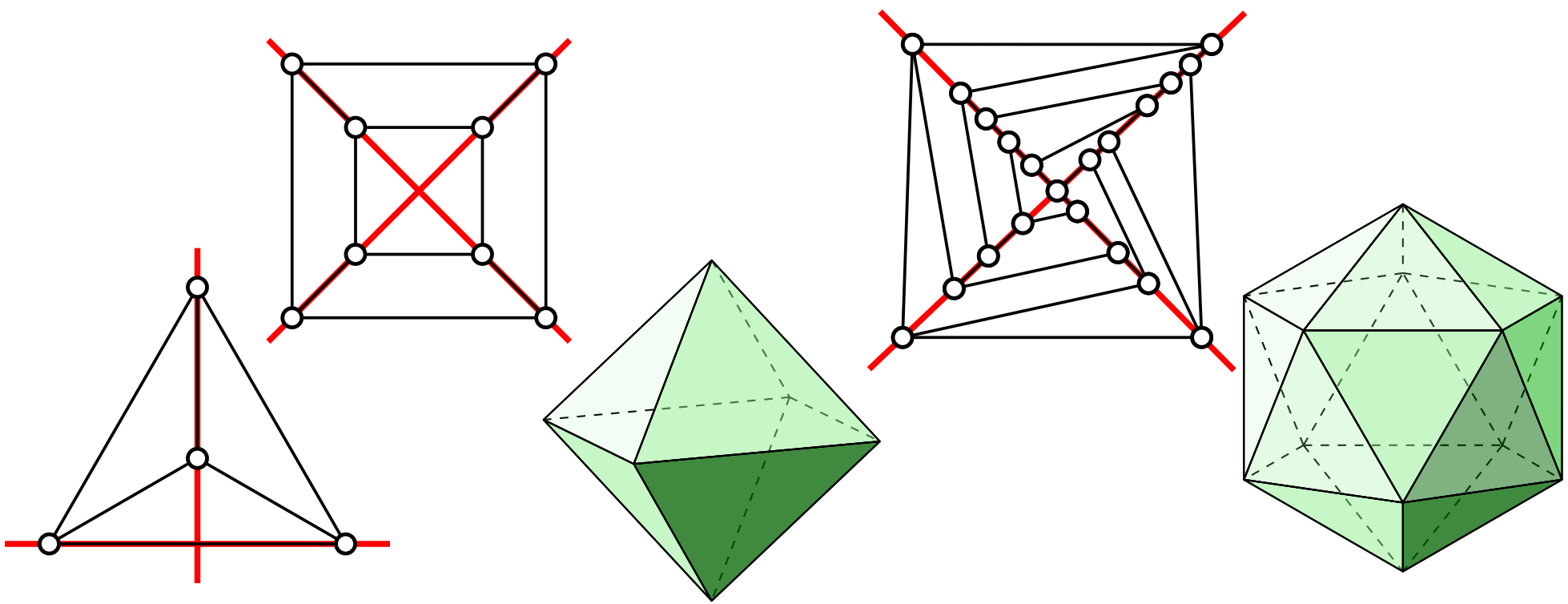
$G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4	6	6		
cube	8	12	6	7	7		
octahedron	6	12	8	9	9		
dodecahedron	20	30	12	9...10	9...10		
icosahedron	12	30	20	13...15	13...15		



Line Cover Numbers of the Platonic Solids

[Kryven et al., CALDAM'18] [Firman, MTh. '17]

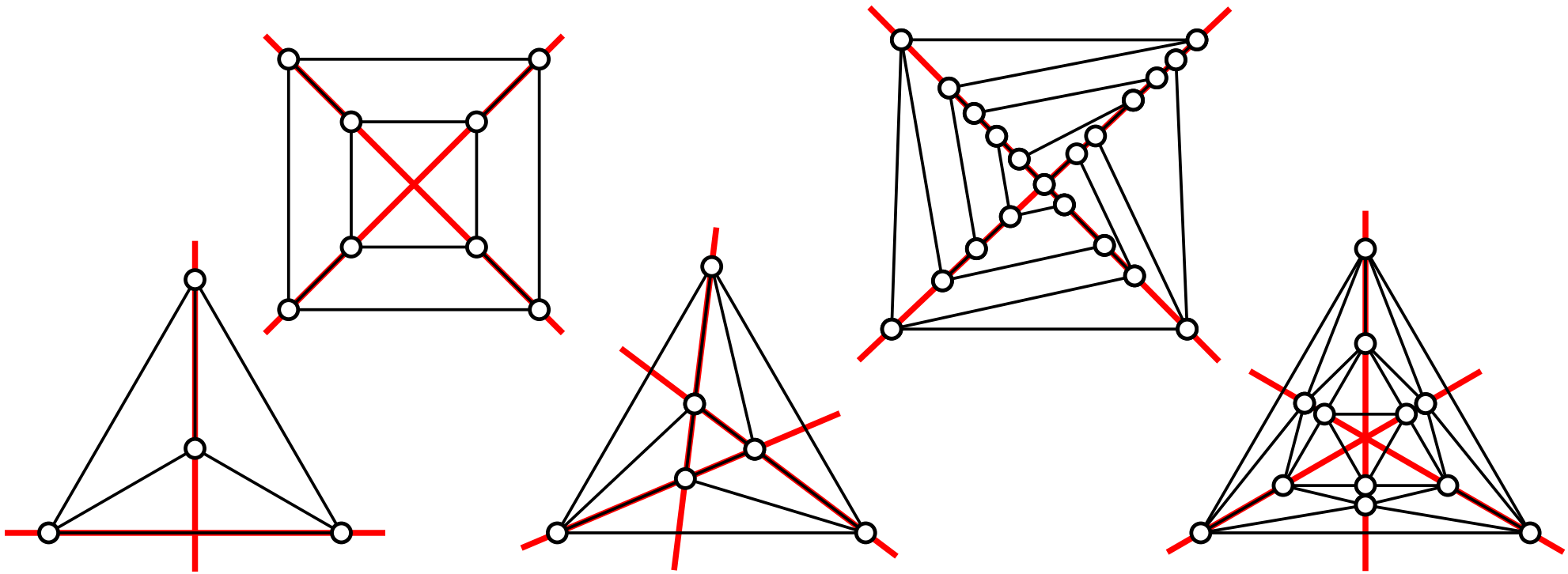
$G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4	6	6	2	
cube	8	12	6	7	7	2	
octahedron	6	12	8	9	9		
dodecahedron	20	30	12	9...10	9...10	2	
icosahedron	12	30	20	13...15	13...15		



Line Cover Numbers of the Platonic Solids

[Kryven et al., CALDAM'18] [Firman, MTh. '17]

$G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4	6	6	2	
cube	8	12	6	7	7	2	
octahedron	6	12	8	9	9	3	
dodecahedron	20	30	12	9...10	9...10	2	
icosahedron	12	30	20	13...15	13...15	3	



Line Cover Numbers of the Platonic Solids

[Kryven et al., CALDAM'18] [Firman, MTh. '17]

$G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4	6	6	2	
cube	8	12	6	7	7	2	
octahedron	6	12	8	9	9	3	
dodecahedron	20	30	12	9...10	9...10	2	
icosahedron	12	30	20	13...15	13...15	3	

Line Cover Numbers of the Platonic Solids

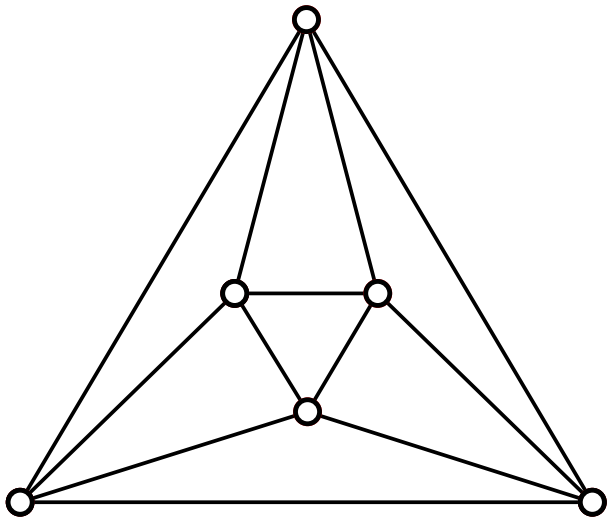
[Kryven et al., CALDAM'18] [Firman, MTh. '17]

$G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4	6	6	2	2
cube	8	12	6	7	7	2	2
octahedron	6	12	8	9	9	3	
dodecahedron	20	30	12	9...10	9...10	2	2
icosahedron	12	30	20	13...15	13...15	3	

Line Cover Numbers of the Platonic Solids

[Kryven et al., CALDAM'18] [Firman, MTh. '17]

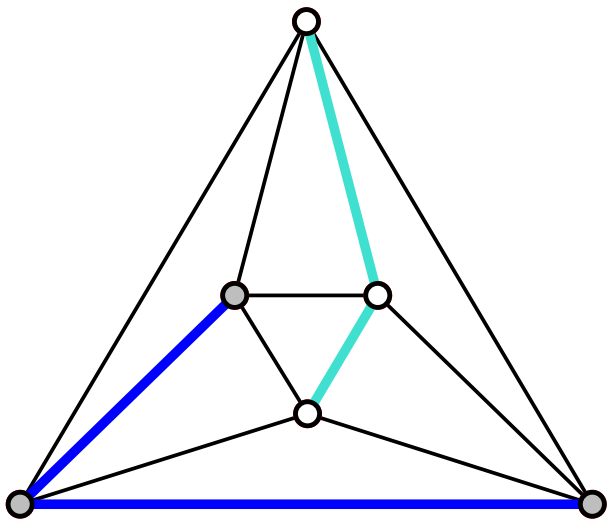
$G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4	6	6	2	2
cube	8	12	6	7	7	2	2
octahedron	6	12	8	9	9	3	
dodecahedron	20	30	12	9...10	9...10	2	2
icosahedron	12	30	20	13...15	13...15	3	



Line Cover Numbers of the Platonic Solids

[Kryven et al., CALDAM'18] [Firman, MTh. '17]

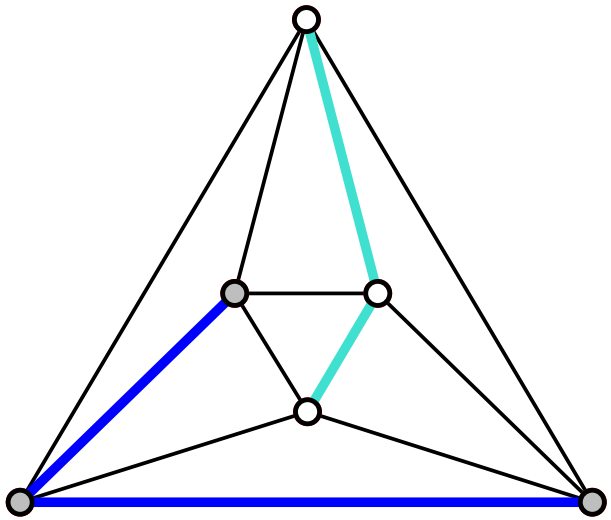
$G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4	6	6	2	2
cube	8	12	6	7	7	2	2
octahedron	6	12	8	9	9	3	2
dodecahedron	20	30	12	9...10	9...10	2	2
icosahedron	12	30	20	13...15	13...15	3	

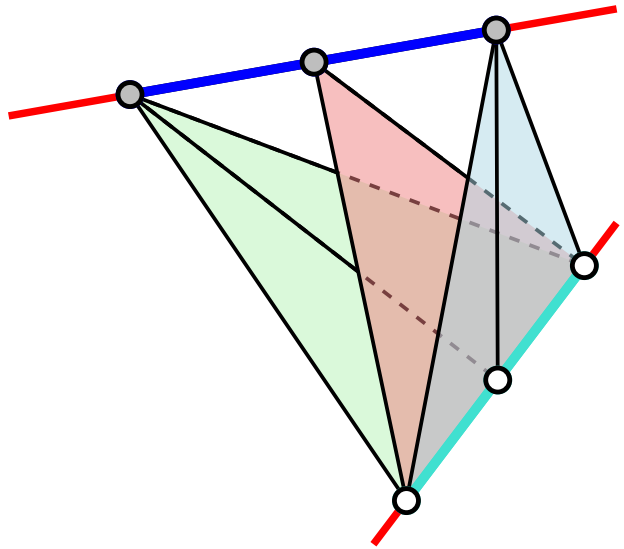


Line Cover Numbers of the Platonic Solids

[Kryven et al., CALDAM'18] [Firman, MTh. '17]

$G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4	6	6	2	2
cube	8	12	6	7	7	2	2
octahedron	6	12	8	9	9	3	2
dodecahedron	20	30	12	9...10	9...10	2	2
icosahedron	12	30	20	13...15	13...15	3	

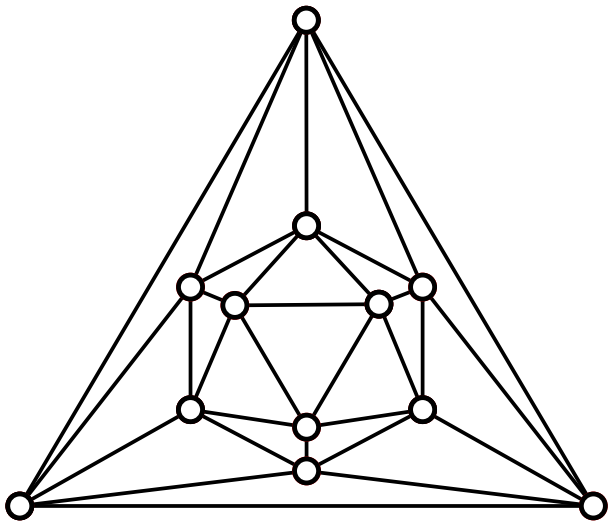




Line Cover Numbers of the Platonic Solids

[Kryven et al., CALDAM'18] [Firman, MTh. '17]

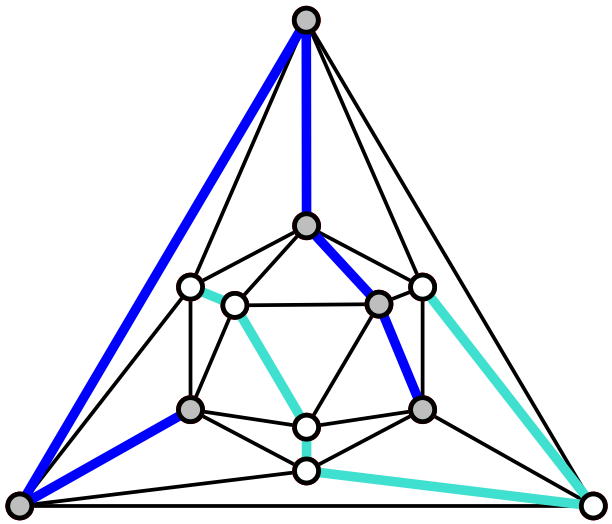
$G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4	6	6	2	2
cube	8	12	6	7	7	2	2
octahedron	6	12	8	9	9	3	2
dodecahedron	20	30	12	9...10	9...10	2	2
icosahedron	12	30	20	13...15	13...15	3	



Line Cover Numbers of the Platonic Solids

[Kryven et al., CALDAM'18] [Firman, MTh. '17]

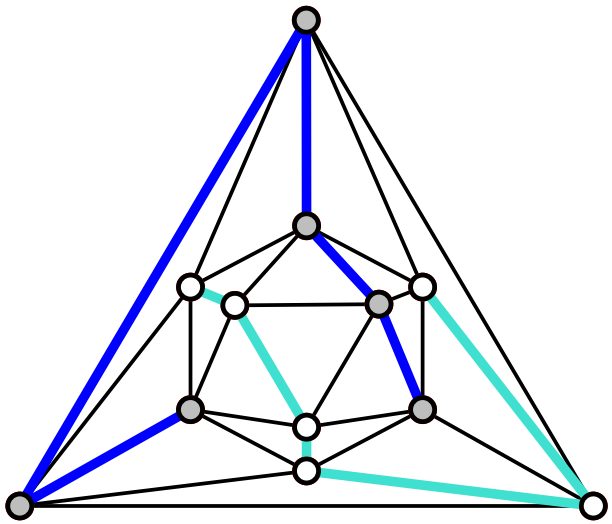
$G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4	6	6	2	2
cube	8	12	6	7	7	2	2
octahedron	6	12	8	9	9	3	2
dodecahedron	20	30	12	9...10	9...10	2	2
icosahedron	12	30	20	13...15	13...15	3	2

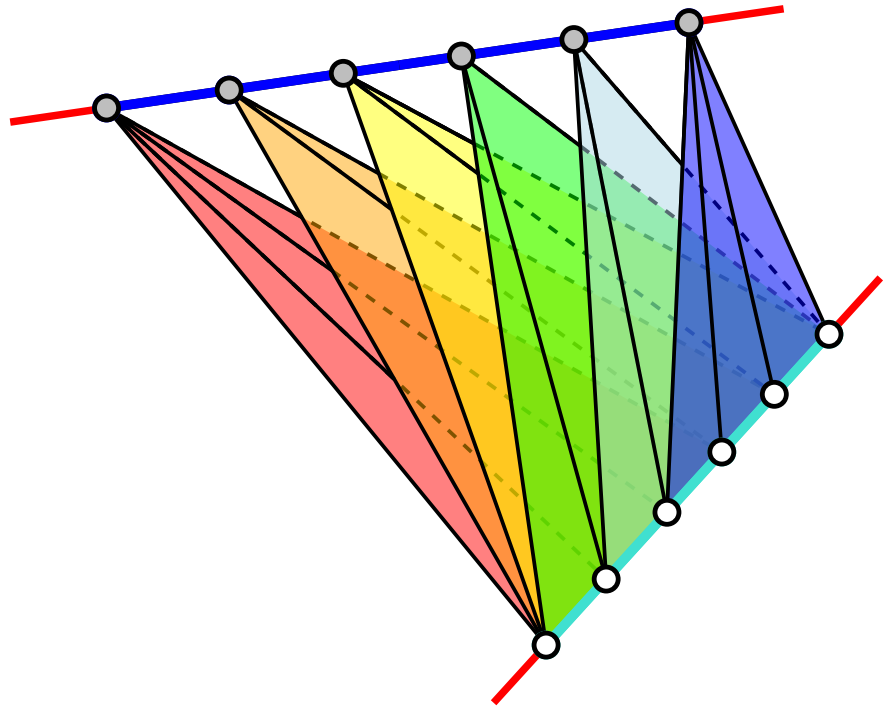


Line Cover Numbers of the Platonic Solids

[Kryven et al., CALDAM'18] [Firman, MTh. '17]

$G=(V,E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4	6	6	2	2
cube	8	12	6	7	7	2	2
octahedron	6	12	8	9	9	3	2
dodecahedron	20	30	12	9...10	9...10	2	2
icosahedron	12	30	20	13...15	13...15	3	2





Contact Representations in 3D

[Evans, Rzążewski, Saeedi, Shin, W., GD'19]

Many settings have been studied...

For example:

Contact Representations in 3D

[Evans, Rzążewski, Saeedi, Shin, W., GD'19]

Many settings have been studied...

We propose:

For example:

Contact Representations in 3D

[Evans, Rzążewski, Saeedi, Shin, W., GD'19]

Many settings have been studied...

We propose:

vertices $\rightarrow$

edges $\rightarrow$

For example:

Contact Representations in 3D

[Evans, Rzążewski, Saeedi, Shin, W., GD'19]

Many settings have been studied...

We propose:

vertices $\rightarrow$ flat convex polygons

edges $\rightarrow$

For example:

Contact Representations in 3D

[Evans, Rzążewski, Saeedi, Shin, W., GD'19]

Many settings have been studied...

We propose:

vertices $\rightarrow$ flat convex polygons

edges $\rightarrow$ vertex contacts

For example:

Contact Representations in 3D

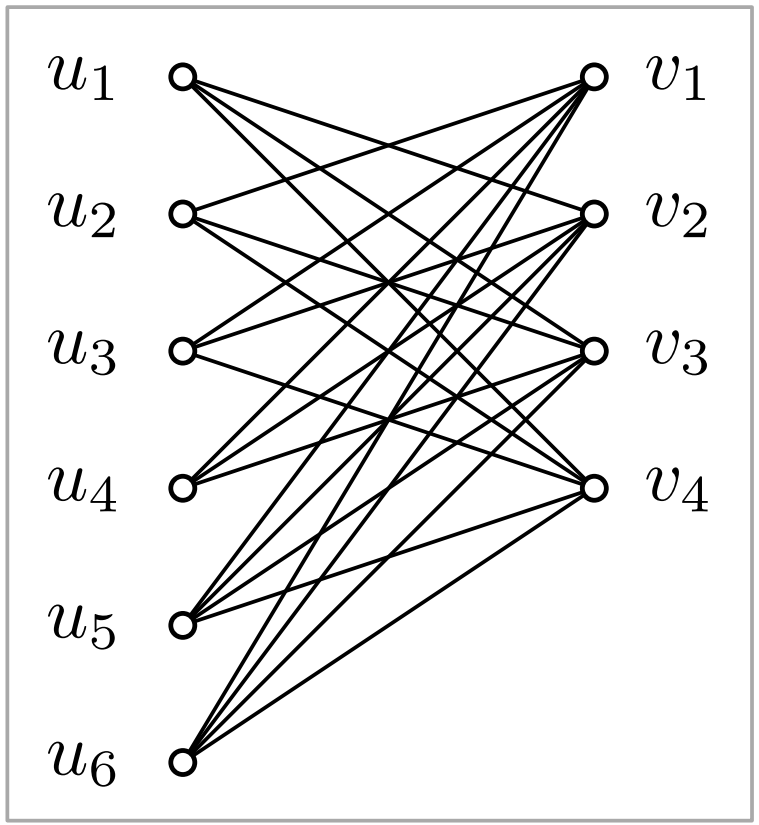
[Evans, Rzążewski, Saeedi, Shin, W., GD'19]

Many settings have been studied. . . We propose:

vertices $\rightarrow$ flat convex polygons

edges $\rightarrow$ vertex contacts

For example:



Contact Representations in 3D

[Evans, Rzążewski, Saeedi, Shin, W., GD'19]

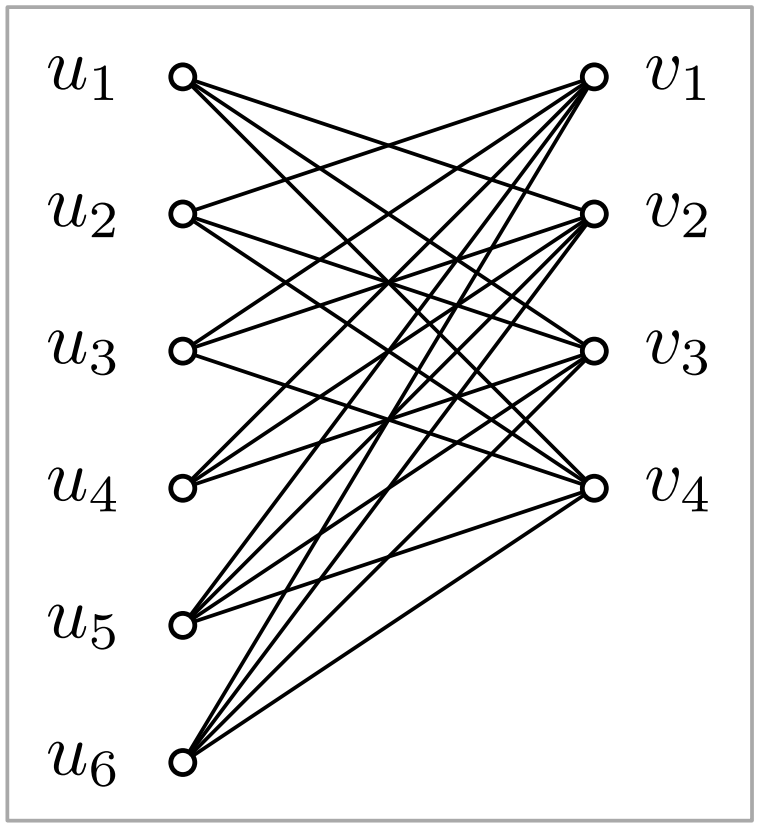
Many settings have been studied...

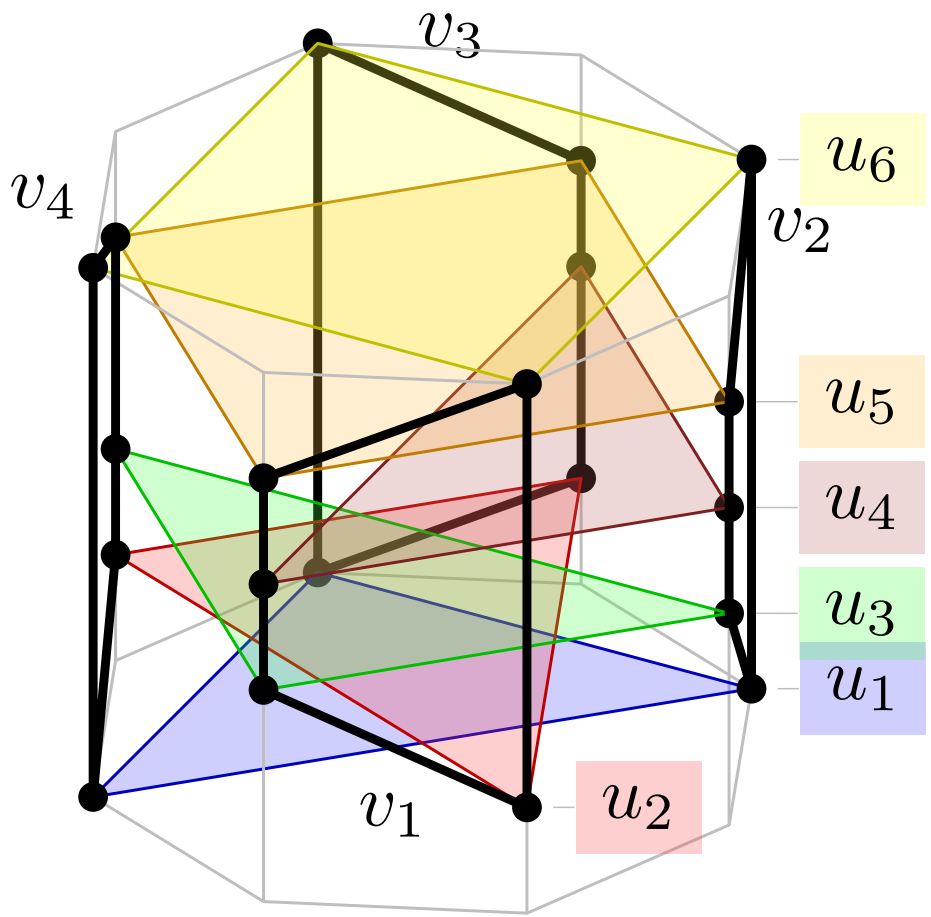
We propose:

vertices $\rightarrow$ flat convex polygons

edges $\rightarrow$ vertex contacts

For example:





Any Graph Admits such a Contact Representation

We just need two ingredients:

Any Graph Admits such a Contact Representation

We just need two ingredients:

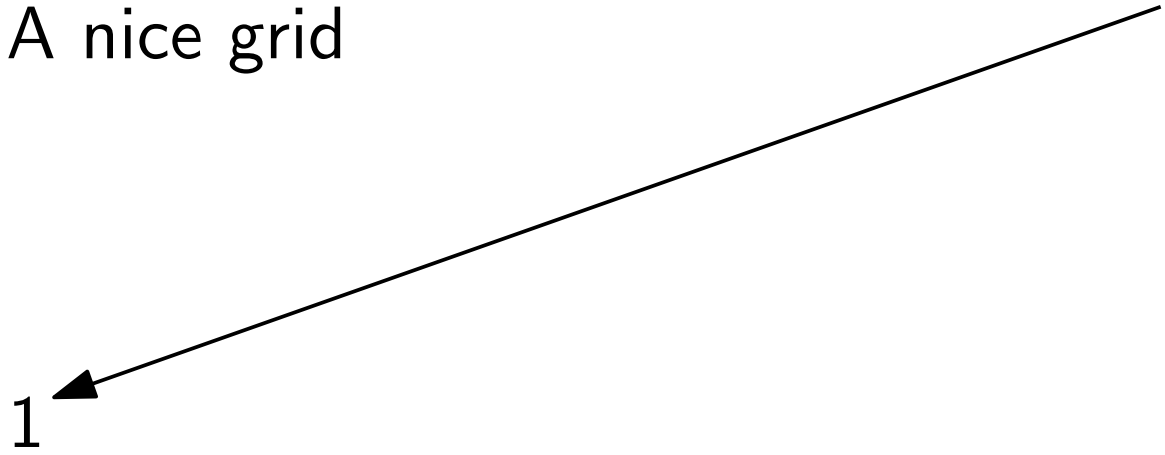
1. A nice grid

2. Nice vertical polygons

Any Graph Admits such a Contact Representation

We just need two ingredients:

1. A nice grid

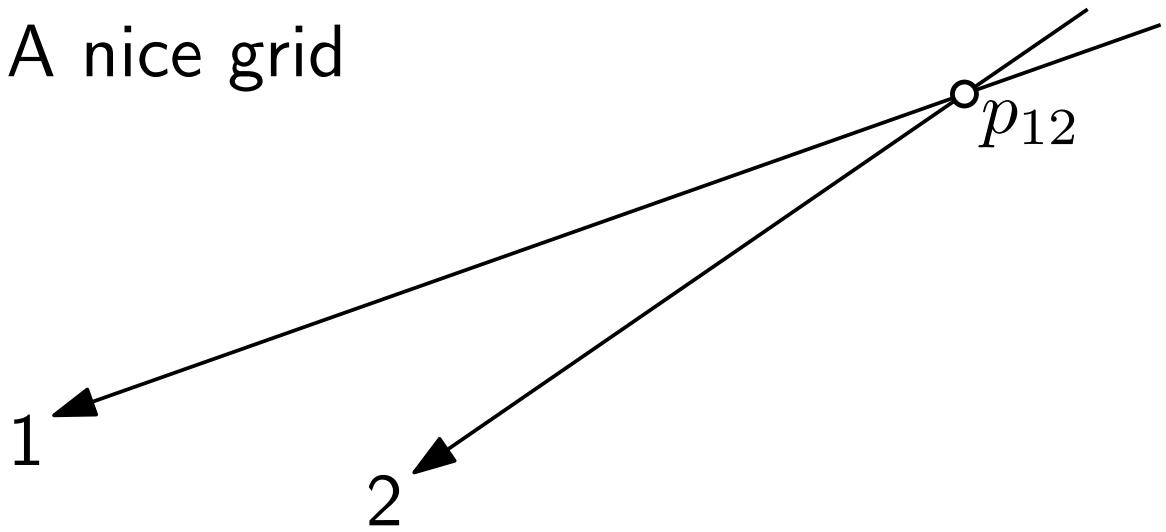


2. Nice vertical polygons

Any Graph Admits such a Contact Representation

We just need two ingredients:

1. A nice grid

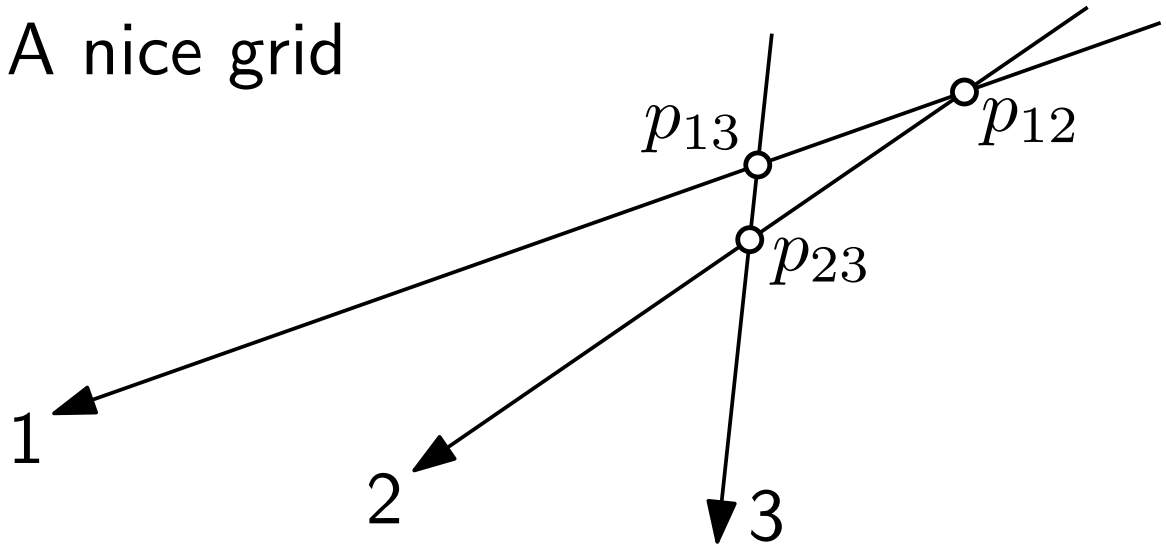


2. Nice vertical polygons

Any Graph Admits such a Contact Representation

We just need two ingredients:

1. A nice grid

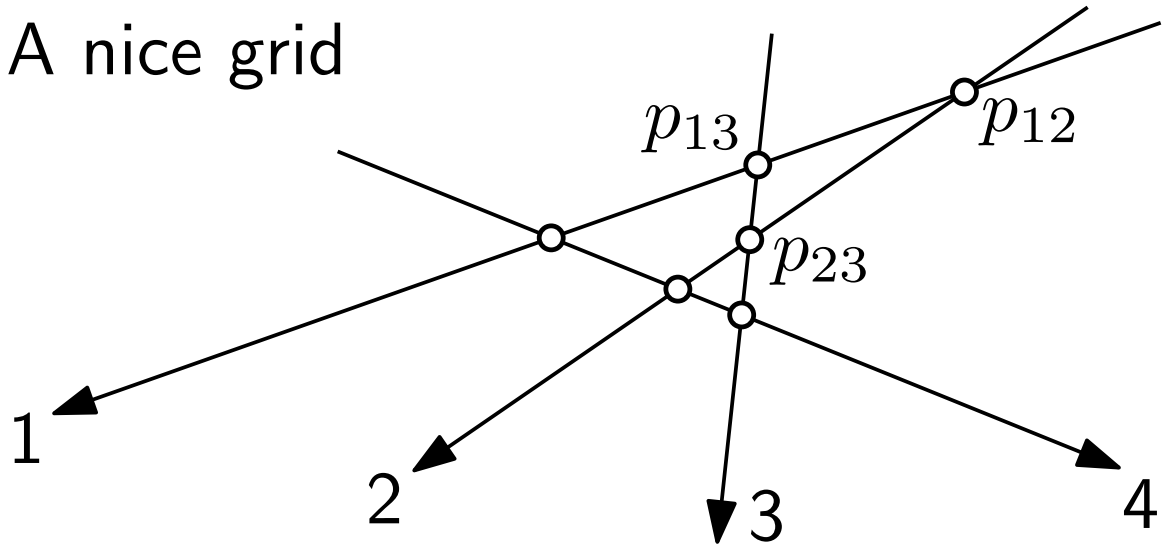


2. Nice vertical polygons

Any Graph Admits such a Contact Representation

We just need two ingredients:

1. A nice grid

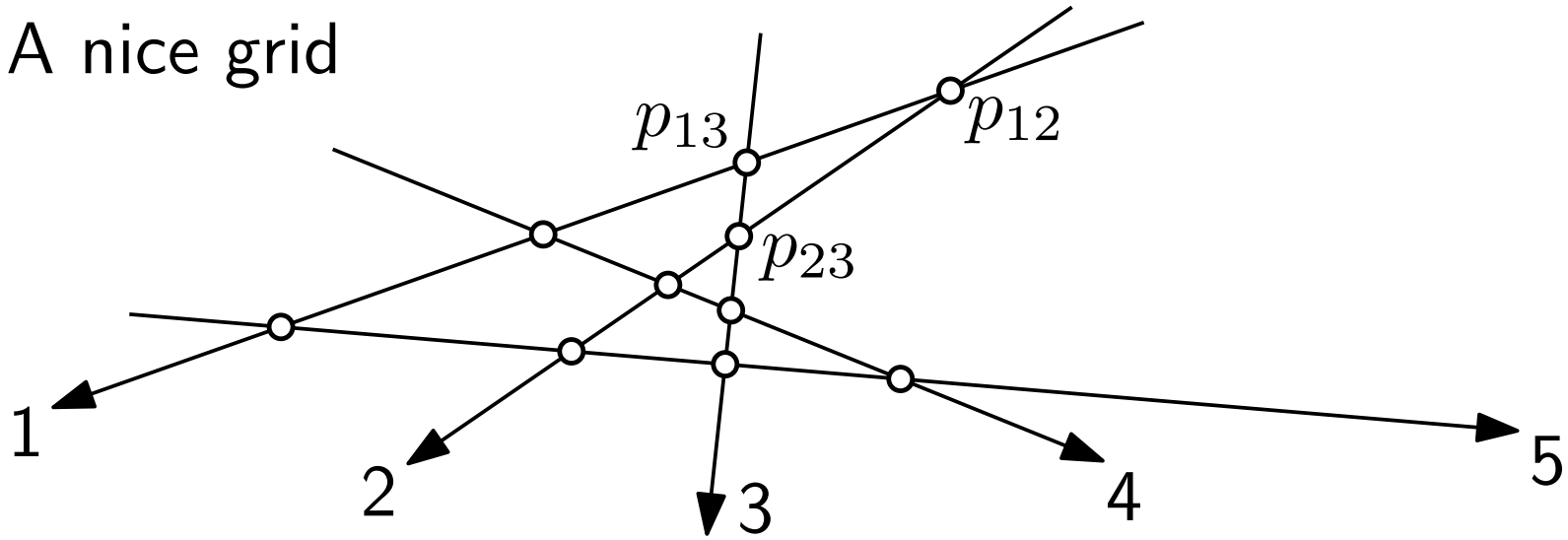


2. Nice vertical polygons

Any Graph Admits such a Contact Representation

We just need two ingredients:

1. A nice grid

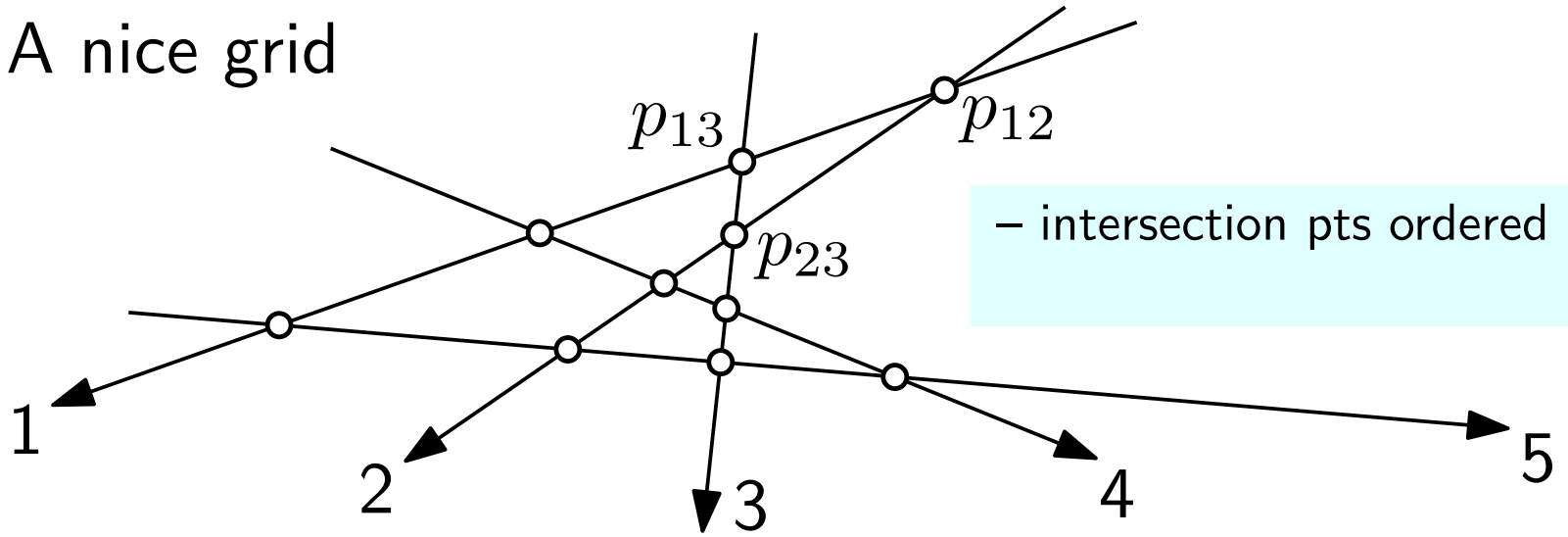


2. Nice vertical polygons

Any Graph Admits such a Contact Representation

We just need two ingredients:

1. A nice grid



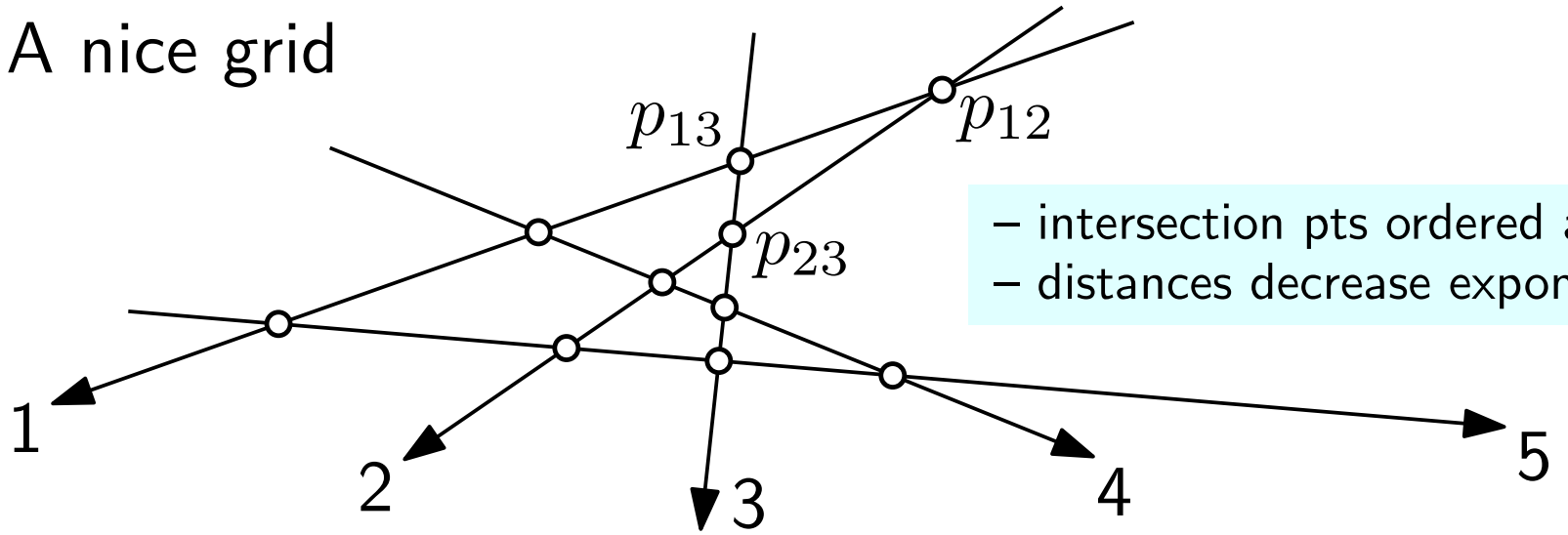
– intersection pts ordered along each line

2. Nice vertical polygons

Any Graph Admits such a Contact Representation

We just need two ingredients:

1. A nice grid



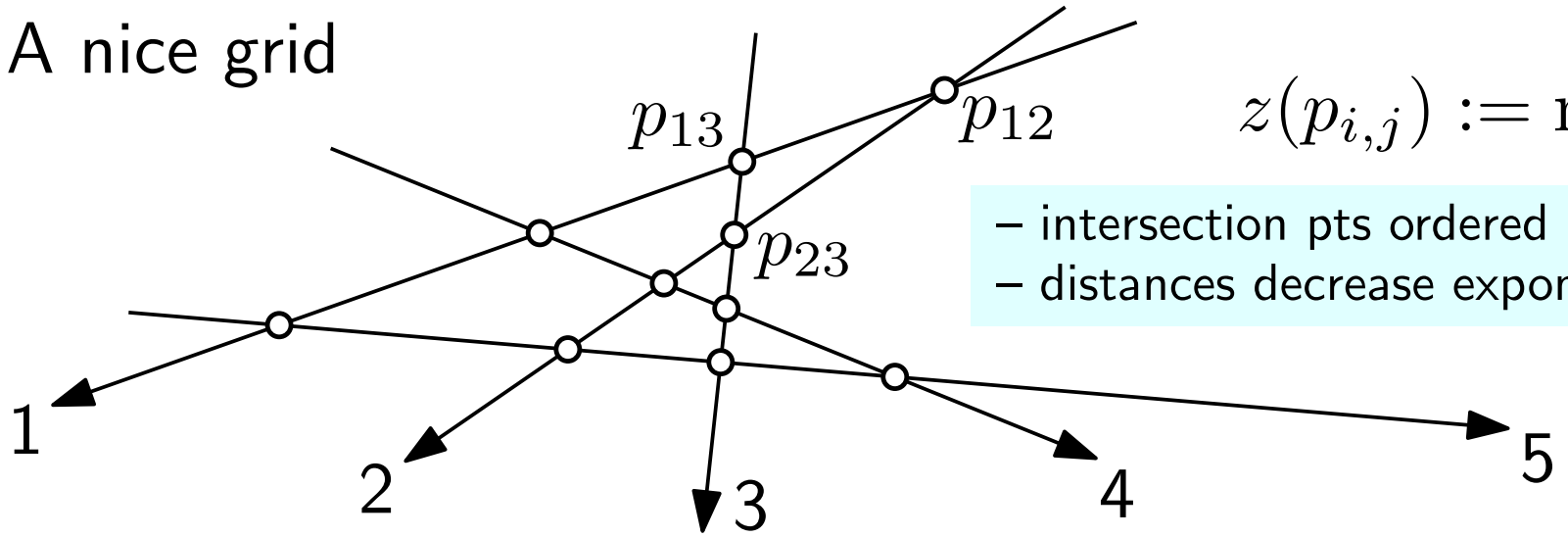
– intersection pts ordered along each line
– distances decrease exponentially

2. Nice vertical polygons

Any Graph Admits such a Contact Representation

We just need two ingredients:

1. A nice grid



$z(p_{i,j}) := \min\{i, j\}$

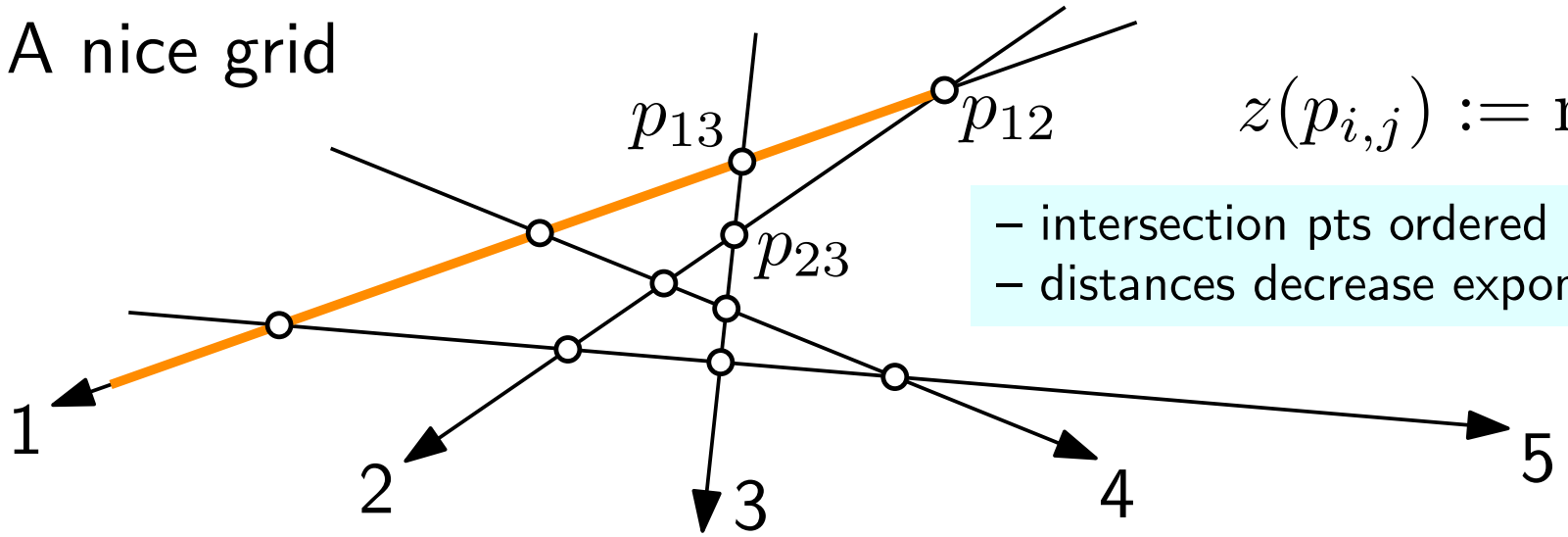
– intersection pts ordered along each line
– distances decrease exponentially

2. Nice vertical polygons

Any Graph Admits such a Contact Representation

We just need two ingredients:

1. A nice grid



$z(p_{i,j}) := \min\{i, j\}$

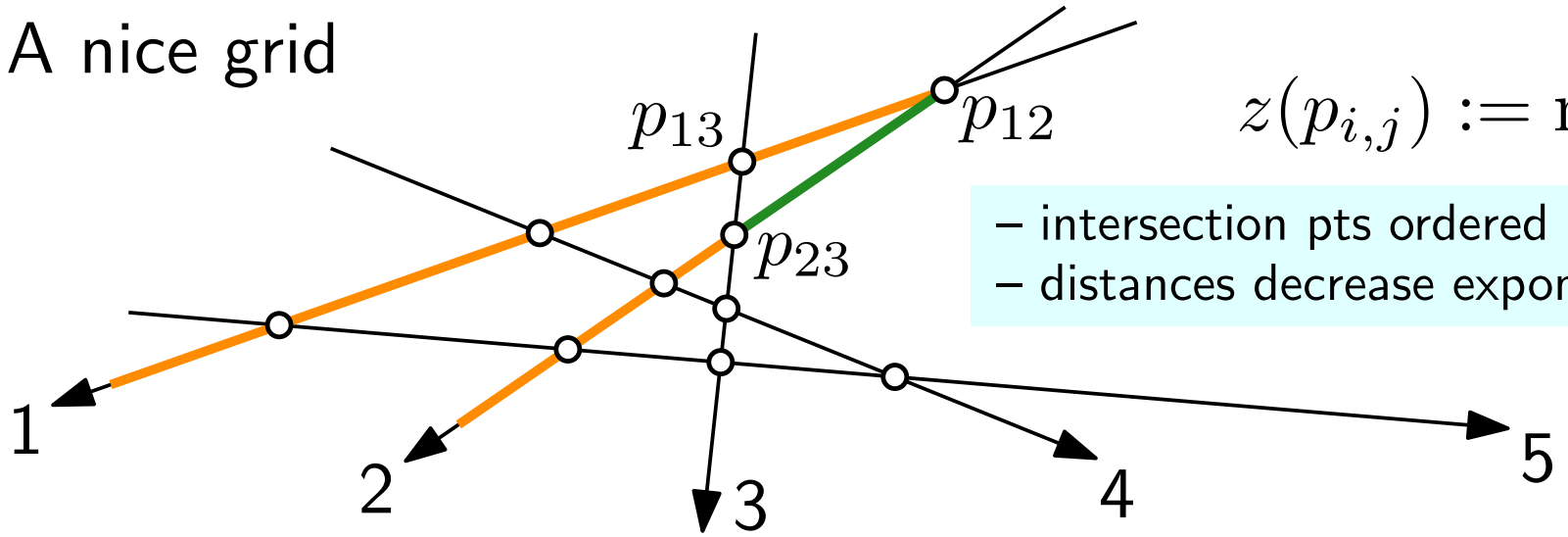
– intersection pts ordered along each line
– distances decrease exponentially

2. Nice vertical polygons

Any Graph Admits such a Contact Representation

We just need two ingredients:

1. A nice grid



$z(p_{i,j}) := \min\{i, j\}$

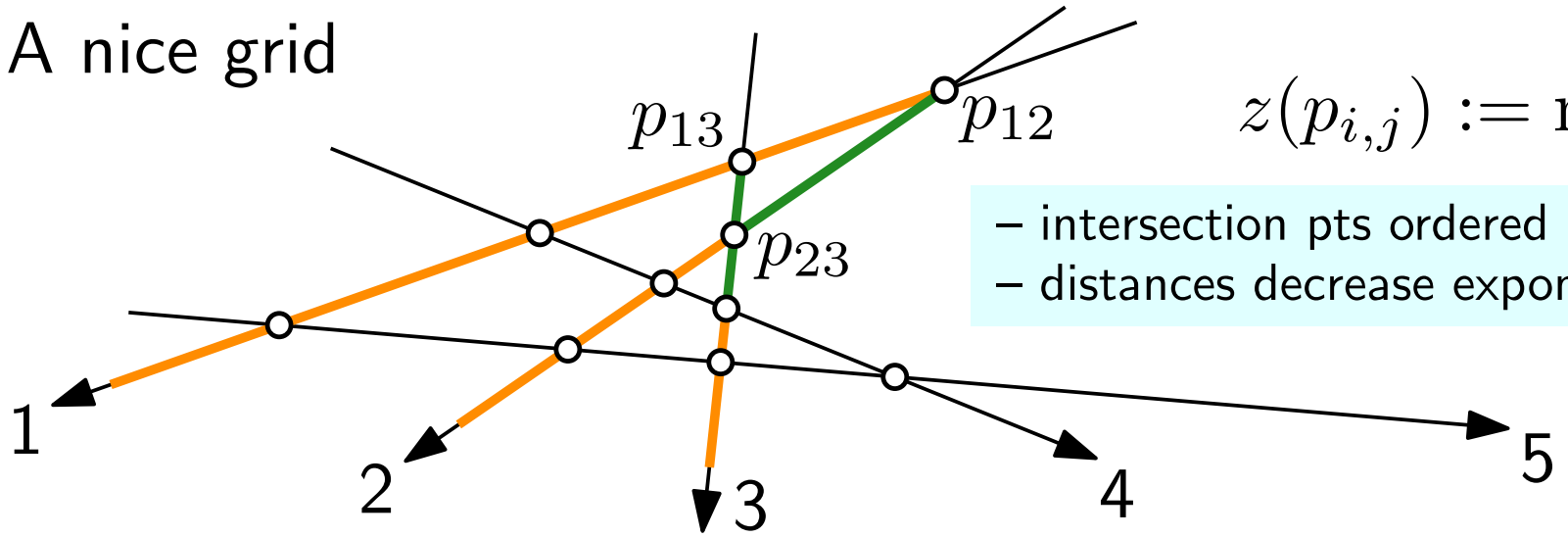
– intersection pts ordered along each line
– distances decrease exponentially

2. Nice vertical polygons

Any Graph Admits such a Contact Representation

We just need two ingredients:

1. A nice grid



$z(p_{i,j}) := \min\{i, j\}$

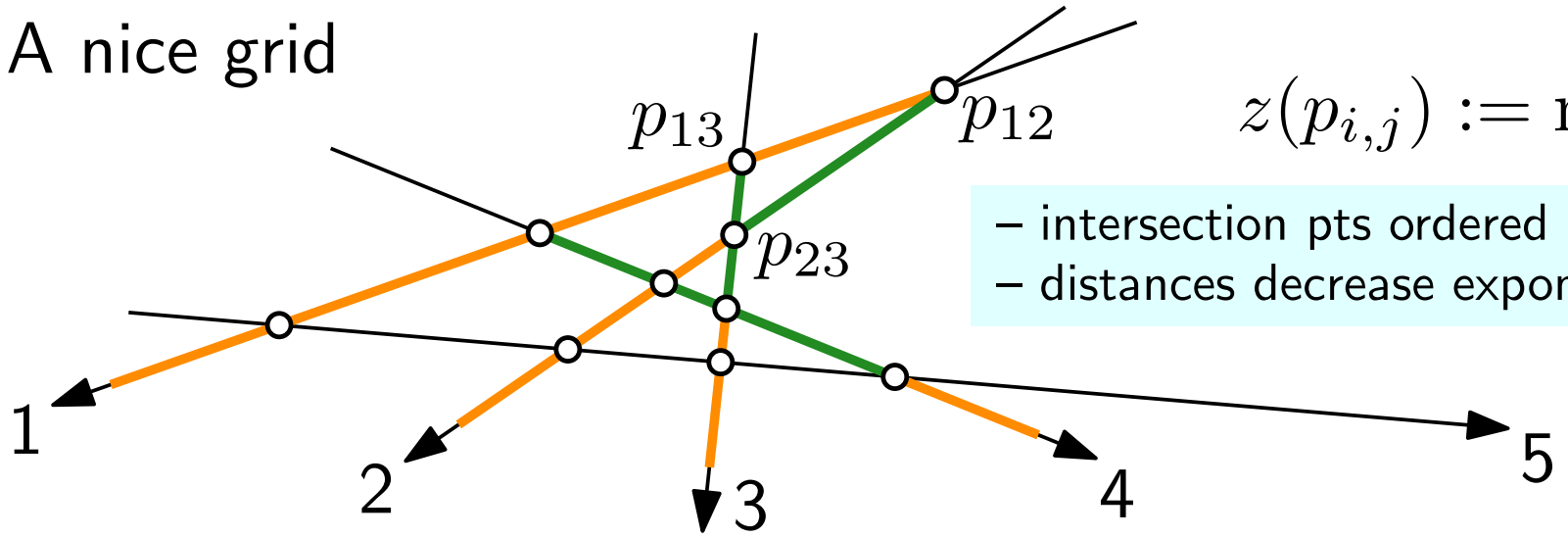
– intersection pts ordered along each line
– distances decrease exponentially

2. Nice vertical polygons

Any Graph Admits such a Contact Representation

We just need two ingredients:

1. A nice grid



$z(p_{i,j}) := \min\{i, j\}$

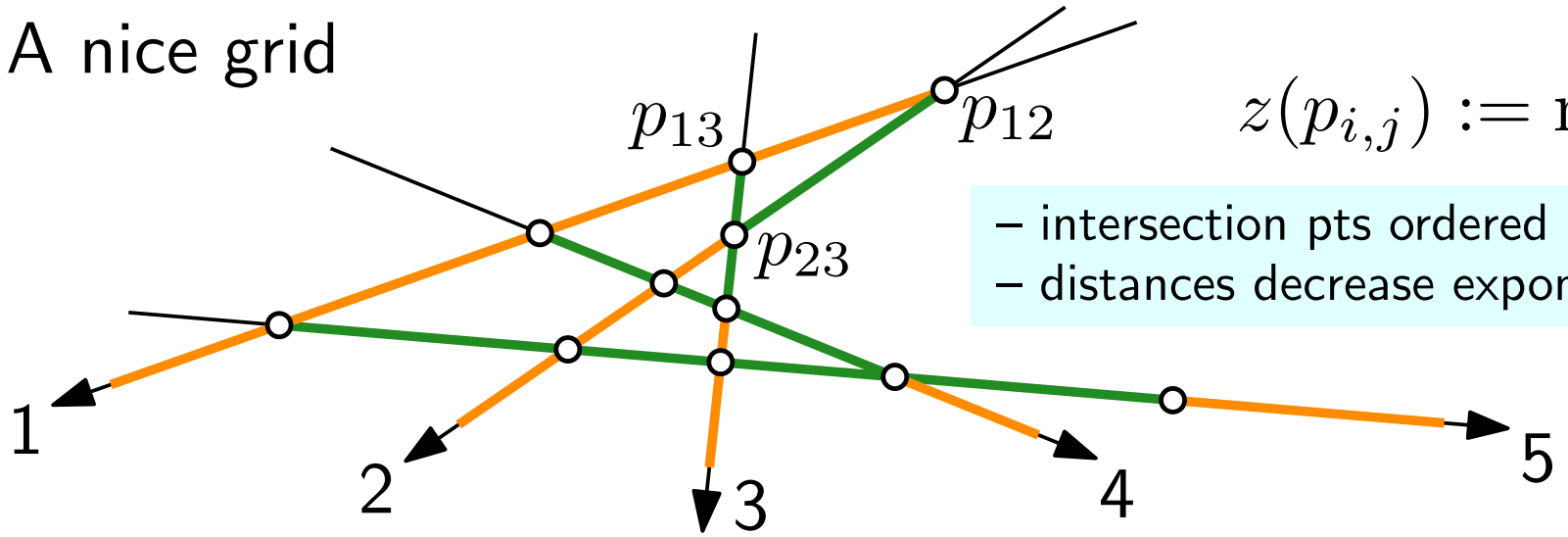
– intersection pts ordered along each line
– distances decrease exponentially

2. Nice vertical polygons

Any Graph Admits such a Contact Representation

We just need two ingredients:

1. A nice grid



$z(p_{i,j}) := \min\{i, j\}$

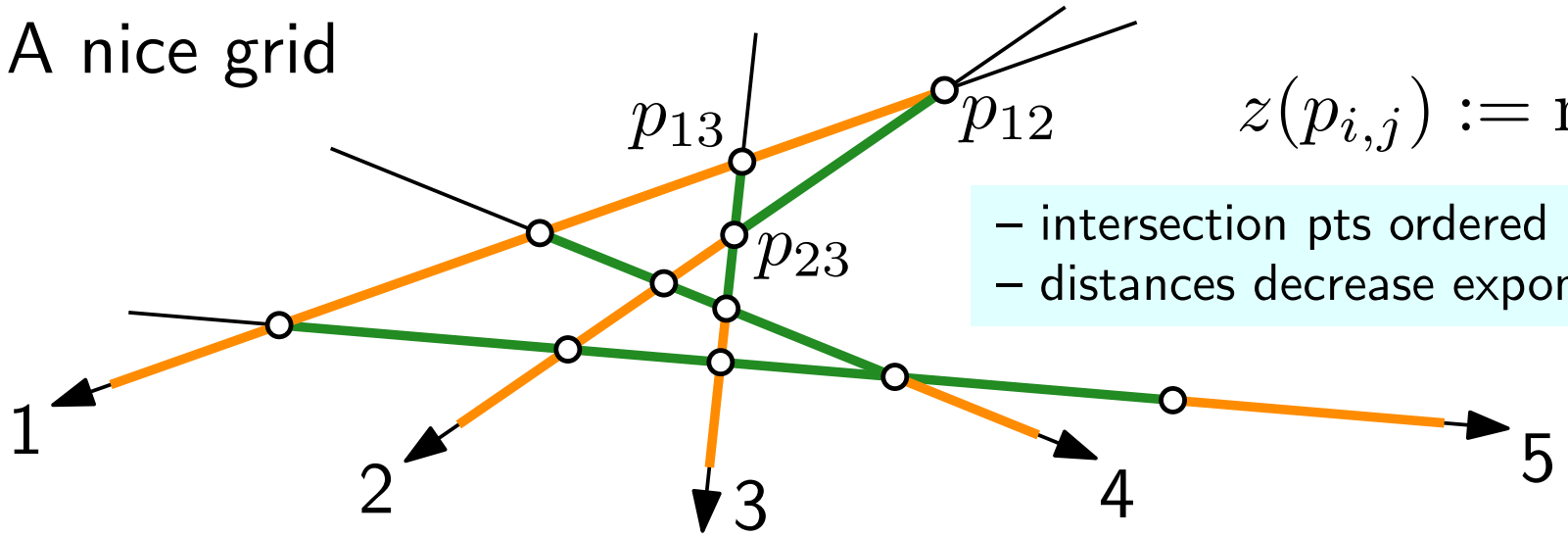
– intersection pts ordered along each line
– distances decrease exponentially

2. Nice vertical polygons

Any Graph Admits such a Contact Representation

We just need two ingredients:

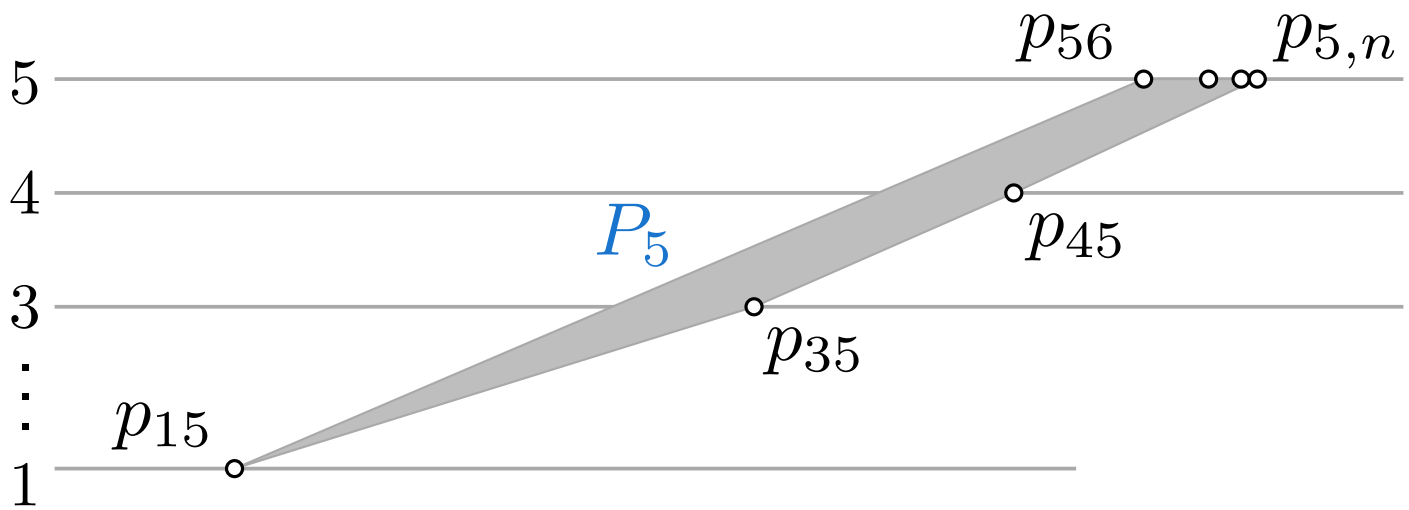
1. A nice grid



$$z(p_{i,j}) := \min\{i, j\}$$

– intersection pts ordered along each line
– distances decrease exponentially

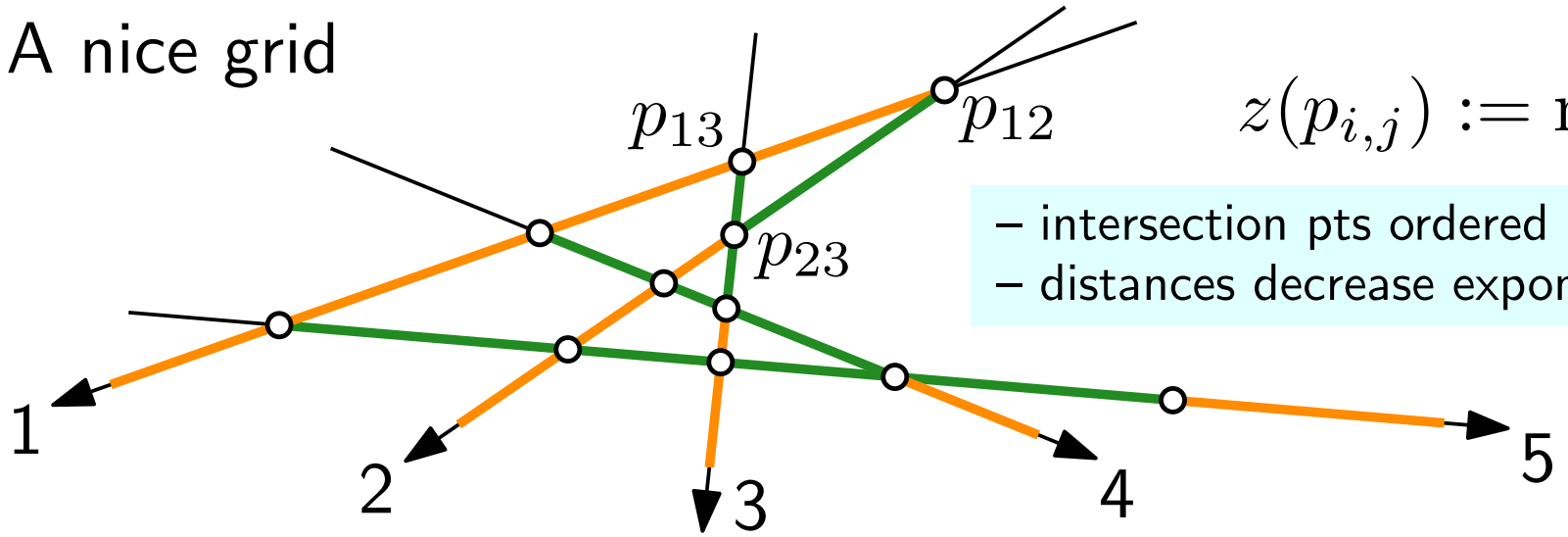
2. Nice vertical polygons



Any Graph Admits such a Contact Representation

We just need two ingredients:

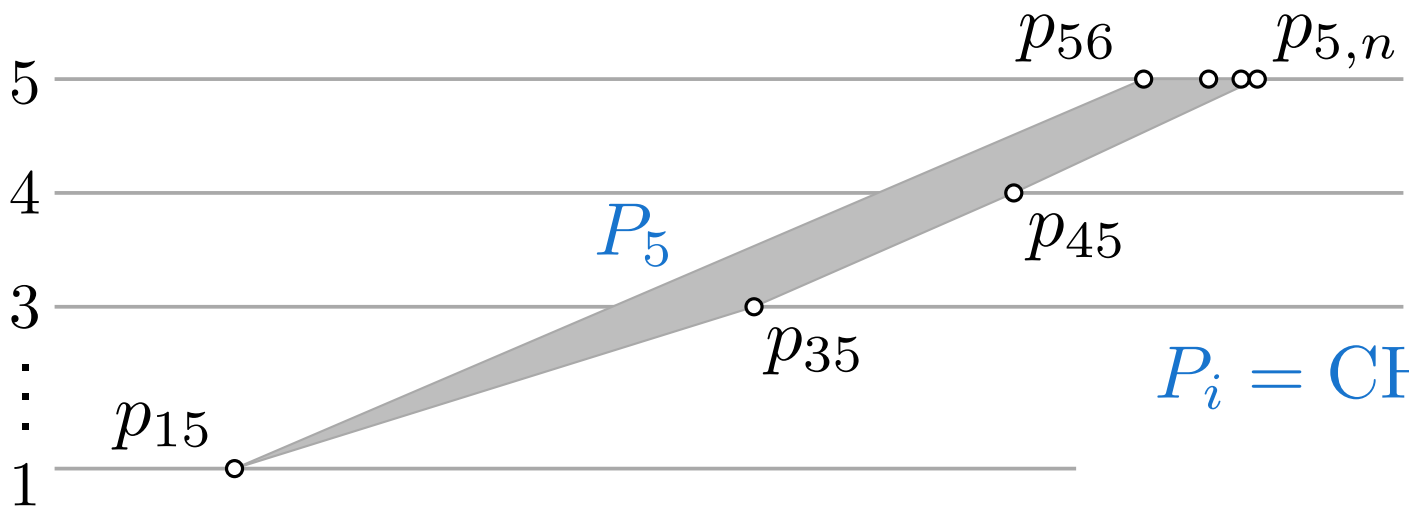
1. A nice grid



$z(p_{i,j}) := \min\{i, j\}$

– intersection pts ordered along each line
– distances decrease exponentially

2. Nice vertical polygons

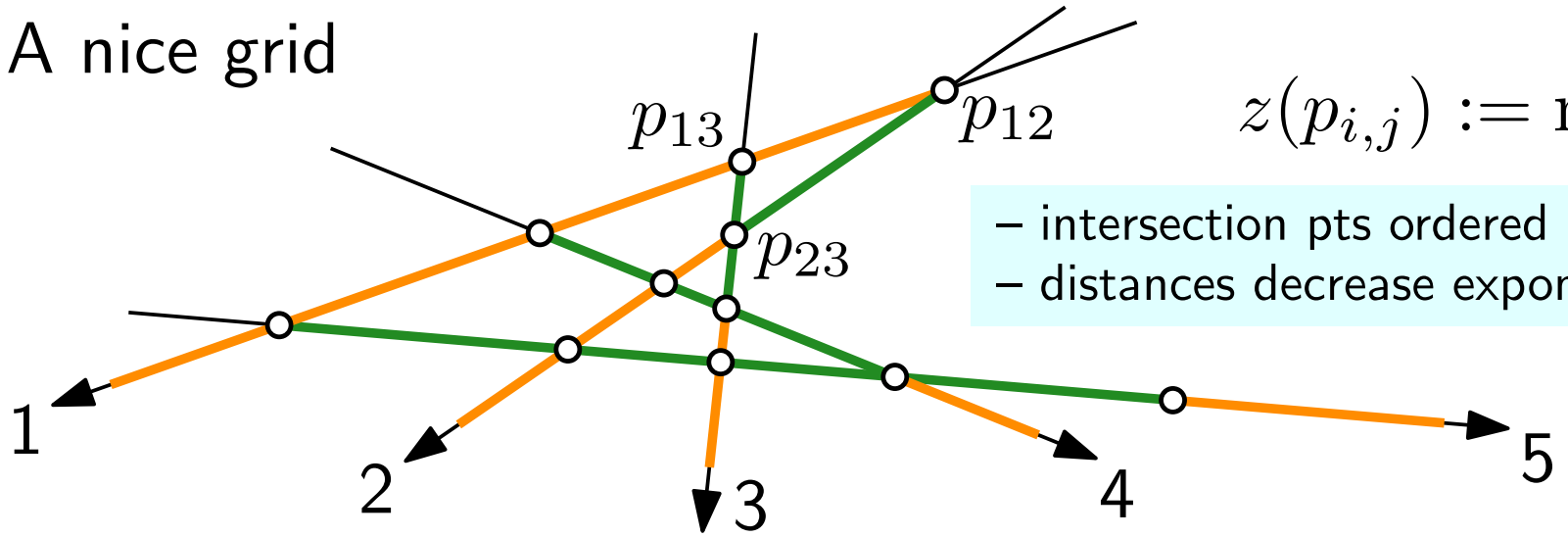


$P_i = \mathrm{CH}(\{p_{ij} : i \neq j\})$

Any Graph Admits such a Contact Representation

We just need two ingredients:

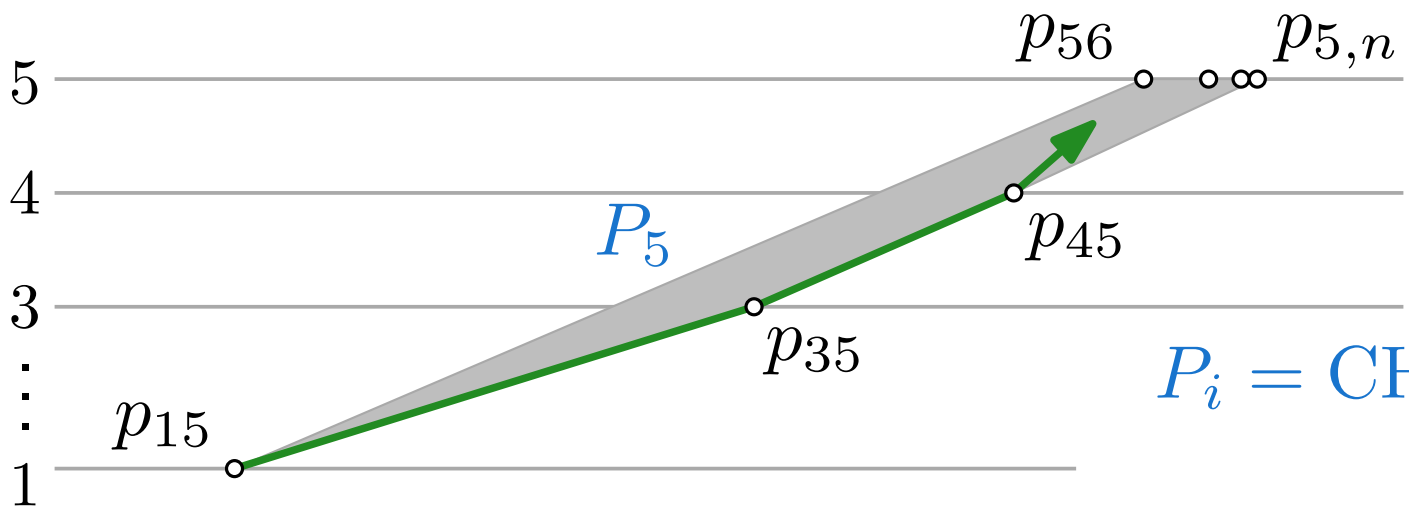
1. A nice grid



$z(p_{i,j}) := \min\{i, j\}$

– intersection pts ordered along each line
– distances decrease exponentially

2. Nice vertical polygons

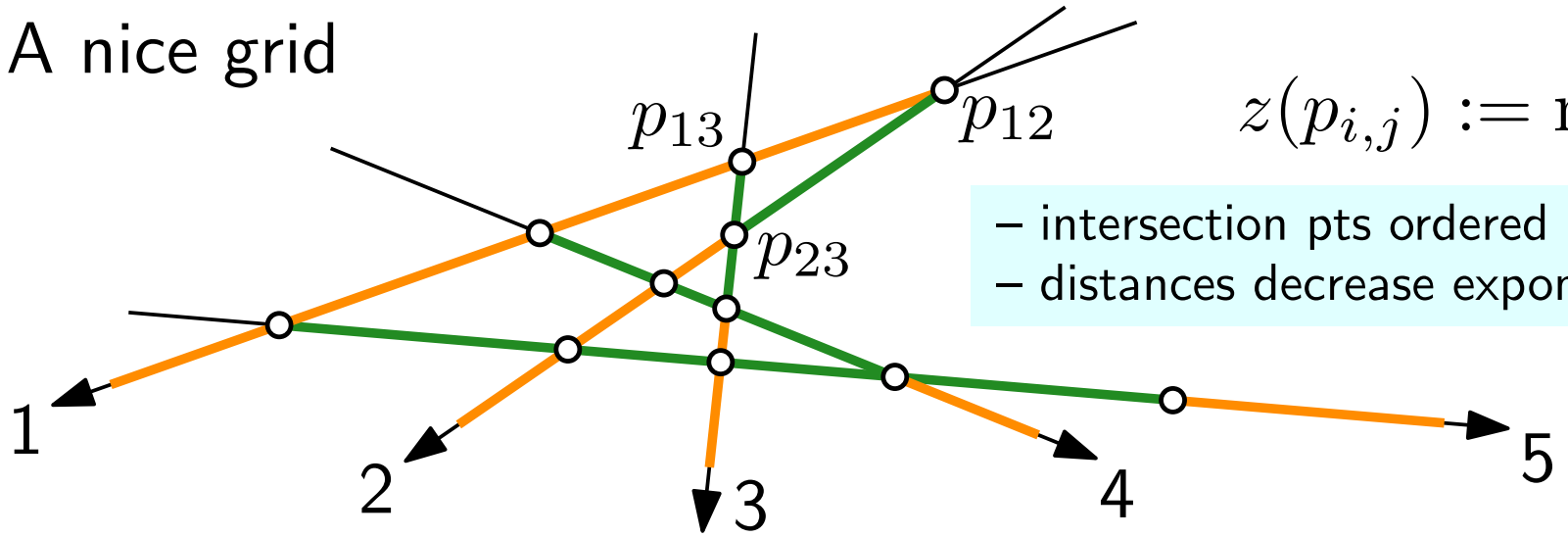


$P_i = \mathrm{CH}(\{p_{ij} : i \neq j\})$

Any Graph Admits such a Contact Representation

We just need two ingredients:

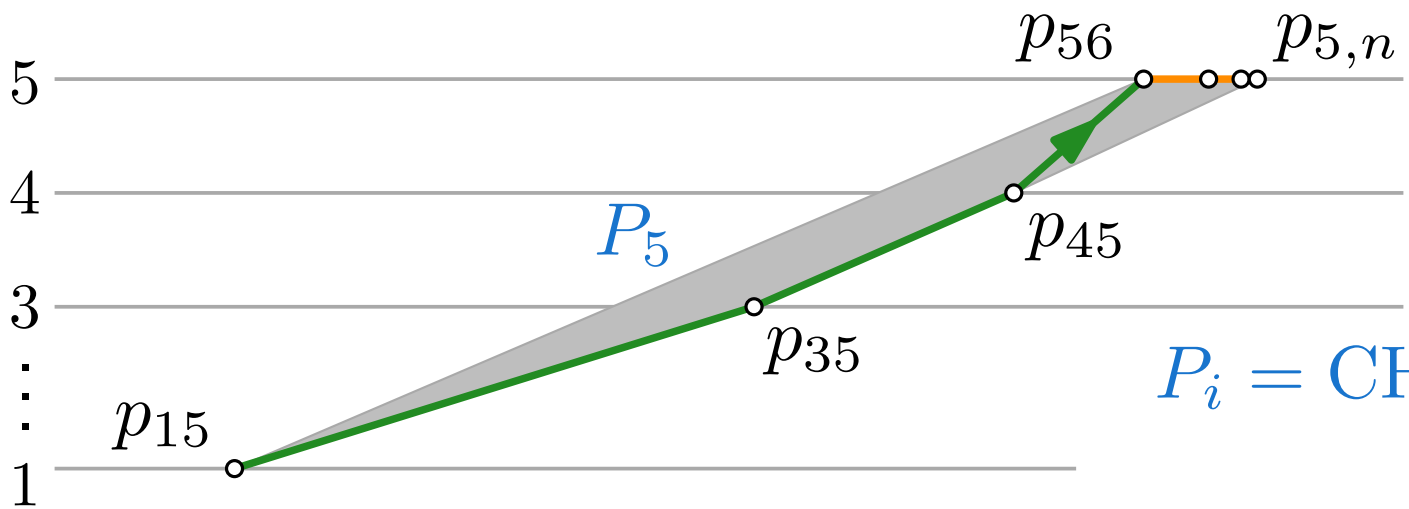
1. A nice grid



$z(p_{i,j}) := \min\{i, j\}$

– intersection pts ordered along each line
– distances decrease exponentially

2. Nice vertical polygons

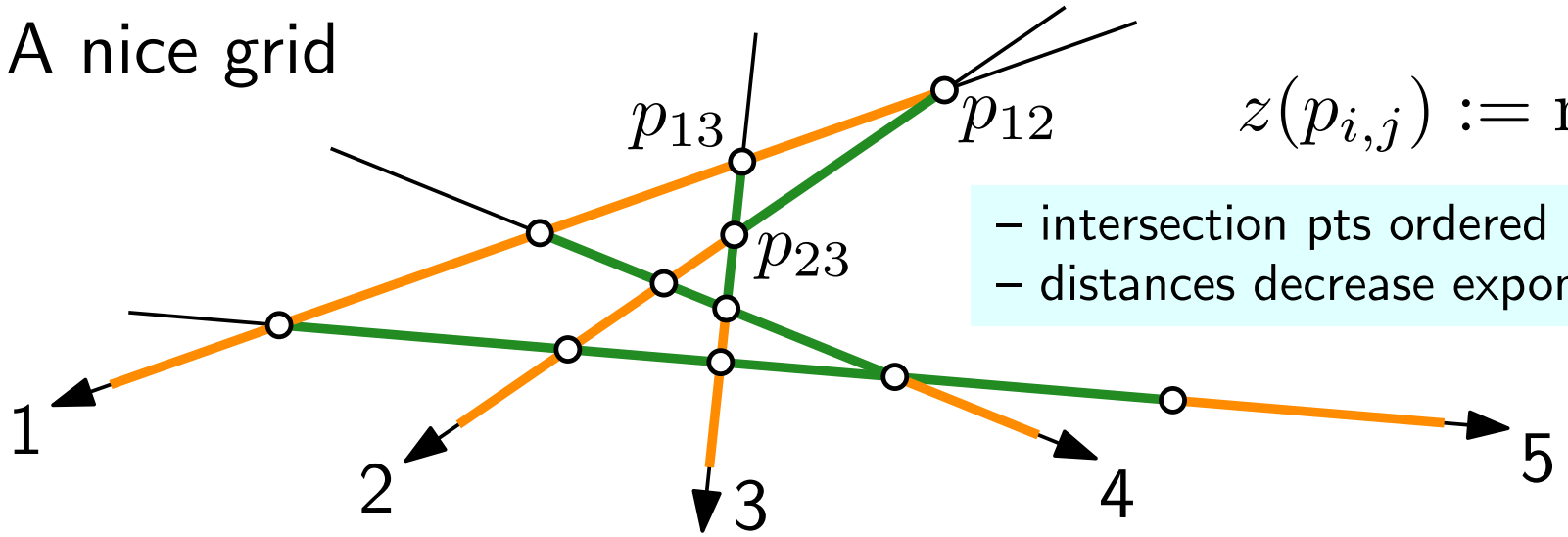


$P_i = \mathrm{CH}(\{p_{ij} : i \neq j\})$

Any Graph Admits such a Contact Representation

We just need two ingredients:

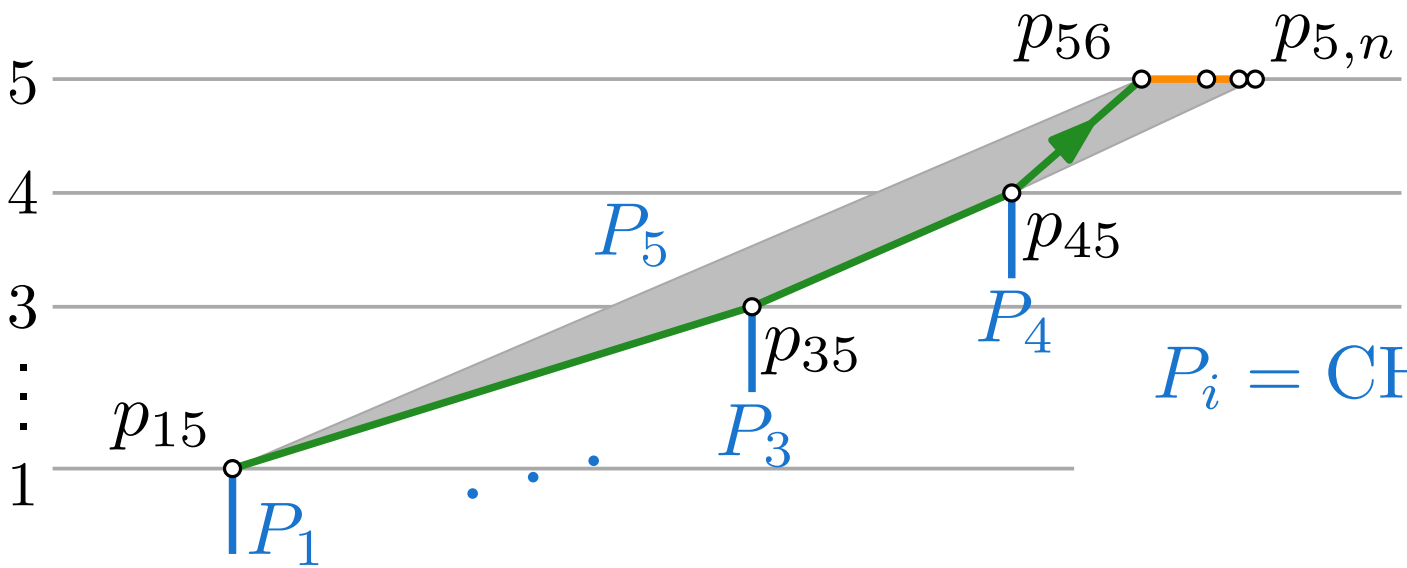
1. A nice grid



$z(p_{i,j}) := \min\{i, j\}$

– intersection pts ordered along each line
– distances decrease exponentially

2. Nice vertical polygons

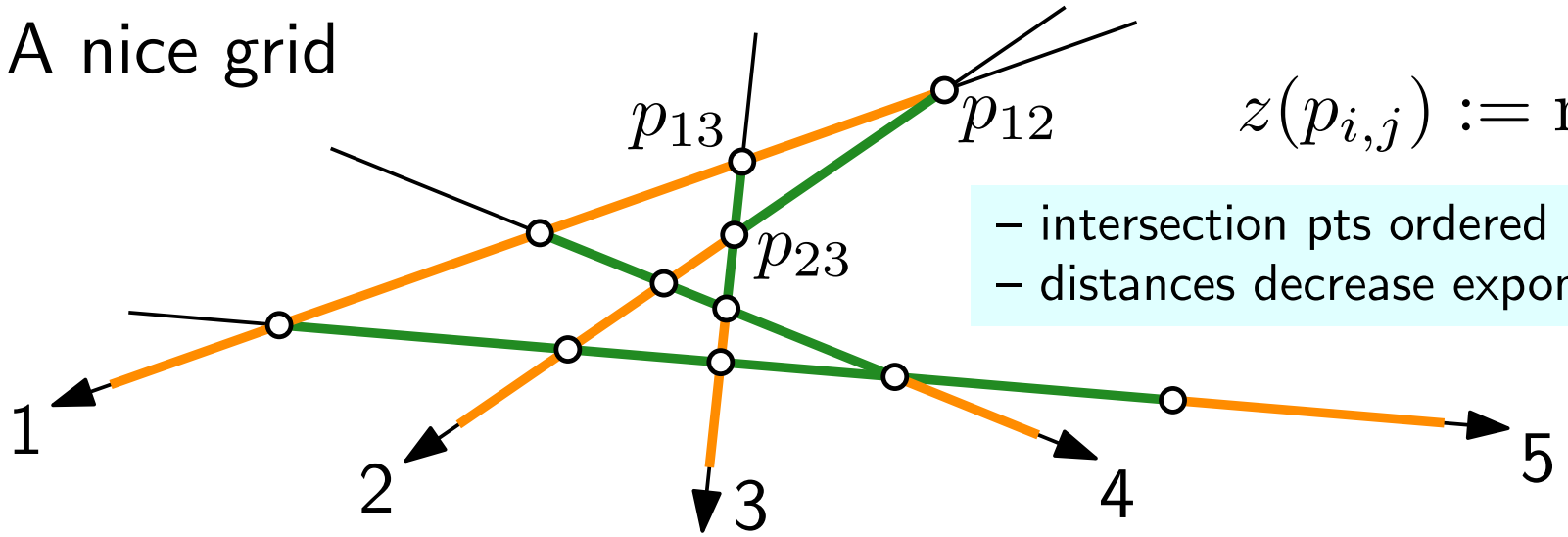


$P_i = \mathrm{CH}(\{p_{ij} : i \neq j\})$

Any Graph Admits such a Contact Representation

We just need two ingredients:

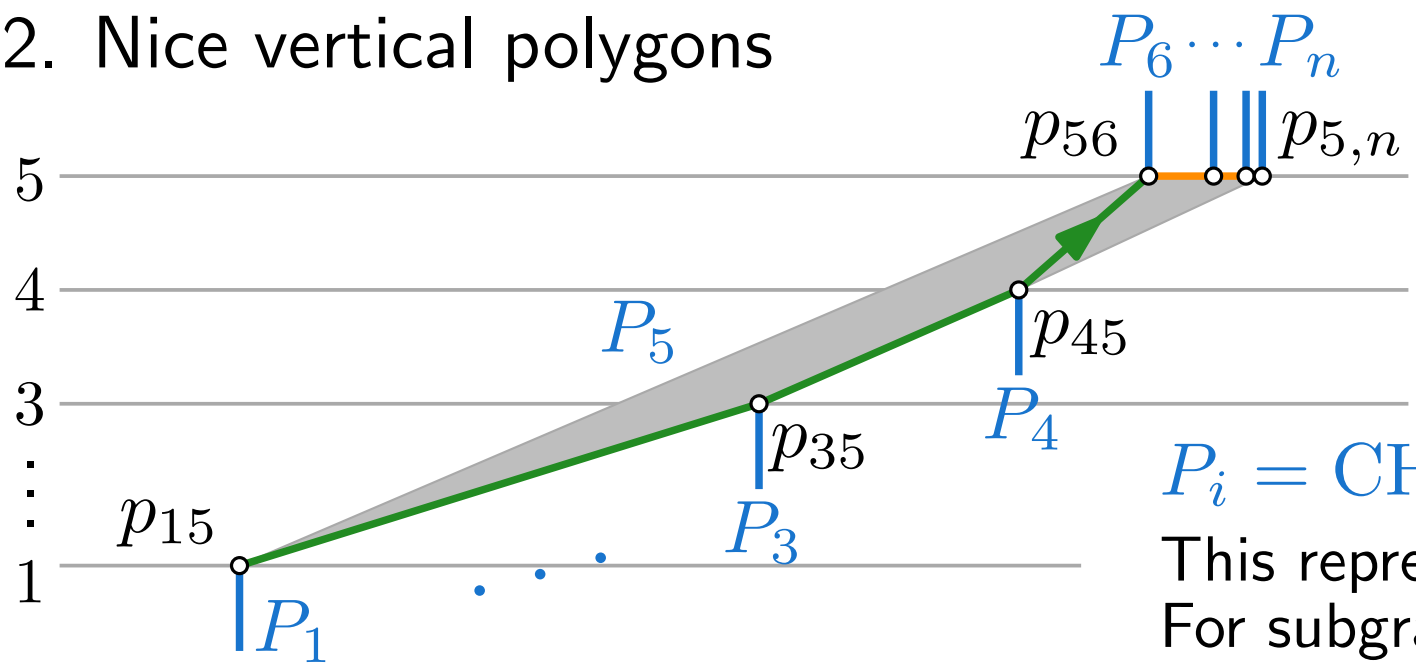
1. A nice grid



$z(p_{i,j}) := \min\{i, j\}$

– intersection pts ordered along each line
– distances decrease exponentially

2. Nice vertical polygons



$P_i = \mathrm{CH}(\{p_{ij} : i \neq j\})$

This represents K_n.
For subgraphs, remove contacts.

Graphs vs. Sets

Graphs vs. Sets

- Graphs are defined by a set of edges, which are sets of two elements.

Graphs vs. Sets

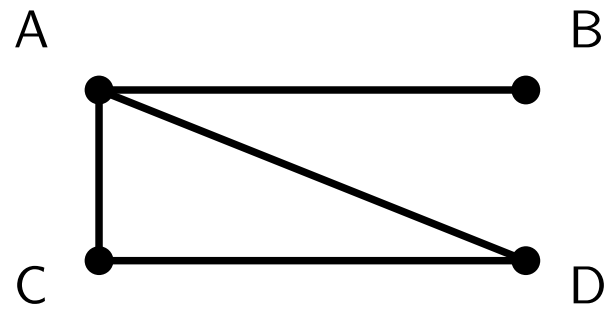
- Graphs are defined by a set of edges, which are sets of two elements.

$$\{A, B\}, \{C, D\},$$
$$\{D, A\}, \{A, C\}$$

Graphs vs. Sets

- Graphs are defined by a set of edges, which are sets of two elements.

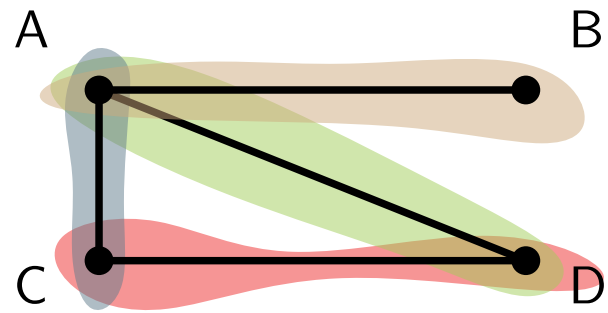
$$\{A, B\}, \{C, D\},$$
$$\{D, A\}, \{A, C\}$$



Graphs vs. Sets

- Graphs are defined by a set of edges, which are sets of two elements.

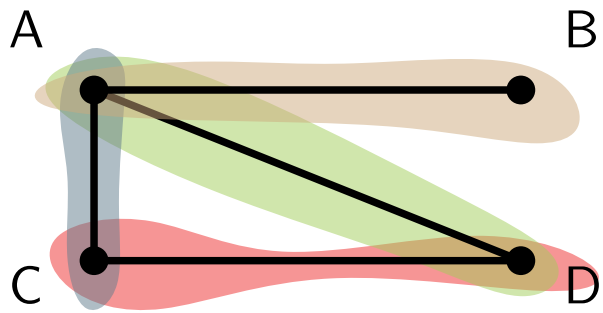
$$\{A, B\}, \{C, D\},$$
$$\{D, A\}, \{A, C\}$$



Graphs vs. Sets

- Graphs are defined by a set of edges, which are sets of two elements.

$$\{A, B\}, \{C, D\},$$
$$\{D, A\}, \{A, C\}$$

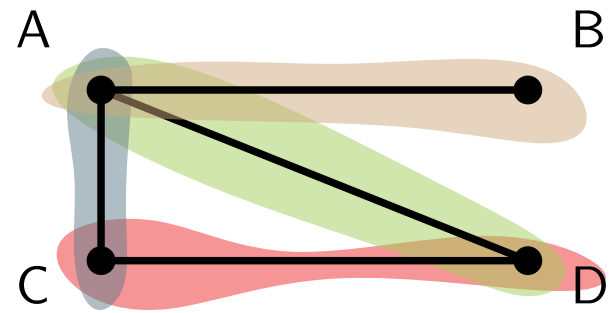


- Hierarchical data can be describes by a tree

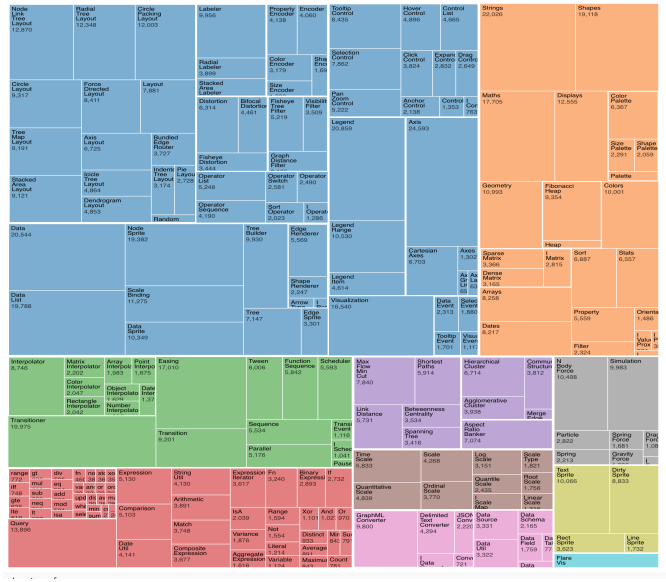
Graphs vs. Sets

- Graphs are defined by a set of edges, which are sets of two elements.

$$\{A, B\}, \{C, D\},$$
$$\{D, A\}, \{A, C\}$$



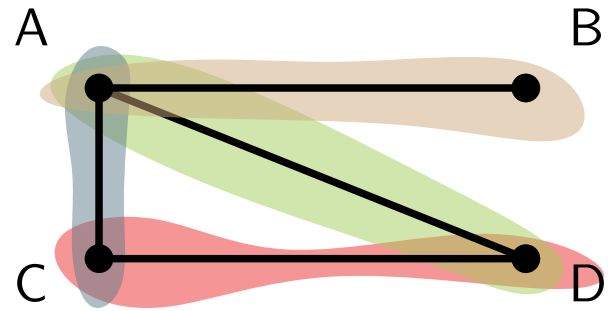
- Hierarchical data can be describes by a tree



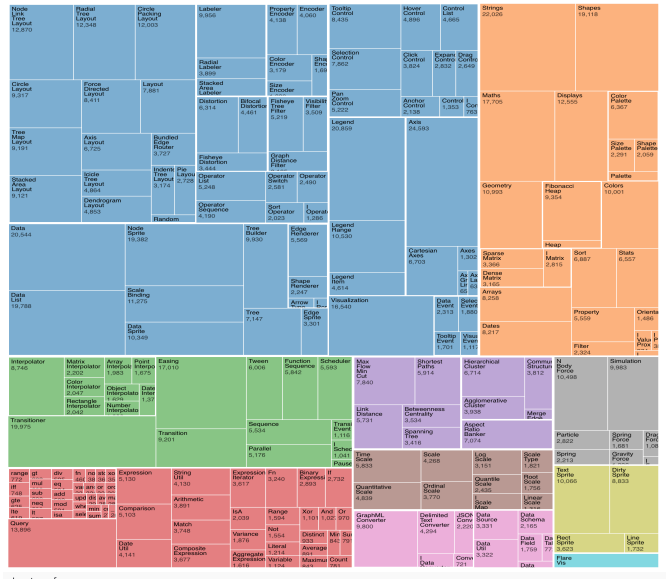
Graphs vs. Sets

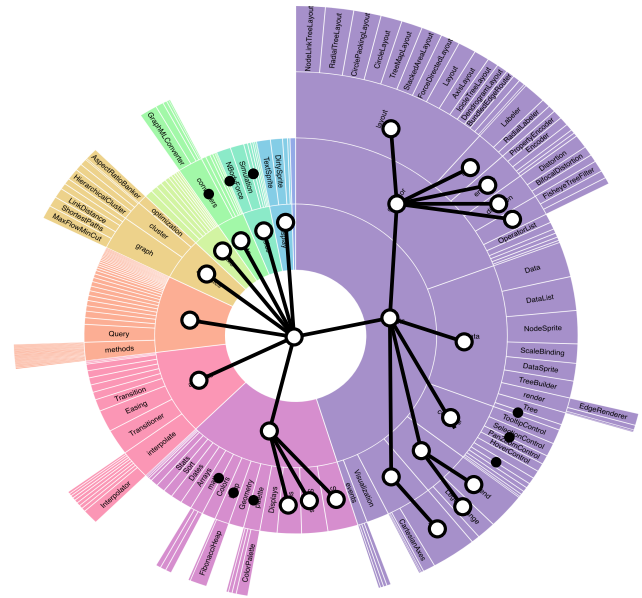
- Graphs are defined by a set of edges, which are sets of two elements.

$$\{A, B\}, \{C, D\},$$
$$\{D, A\}, \{A, C\}$$



- Hierarchical data can be describes by a tree





Hypergraphs

Hypergraphs

- Hypergraphs can model any collection of sets.

Hypergraphs

- Hypergraphs can model any collection of sets.

  Graph $G = (V, E)$
  $E \subseteq \{\{a, b\} \mid a, b \in V\}$

Hypergraphs

- Hypergraphs can model any collection of sets.

Graph $G = (V, E)$
$E \subseteq \{\{a, b\} \mid a, b \in V\}$

Hypergraph $H = (V, E)$
$E \subseteq \{X \mid X \subseteq V\}$

Hypergraphs

- Hypergraphs can model any collection of sets.

Hyperedges

Graph $G = (V, E)$

$E \subseteq \{\{a, b\} \mid a, b \in V\}$

Hypergraph $H = (V, E)$

$E \subseteq \{X \mid X \subseteq V\}$

Hypergraphs

- Hypergraphs can model any collection of sets.

Hyperedges

Graph $G = (V, E)$	Hypergraph $H = (V, E)$
$E \subseteq \{\{a, b\} \mid a, b \in V\}$	$E \subseteq \{X \mid X \subseteq V\}$

Example:

$V = \{\text{black, red, green, yellow, blue, white, orange}\}$

Hypergraphs

- Hypergraphs can model any collection of sets.

Hyperedges

Graph $G = (V, E)$
$E \subseteq \{\{a, b\} \mid a, b \in V\}$

Hypergraph $H = (V, E)$
$E \subseteq \{X \mid X \subseteq V\}$

Example:

$V = \{\text{black, red, green, yellow, blue, white, orange}\}$

$E = \big\{\{\text{red, white}\}, \{\text{black, red, yellow}\}, \{\text{blue, white, red}\},$
$\{\text{blue, yellow}\}, \{\text{green, white, red}\}, \{\text{green, white, orange}\},$
$\{\text{blue, black, white}\}, \{\text{blue, yellow, red}\}, \{\text{blue, white}\},$
$\{\text{green, red}\}, \{\text{red, yellow}\}\big\}$

Hypergraph drawing

Hypergraph drawing

- Many ideas have been proposed to generalize graph drawing methodology for hypergraphs.

Hypergraph drawing

- Many ideas have been proposed to generalize graph drawing methodology for hypergraphs.

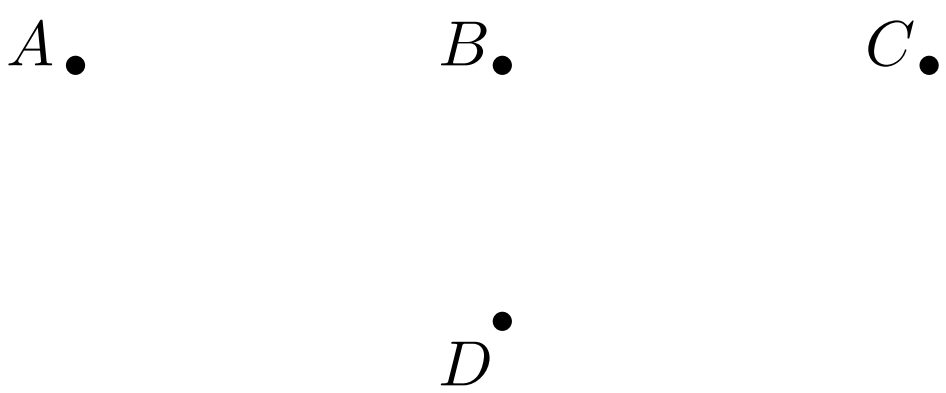
- **Subset-based method:**
  draw for every hyperedge a curve enclosing its vertices

Hypergraph drawing

- Many ideas have been proposed to generalize graph drawing methodology for hypergraphs.

- **Subset-based method:**
  draw for every hyperedge a curve enclosing its vertices

$$H = \Big(\{A, B, C, D\}, \big\{\{A, B\}, \{B, C, D\}, \{A, D, C\}\big\}\Big)$$

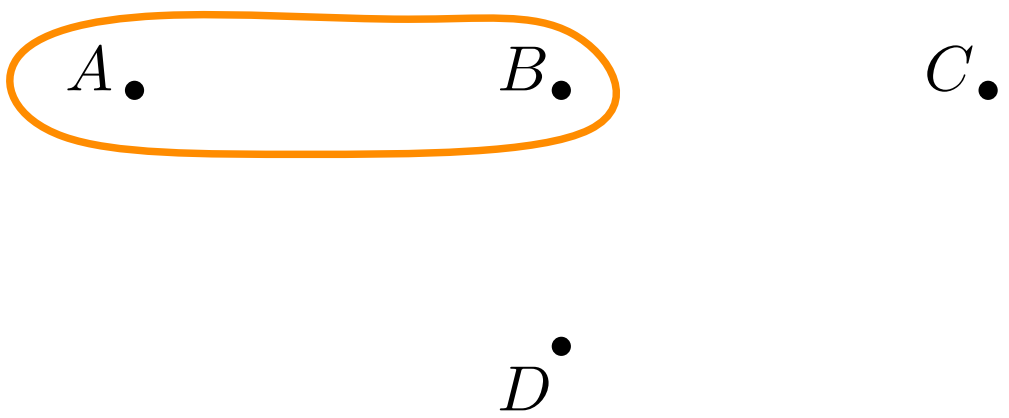


Hypergraph drawing

- Many ideas have been proposed to generalize graph drawing methodology for hypergraphs.

- **Subset-based method:**
  draw for every hyperedge a curve enclosing its vertices

$$H = \Big(\{A, B, C, D\}, \big\{\{A, B\}, \{B, C, D\}, \{A, D, C\}\big\}\Big)$$

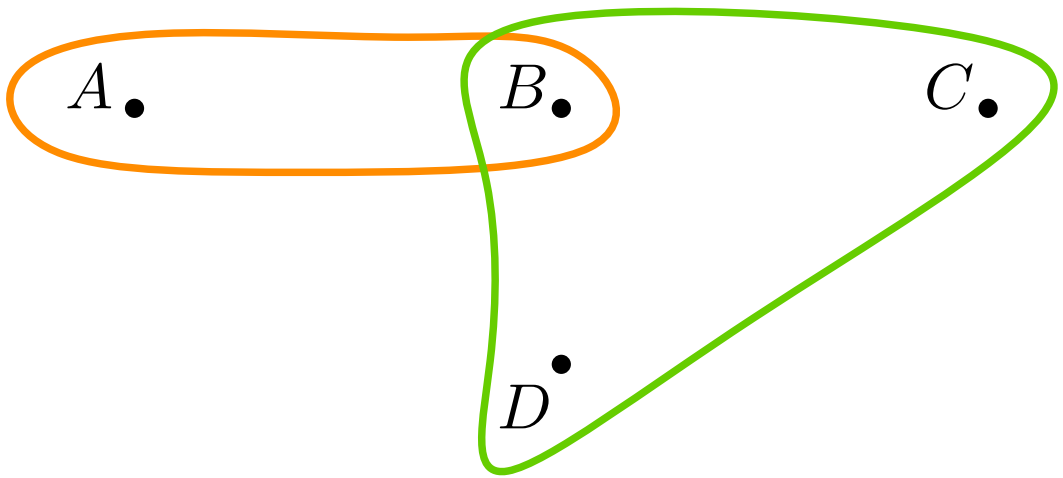


Hypergraph drawing

- Many ideas have been proposed to generalize graph drawing methodology for hypergraphs.

- **Subset-based method:**
  draw for every hyperedge a curve enclosing its vertices

$$H = \Big(\{A, B, C, D\}, \{\{A, B\}, \{B, C, D\}, \{A, D, C\}\}\Big)$$

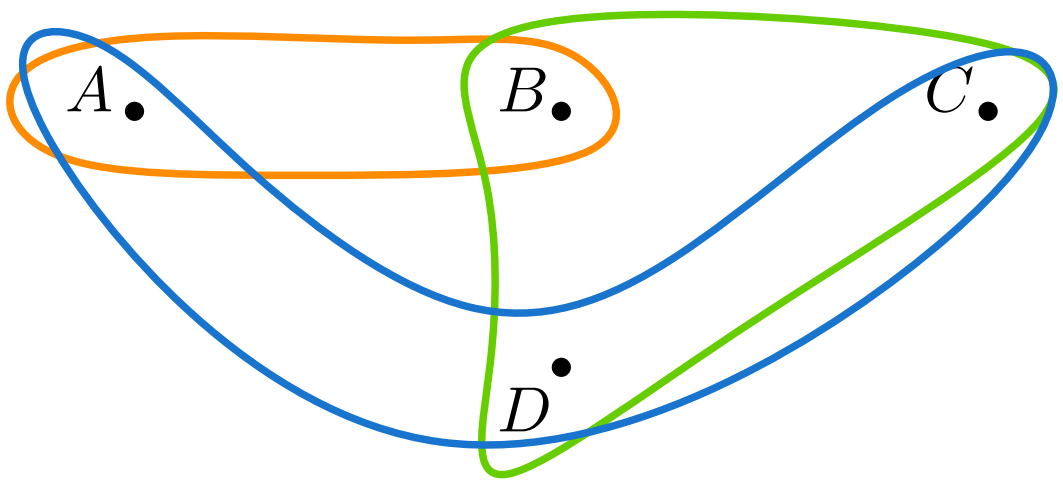


Hypergraph drawing

- Many ideas have been proposed to generalize graph drawing methodology for hypergraphs.

- **Subset-based method:**
  draw for every hyperedge a curve enclosing its vertices

$$H = \Big(\{A, B, C, D\}, \big\{\{A, B\}, \{B, C, D\}, \{A, D, C\}\big\}\Big)$$

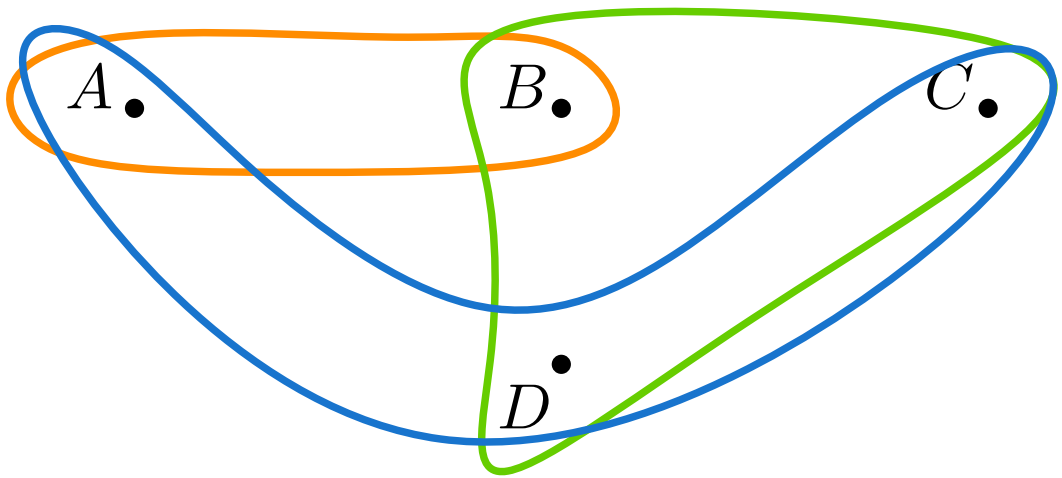


Hypergraph drawing

- Many ideas have been proposed to generalize graph drawing methodology for hypergraphs.

- **Subset-based method:**
  draw for every hyperedge a curve enclosing its vertices

$$H = \Big(\{A, B, C, D\}, \big\{\{A, B\}, \{B, C, D\}, \{A, D, C\}\big\}\Big)$$



- spring embedder algorithm by Bertault and Eades 2000

Hypergraph drawing cont.

Hypergraph drawing cont.

- Subset-based method gets easily confusing.

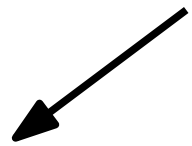
Hypergraph drawing cont.

- Subset-based method gets easily confusing.
- **Alternatives:** subdivision-based & edge-based

Hypergraph drawing cont.

- Subset-based method gets easily confusing.

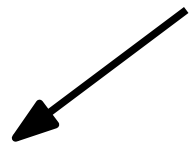
- **Alternatives:** subdivision-based & edge-based



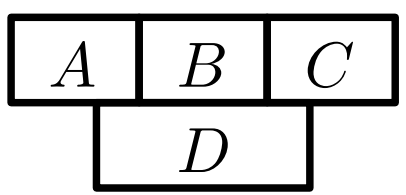
  - vertices are regions
  - hyperedges are connected unions
  - ... and more criteria

Hypergraph drawing cont.

- Subset-based method gets easily confusing.
- **Alternatives:** subdivision-based & edge-based



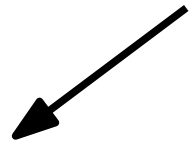
  - vertices are regions
  - hyperedges are connected unions
  - ... and more criteria



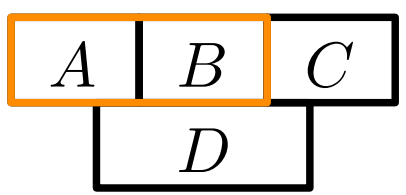
$$H = \Big(\{A, B, C, D\}, \big\{\{A, B\}, \{B, C, D\}, \{A, D, C\}\big\}\Big)$$

Hypergraph drawing cont.

- Subset-based method gets easily confusing.
- **Alternatives:** subdivision-based & edge-based



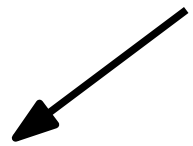
  - vertices are regions
  - hyperedges are connected unions
  - ... and more criteria



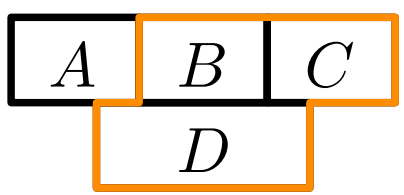
$$H = \Big(\{A, B, C, D\}, \big\{\{A, B\}, \{B, C, D\}, \{A, D, C\}\big\}\Big)$$

Hypergraph drawing cont.

- Subset-based method gets easily confusing.
- **Alternatives:** subdivision-based & edge-based



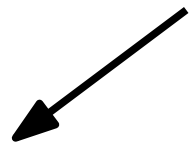
  - vertices are regions
  - hyperedges are connected unions
  - ... and more criteria



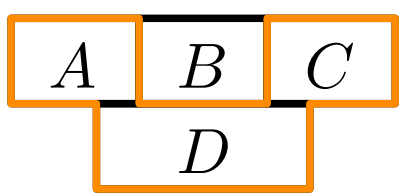
$$H = \Big(\{A, B, C, D\}, \big\{\{A, B\}, \{B, C, D\}, \{A, D, C\}\big\}\Big)$$

Hypergraph drawing cont.

- Subset-based method gets easily confusing.
- **Alternatives:** subdivision-based & edge-based



  - vertices are regions
  - hyperedges are connected unions
  - ... and more criteria



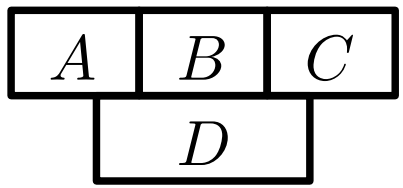
$$H = \Big(\{A, B, C, D\}, \big\{\{A, B\}, \{B, C, D\}, \{A, D, C\}\big\}\Big)$$

Hypergraph drawing cont.

- Subset-based method gets easily confusing.
- **Alternatives:** subdivision-based & edge-based

  – vertices are regions
  – hyperedges are connected unions
  – ... and more criteria

  – drawn as node-link diagram, with vertices as some nodes
  – hyperedges yield connected subgraphs
  – ... and more criteria



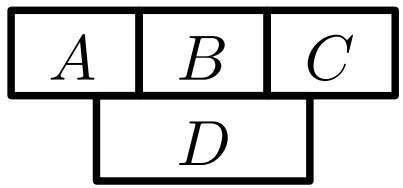
$$H = \Big(\{A, B, C, D\}, \big\{\{A, B\}, \{B, C, D\}, \{A, D, C\}\big\}\Big)$$

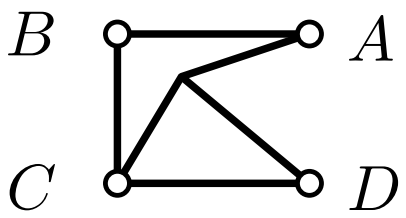
Hypergraph drawing cont.

- Subset-based method gets easily confusing.
- **Alternatives:** subdivision-based & edge-based

  – vertices are regions
  – hyperedges are connected unions
  – ... and more criteria

  – drawn as node-link diagram, with vertices as some nodes
  – hyperedges yield connected subgraphs
  – ... and more criteria





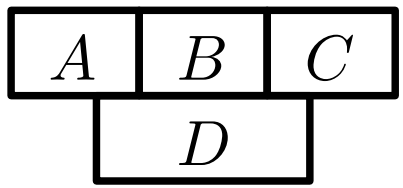
$$H = \Big(\{A, B, C, D\}, \big\{\{A, B\}, \{B, C, D\}, \{A, D, C\}\big\}\Big)$$

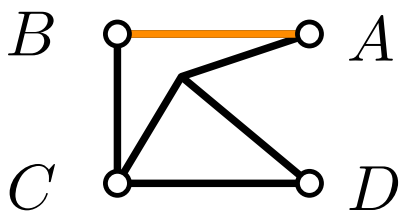
Hypergraph drawing cont.

- Subset-based method gets easily confusing.
- **Alternatives:** subdivision-based & edge-based

  – vertices are regions
  – hyperedges are connected unions
  – ... and more criteria

  – drawn as node-link diagram, with vertices as some nodes
  – hyperedges yield connected subgraphs
  – ... and more criteria





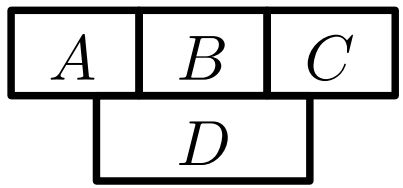
$$H = \Big(\{A, B, C, D\}, \big\{\underline{\{A, B\}}, \{B, C, D\}, \{A, D, C\}\big\}\Big)$$

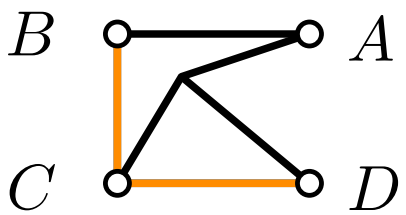
Hypergraph drawing cont.

- Subset-based method gets easily confusing.
- **Alternatives:** subdivision-based & edge-based

  – vertices are regions
  – hyperedges are connected unions
  – ... and more criteria

  – drawn as node-link diagram, with vertices as some nodes
  – hyperedges yield connected subgraphs
  – ... and more criteria





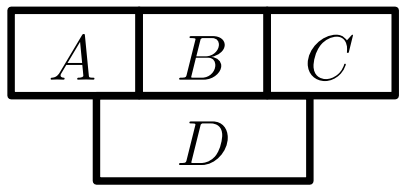
$$H = \Big(\{A, B, C, D\}, \big\{\{A, B\}, \{B, C, D\}, \{A, D, C\}\big\}\Big)$$

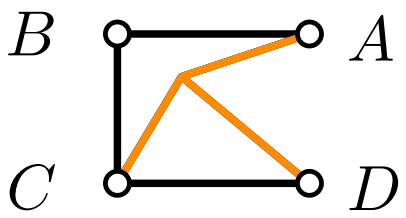
Hypergraph drawing cont.

- Subset-based method gets easily confusing.
- **Alternatives:** subdivision-based & edge-based

  – vertices are regions
  – hyperedges are connected unions
  – ... and more criteria

  – drawn as node-link diagram, with vertices as some nodes
  – hyperedges yield connected subgraphs
  – ... and more criteria





$$H = \Big(\{A, B, C, D\}, \big\{\{A, B\}, \{B, C, D\}, \underline{\{A, D, C\}}\big\}\Big)$$

Subdivision-based methods

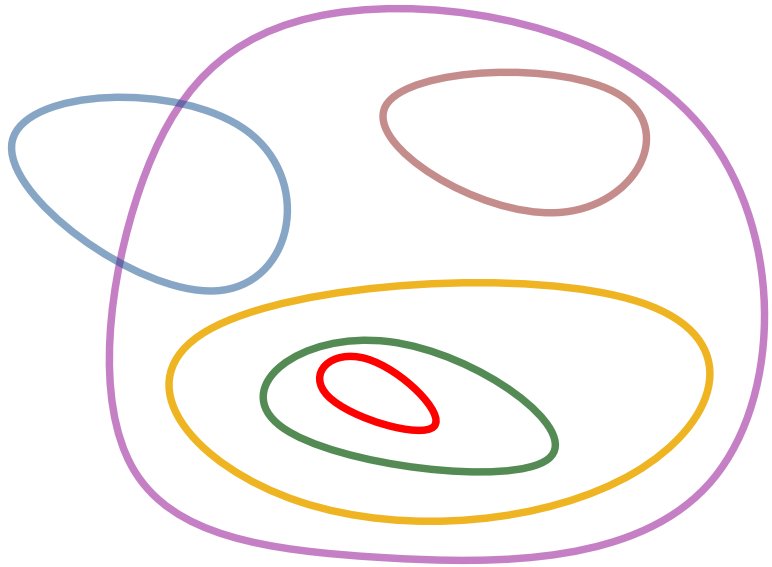
Subdivision-based methods

Concrete Euler Diagrams

Subdivision-based methods

Concrete Euler Diagrams

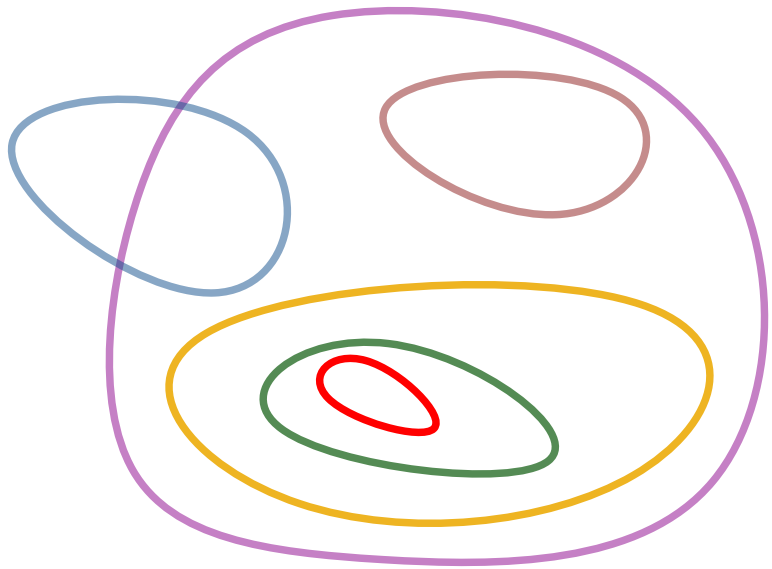
- hyperedges are drawn as simple closed curves (interior/exterior)



Subdivision-based methods

Concrete Euler Diagrams

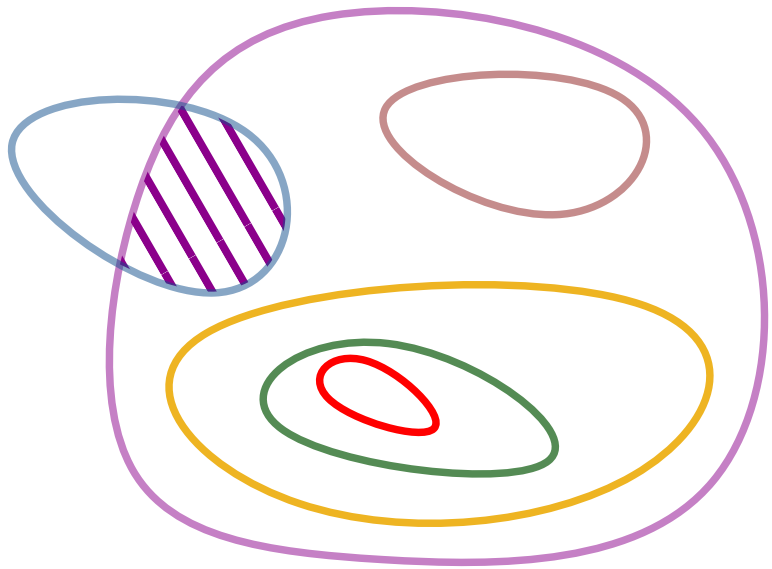
- hyperedges are drawn as simple closed curves (interior/exterior)
- intersections of hyperedges = zone



Subdivision-based methods

Concrete Euler Diagrams

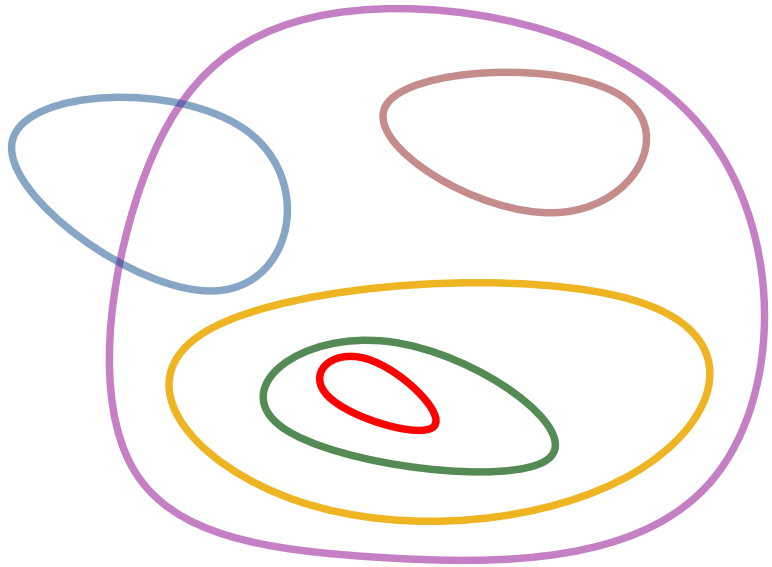
- hyperedges are drawn as simple closed curves (interior/exterior)
- intersections of hyperedges = zone



Subdivision-based methods

Concrete Euler Diagrams

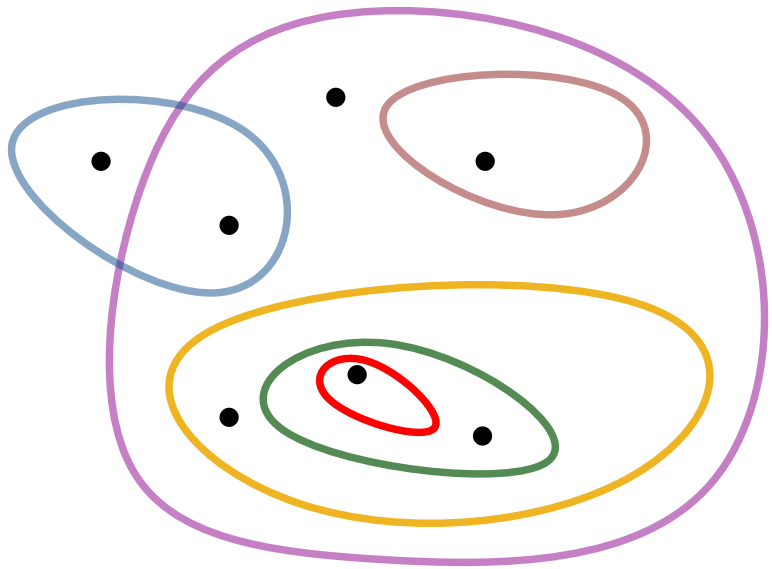
- hyperedges are drawn as simple closed curves (interior/exterior)
- intersections of hyperedges = zone
- for every zone there is a vertex in the corresponding intersection



Subdivision-based methods

Concrete Euler Diagrams

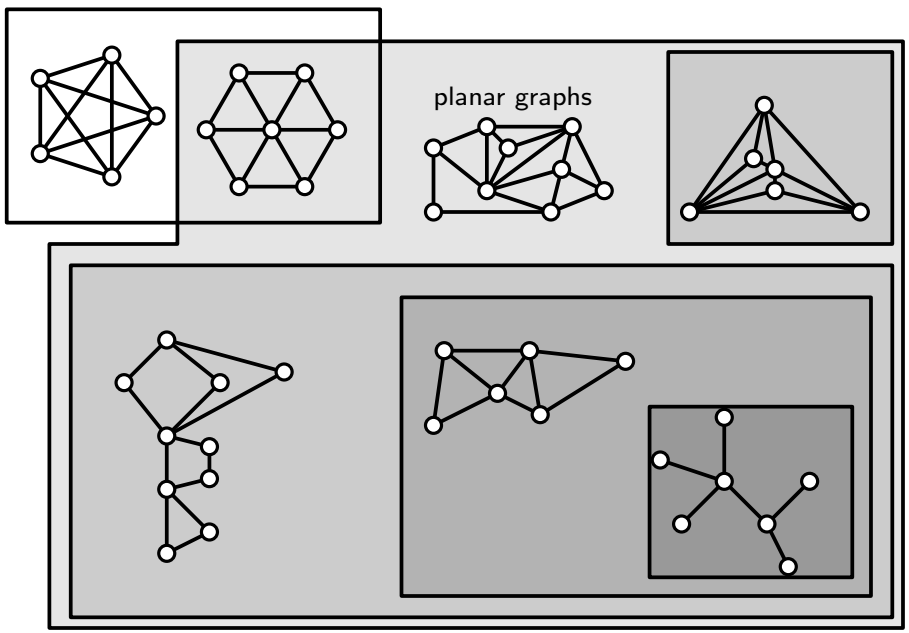
- hyperedges are drawn as simple closed curves (interior/exterior)
- intersections of hyperedges = zone
- for every zone there is a vertex in the corresponding intersection

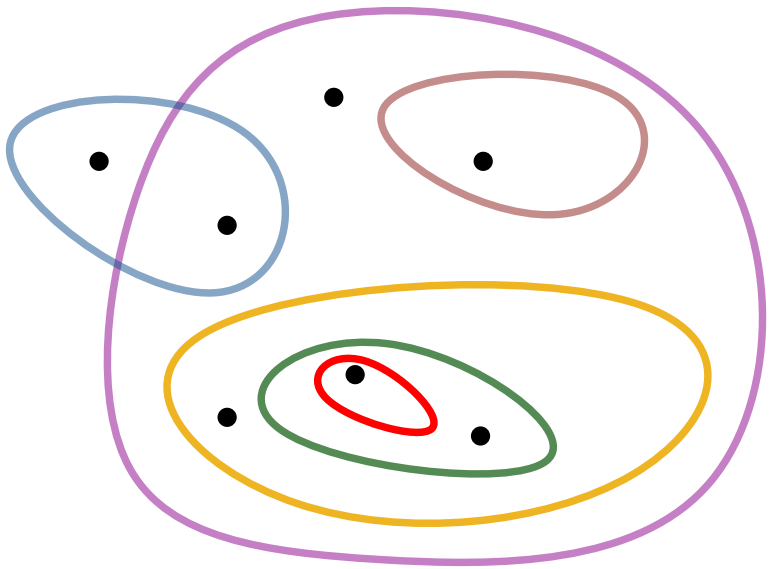


Subdivision-based methods

Concrete Euler Diagrams

- hyperedges are drawn as simple closed curves (interior/exterior)
- intersections of hyperedges = zone
- for every zone there is a vertex in the corresponding intersection



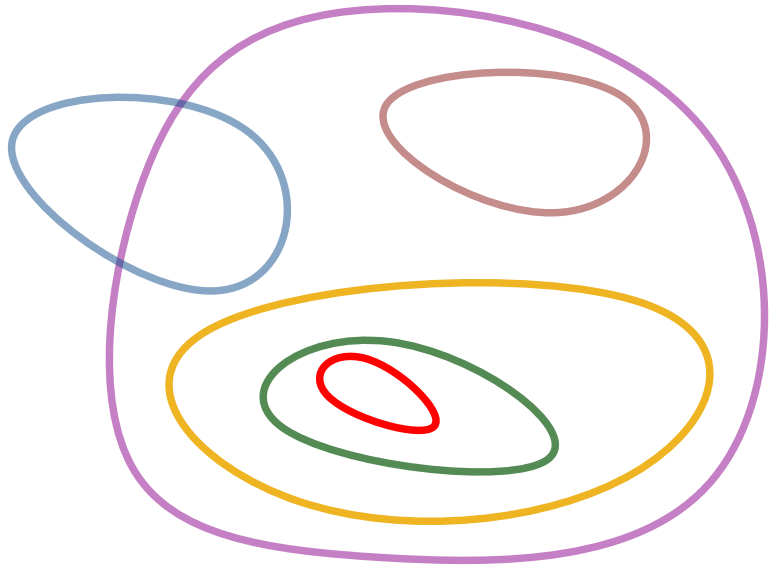


Subdivision-based methods

Concrete Euler Diagrams

- hyperedges are drawn as simple closed curves (interior/exterior)
- intersections of hyperedges = zone
- for every zone there is a vertex in the corresponding intersection
- no two zones for the same intersection

NO

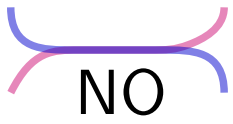


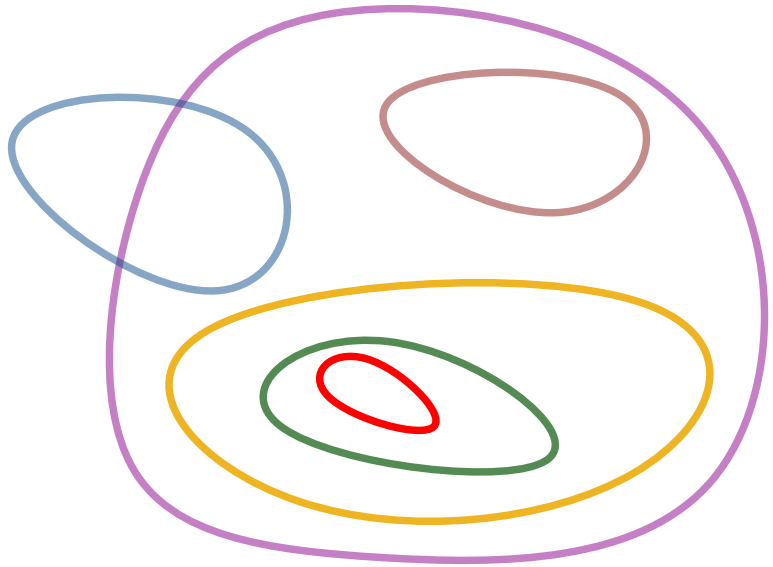
Subdivision-based methods

Concrete Euler Diagrams

- hyperedges are drawn as simple closed curves (interior/exterior)
- intersections of hyperedges = zone
- for every zone there is a vertex in the corresponding intersection
- no two zones for the same intersection

- only proper crossings





Subdivision-based methods

Concrete Euler Diagrams

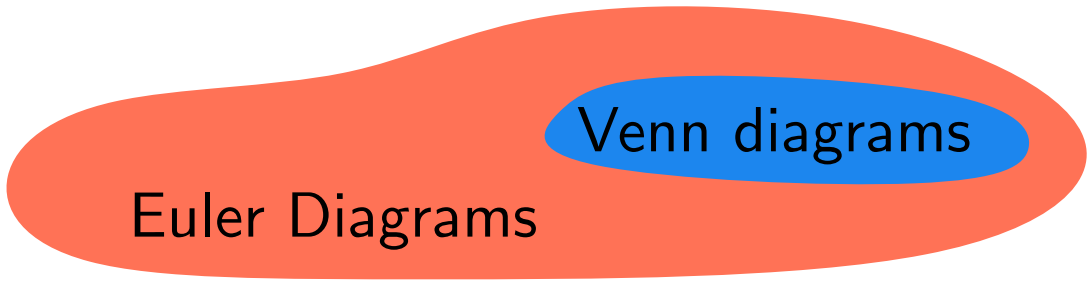
- hyperedges are drawn as simple closed
- intersections of hyperedges = zone
- for every zone there is a vertex in the c
- no two zones for the same intersection

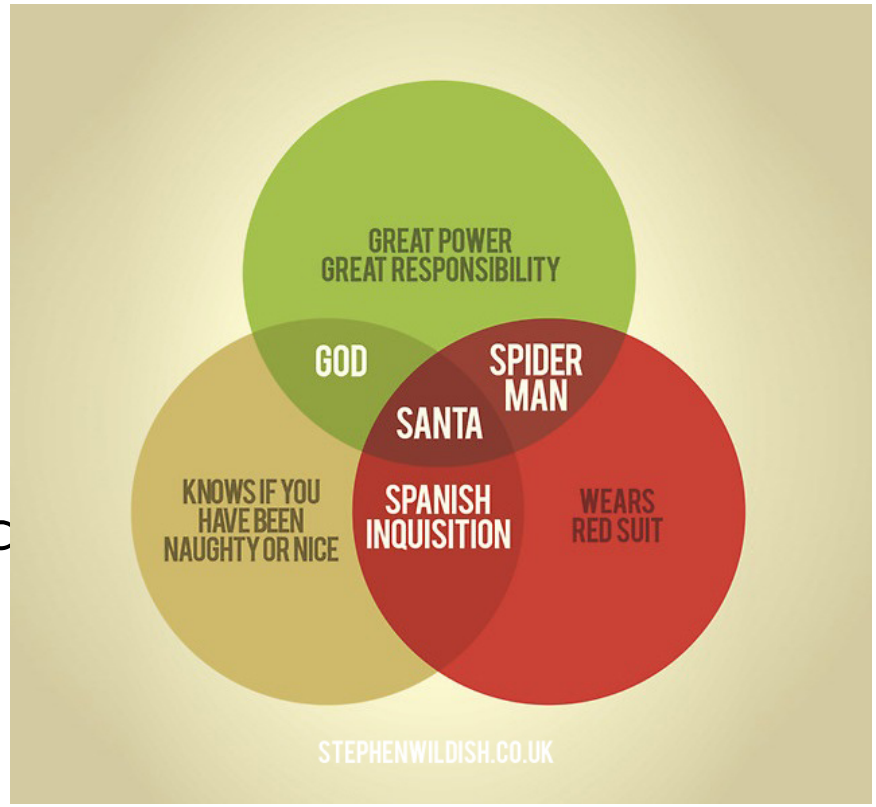
NO

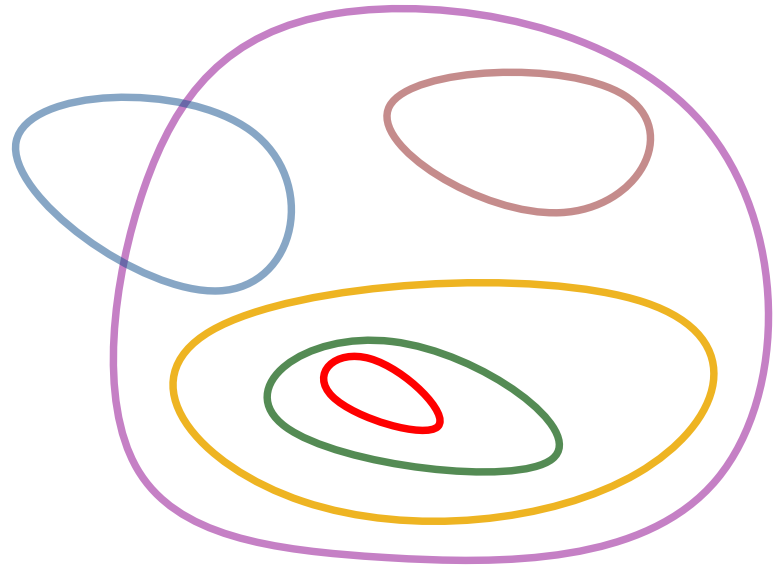
- only proper crossings

NO

NO

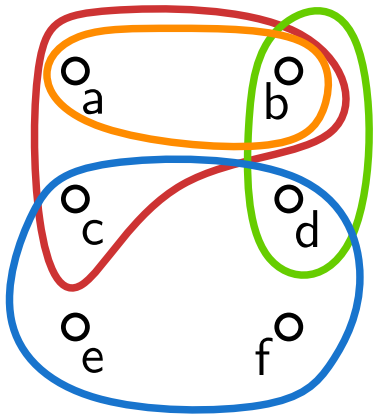






Edge-based methods

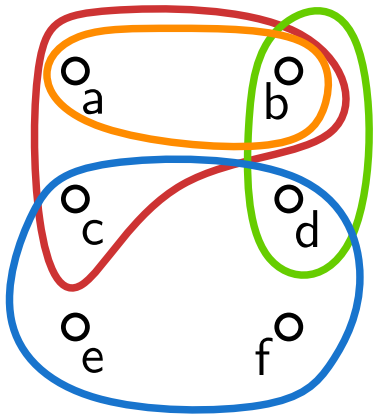
Edge-based methods



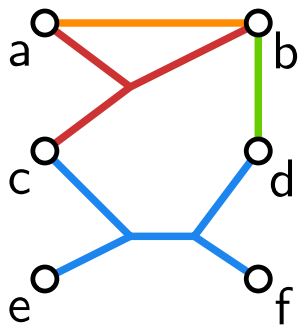
subset-based drawing

Edge-based methods

Examples:



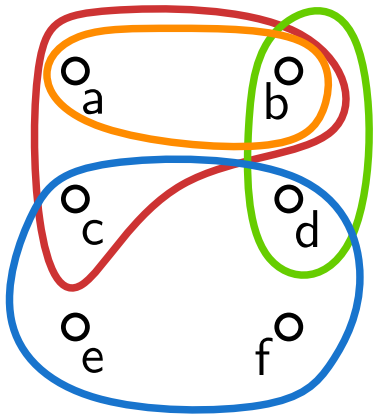
subset-based drawing



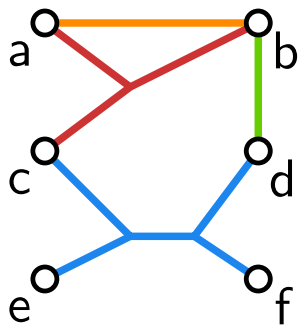
edge-based drawing

Edge-based methods

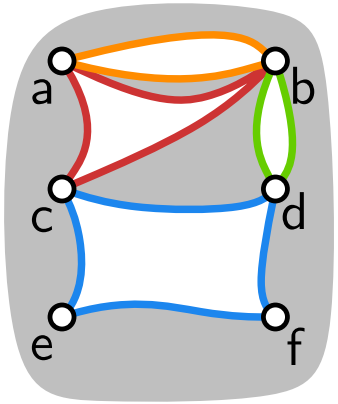
Examples:



subset-based drawing



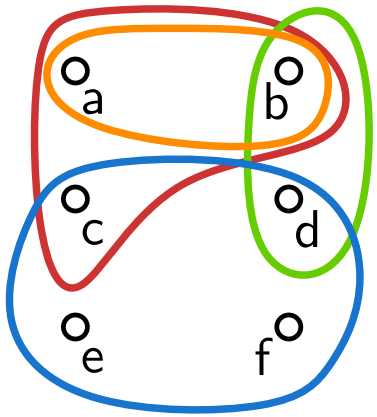
edge-based drawing



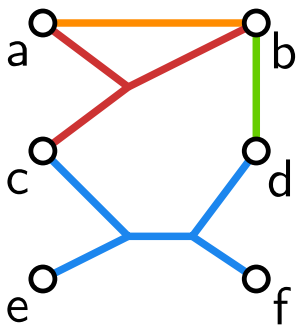
Zykov representation

Edge-based methods

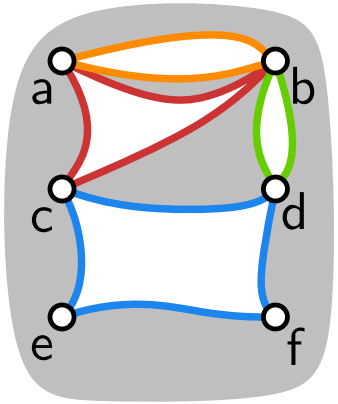
Examples:



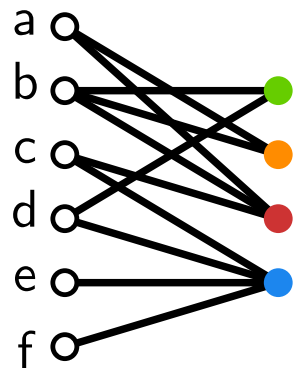
subset-based drawing



edge-based drawing



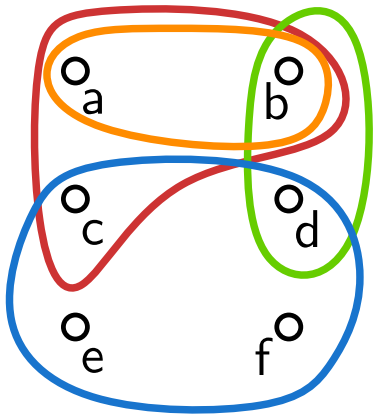
Zykov representation



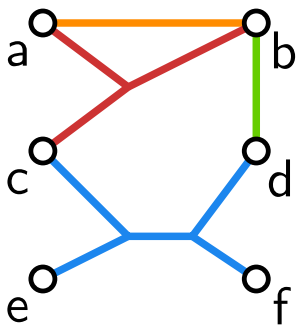
incidence representation

Edge-based methods

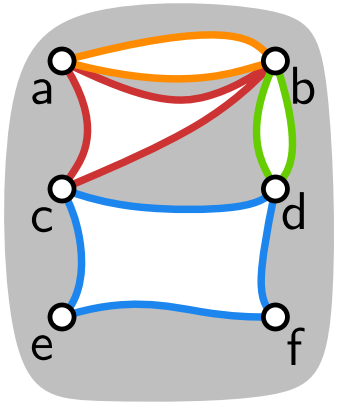
Examples:



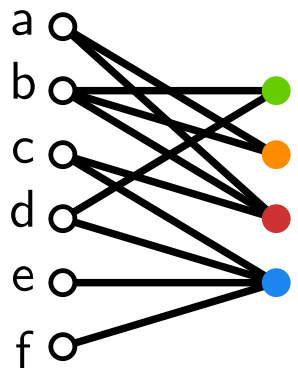
subset-based drawing



edge-based drawing



Zykov representation

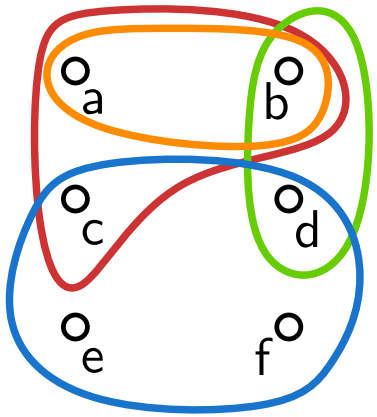


incidence representation

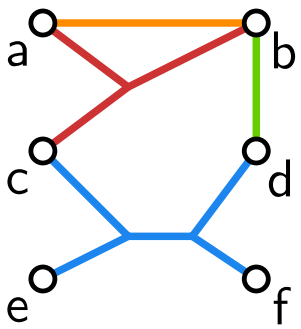
planarity is equivalent in all three models

Edge-based methods

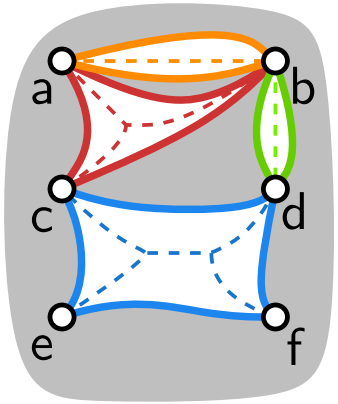
Examples:



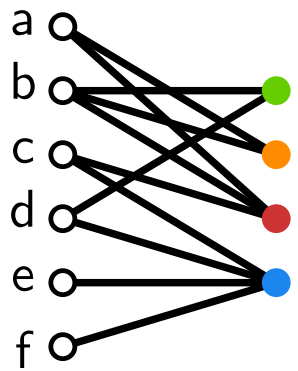
subset-based drawing



edge-based drawing



Zykov representation

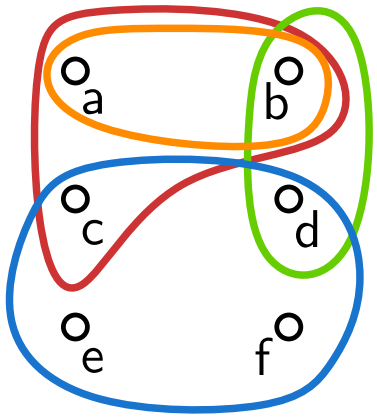


incidence representation

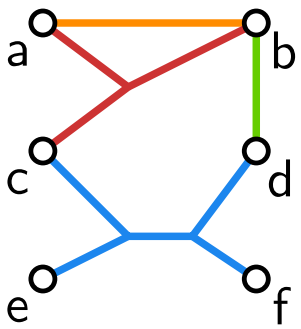
planarity is equivalent in all three models

Edge-based methods

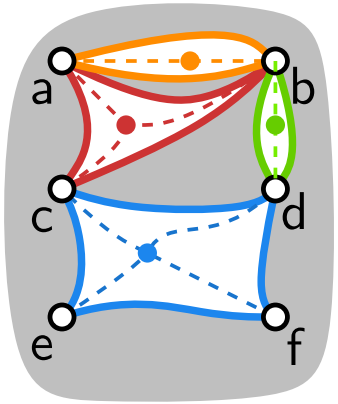
Examples:



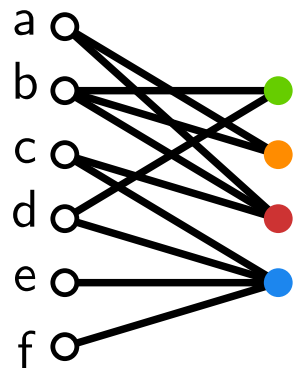
subset-based drawing



edge-based drawing



Zykov representation

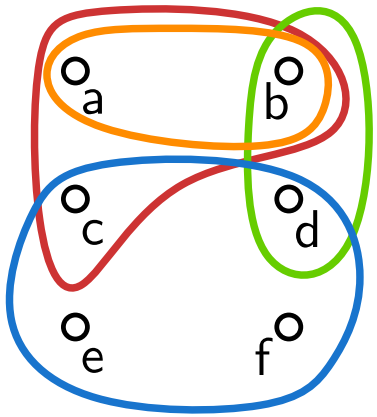


incidence representation

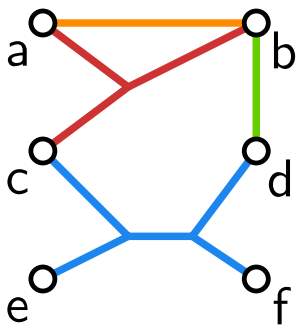
planarity is equivalent in all three models

Edge-based methods

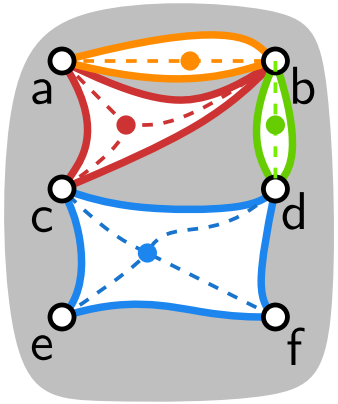
Examples:



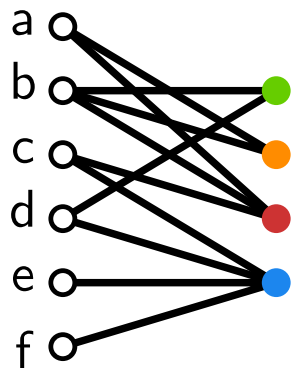
subset-based drawing



edge-based drawing



Zykov representation



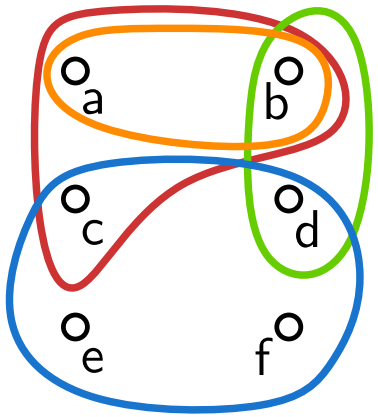
incidence representation

planarity is equivalent in all three models $\in$ P

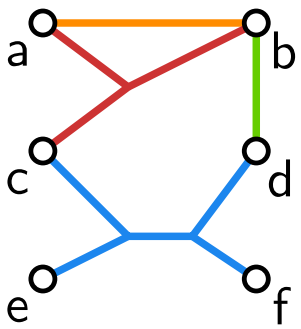
"planarity"
NP-complete

Edge-based methods

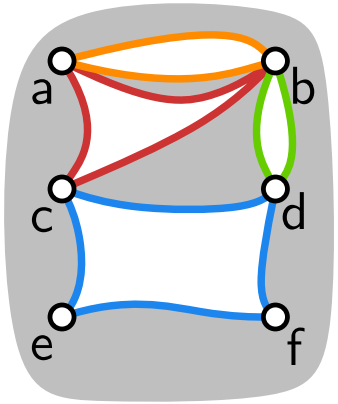
Examples:



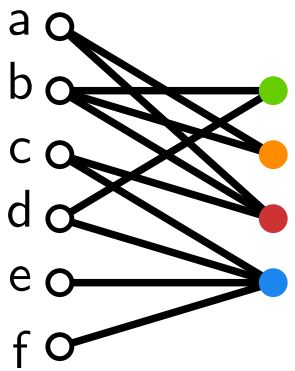
subset-based drawing



edge-based drawing



Zykov representation

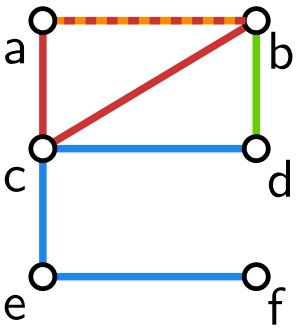


incidence representation

"planarity" NP-complete

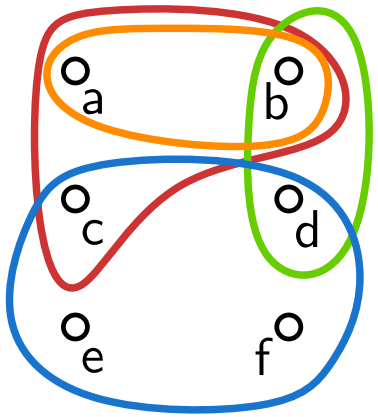
planarity is equivalent in all three models $\in$ P

Def.: Support of a hypergraph is a graph such that every hyperedge induces a connected subgraph.

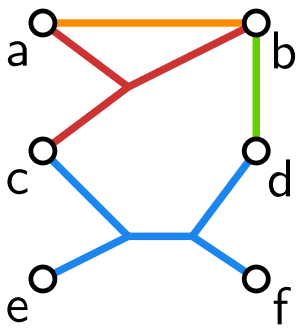


Edge-based methods

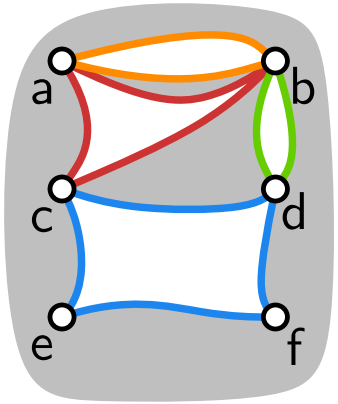
Examples:



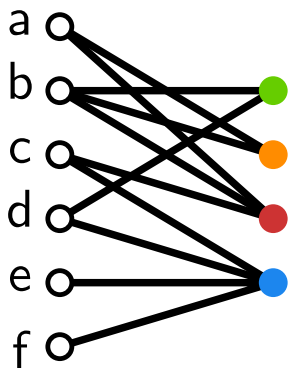
subset-based drawing



edge-based drawing



Zykov representation



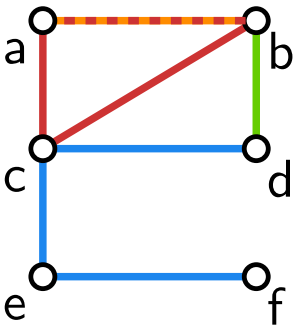
incidence representation

planarity is equivalent in all three models $\in$ P

"planarity" NP-complete

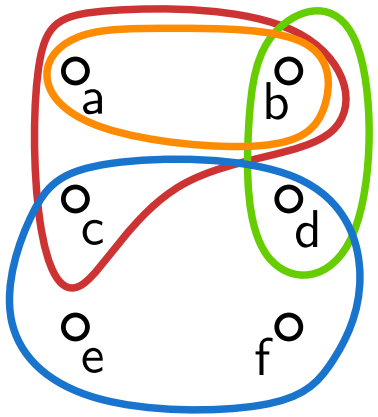
= planar support

Def.: Support of a hypergraph is a graph such that every hyperedge induces a connected subgraph.

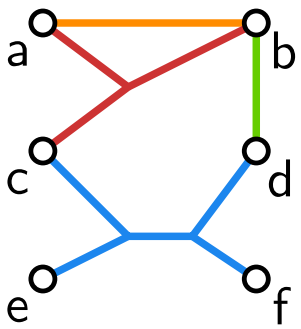


Edge-based methods

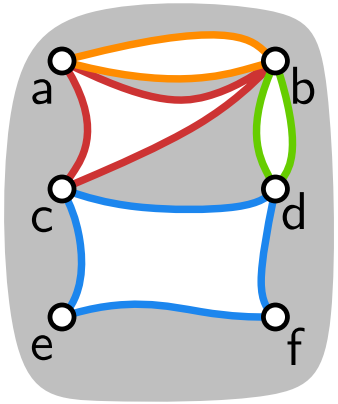
Examples:



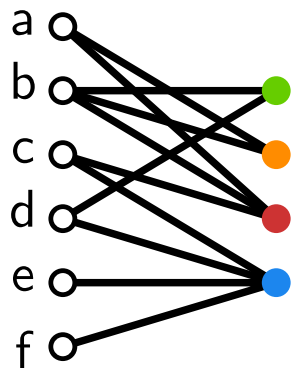
subset-based drawing



edge-based drawing



Zykov representation



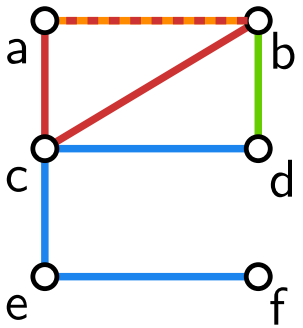
incidence representation

planarity is equivalent in all three models $\in$ P

"planarity" NP-complete

Def.: Support of a hypergraph is a graph such that every hyperedge induces a connected subgraph.

= planar support



Test for cycle-, tree-, or cactus-support is feasible.

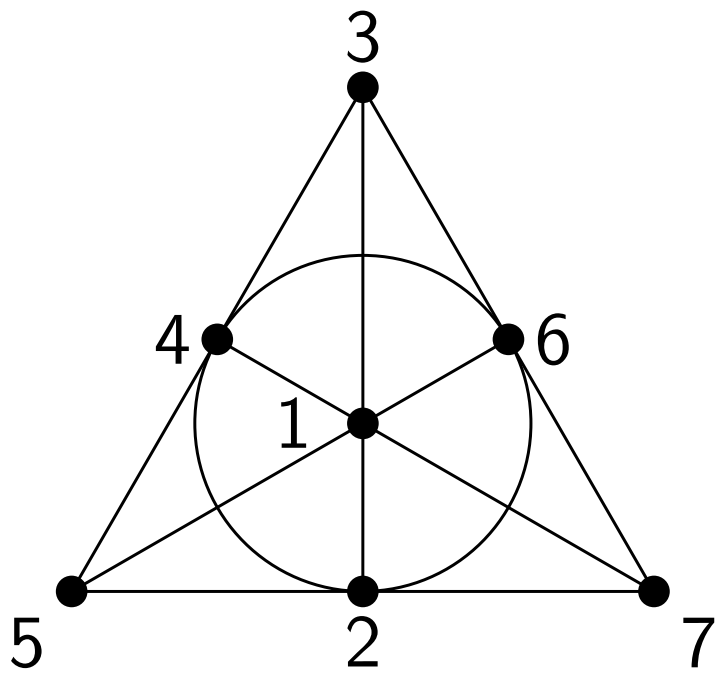
Contact Representations for Hypergraphs in 3D

[Evans, Rzążewski,
Saeedi, Shin, W., GD'19]

Contact Representations for Hypergraphs in 3D

[Evans, Rzążewski, Saeedi, Shin, W., GD'19]

The Fano plane $S(2, 3, 7)$:

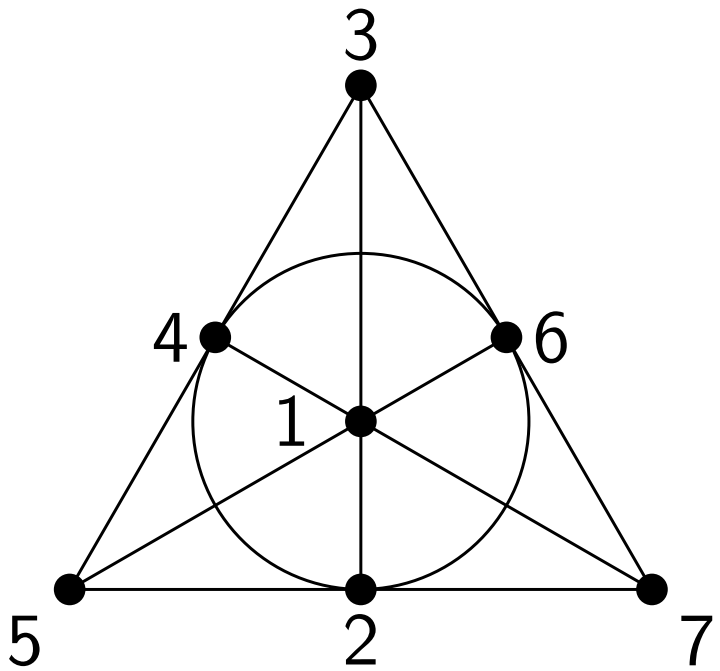


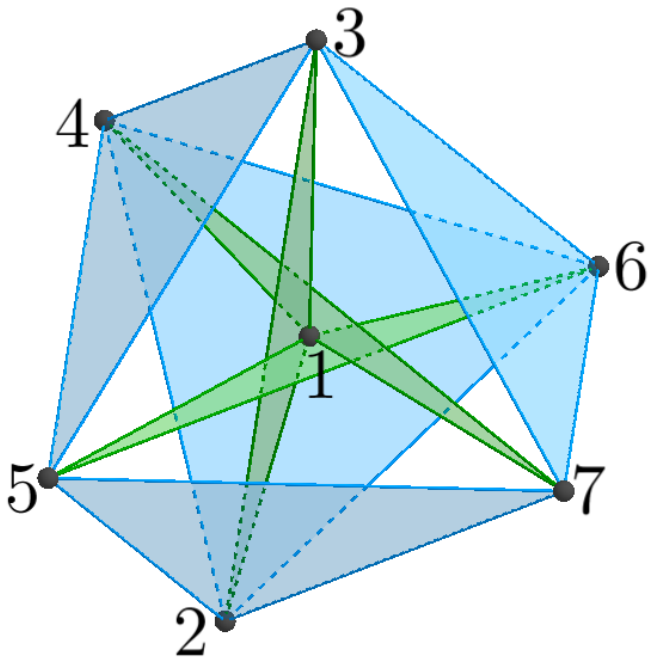
Contact Representations for Hypergraphs in 3D

[Evans, Rzążewski, Saeedi, Shin, W., GD'19]

The Fano plane $S(2,3,7)$:

...can be realized by touching triangles in 3D:



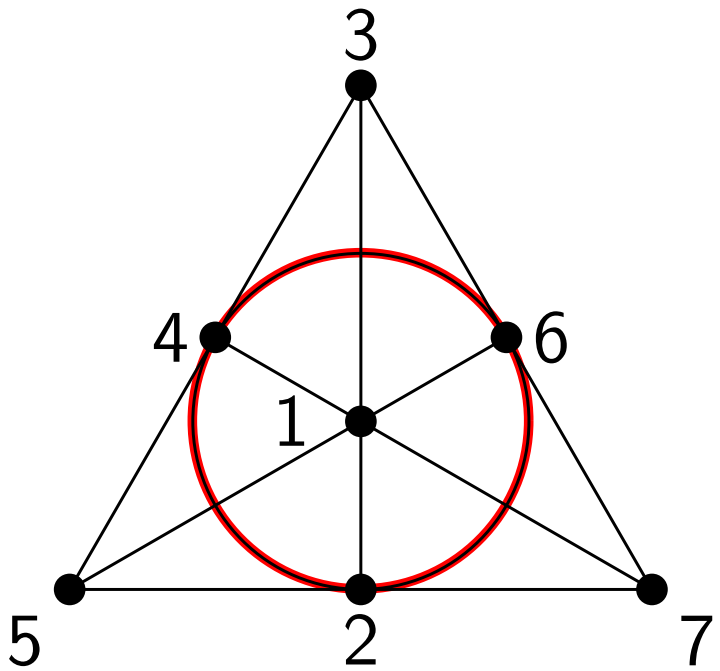


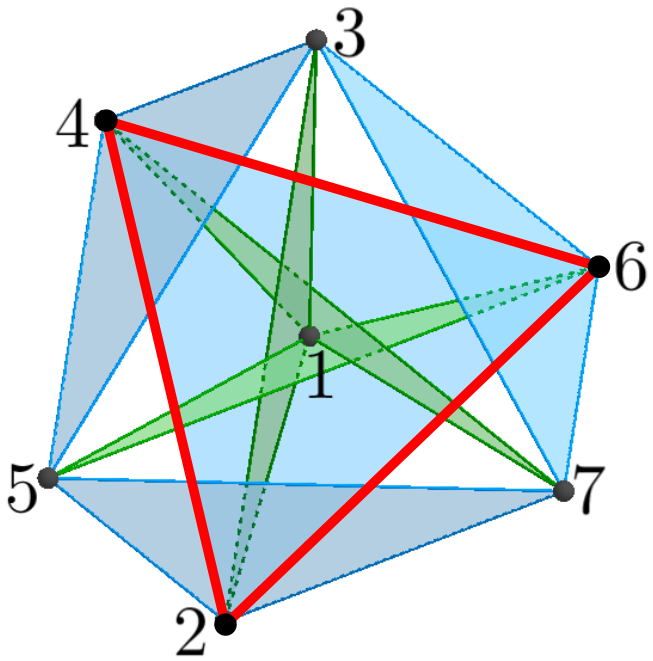
Contact Representations for Hypergraphs in 3D

[Evans, Rzążewski, Saeedi, Shin, W., GD'19]

The Fano plane $S(2,3,7)$:

...can be realized by touching triangles in 3D:



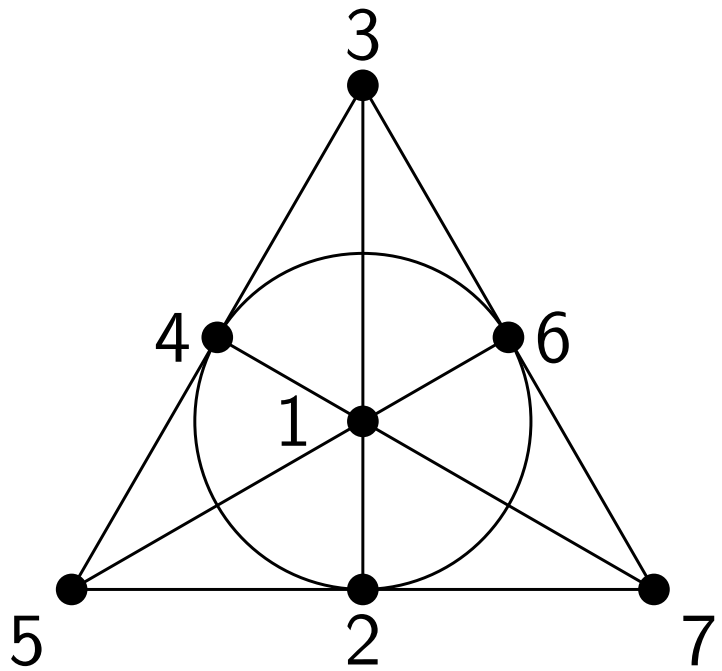


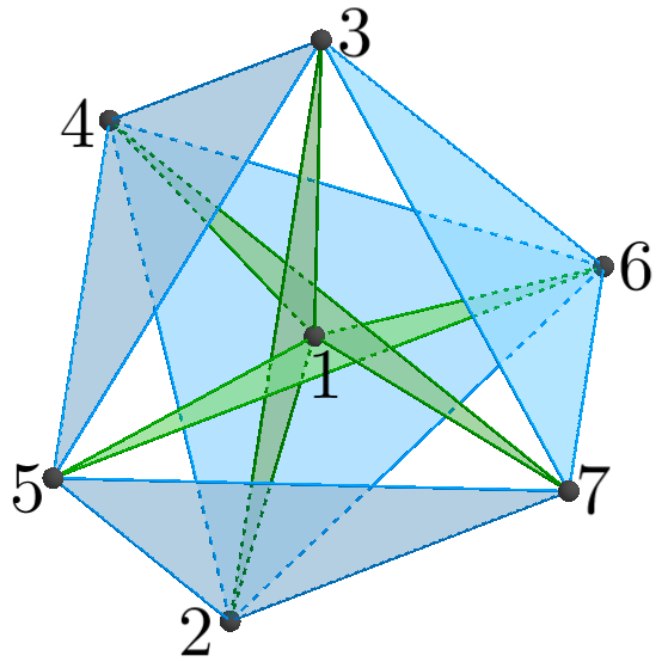
Contact Representations for Hypergraphs in 3D

[Evans, Rzążewski, Saeedi, Shin, W., GD'19]

The Fano plane $S(2,3,7)$:

...can be realized by touching triangles in 3D:





Theorem. [Carmesin/Pardon]

For any $n \geq 6$, the 3-uniform complete hypergraph with n vertices, $\mathcal{K}_n^3$, *cannot* be realized by touching triangles in 3D.

Contact Representations for Hypergraphs in 3D

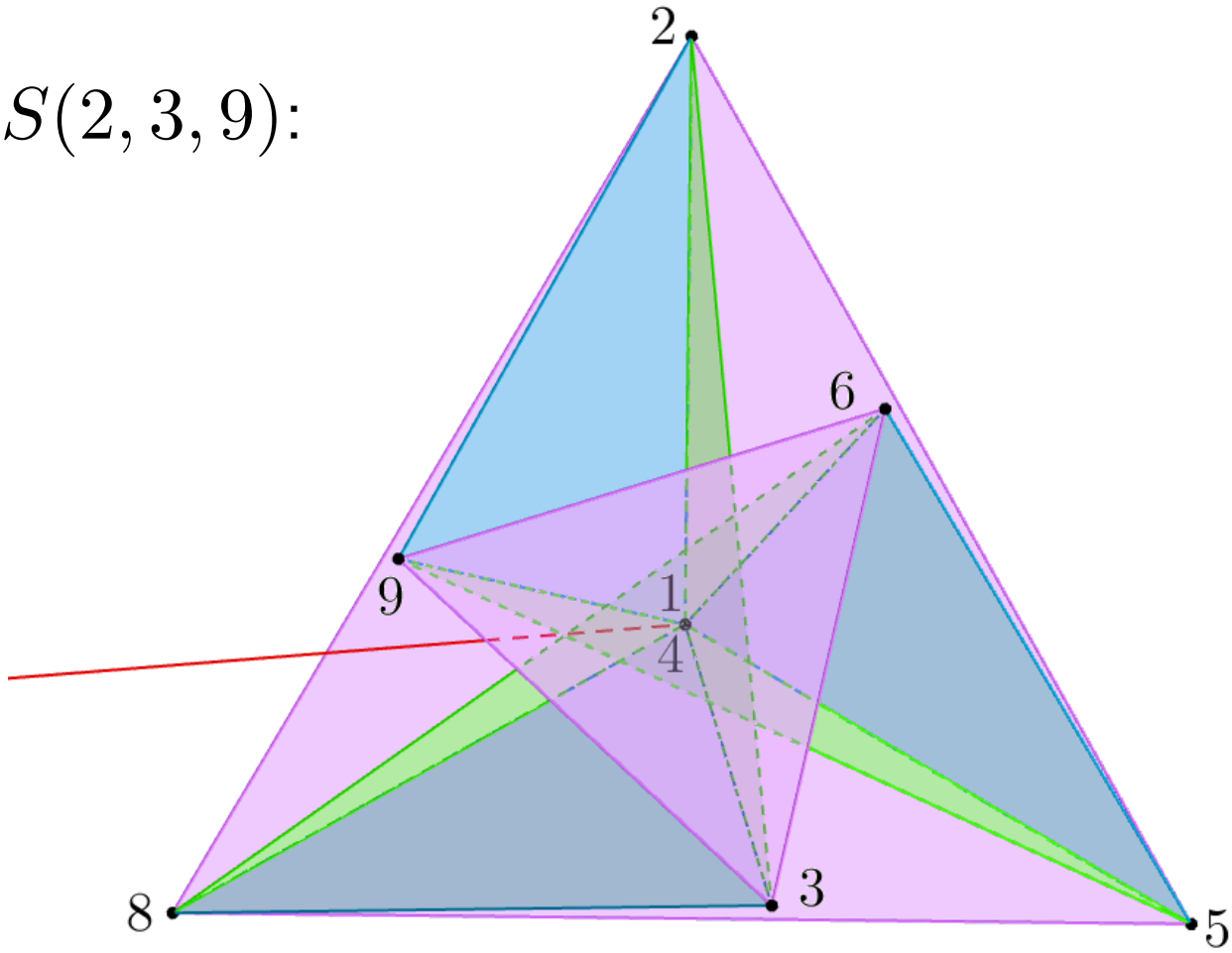
The Steiner triple system $S(2, 3, 9)$:

1 2 3	1 5 9
4 5 6	2 6 7
7 8 9	3 4 8
1 4 7	1 6 8
2 5 8	2 4 9
3 6 9	3 5 7

Contact Representations for Hypergraphs in 3D

The Steiner triple system $S(2, 3, 9)$:

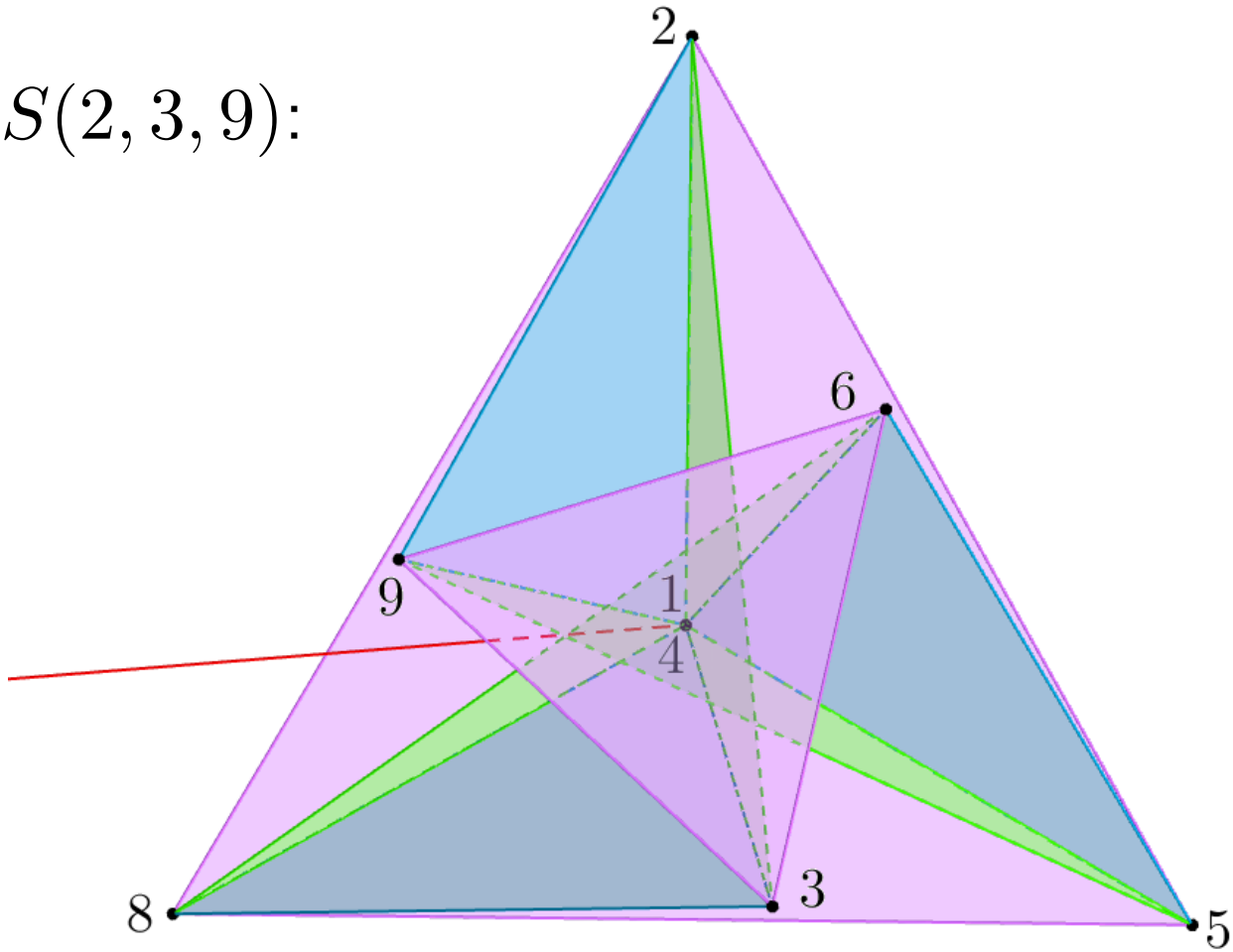
1 2 3	1 5 9
4 5 6	2 6 7
7 8 9	3 4 8
1 4 7	1 6 8
2 5 8	2 4 9
3 6 9	3 5 7



Contact Representations for Hypergraphs in 3D

The Steiner triple system $S(2, 3, 9)$:

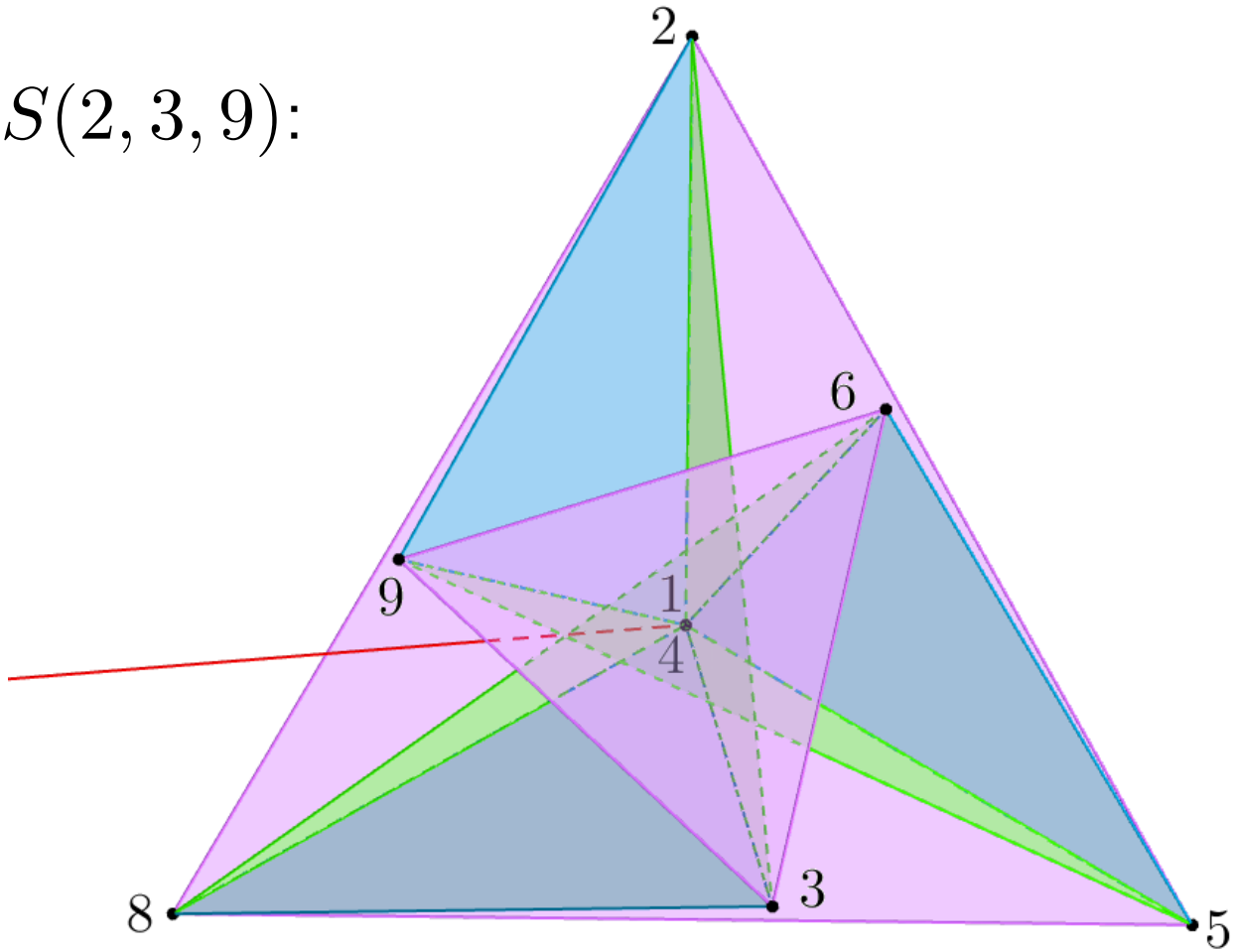
1 2 3	1 5 9
4 5 6	2 6 7
7 8 9	3 4 8
1 4 7	1 6 8
2 5 8	2 4 9
3 6 9	3 5 7



Contact Representations for Hypergraphs in 3D

The Steiner triple system $S(2, 3, 9)$:

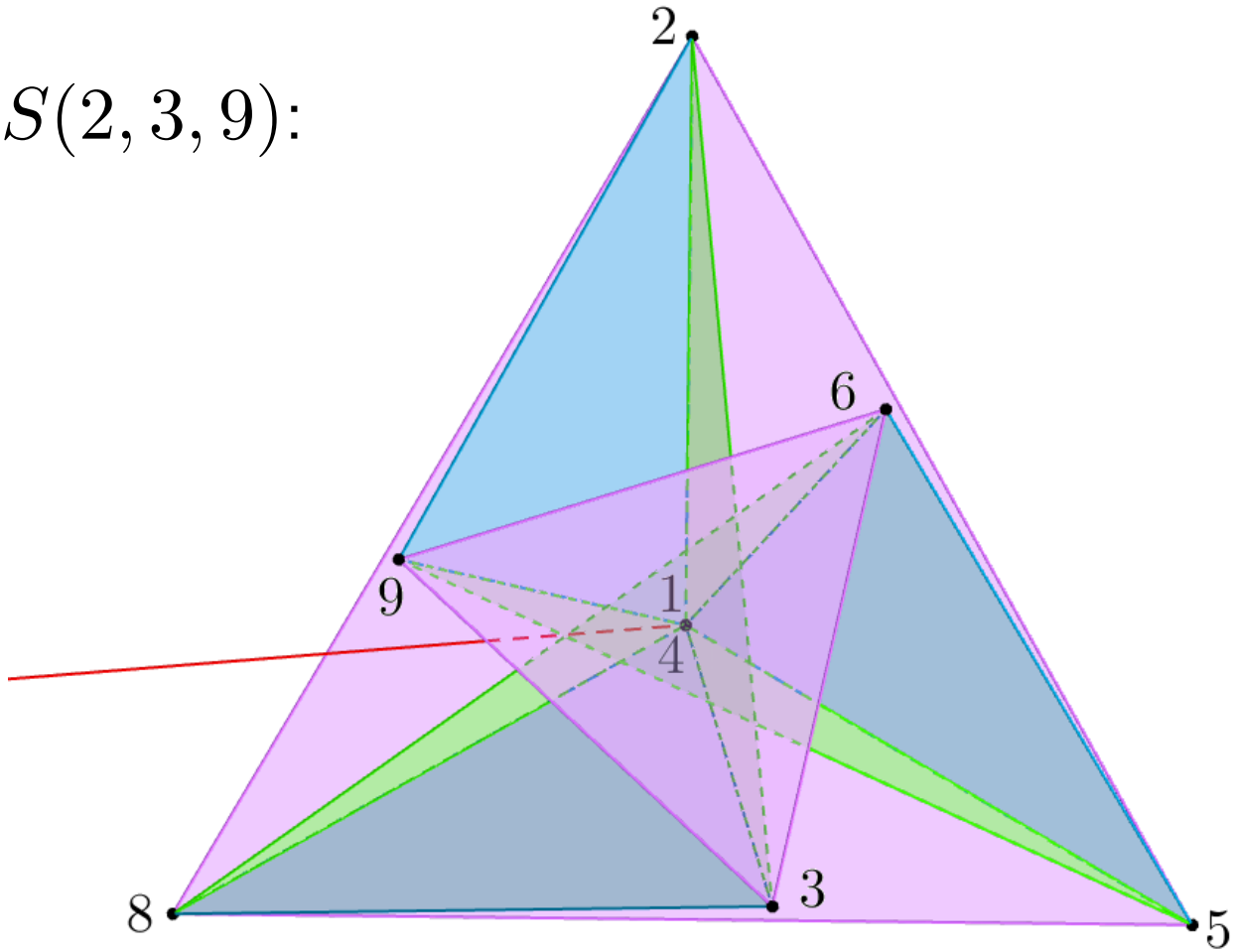
1 2 3	1 5 9
4 5 6	2 6 7
7 8 9	3 4 8
1 4 7	1 6 8
2 5 8	2 4 9
3 6 9	3 5 7

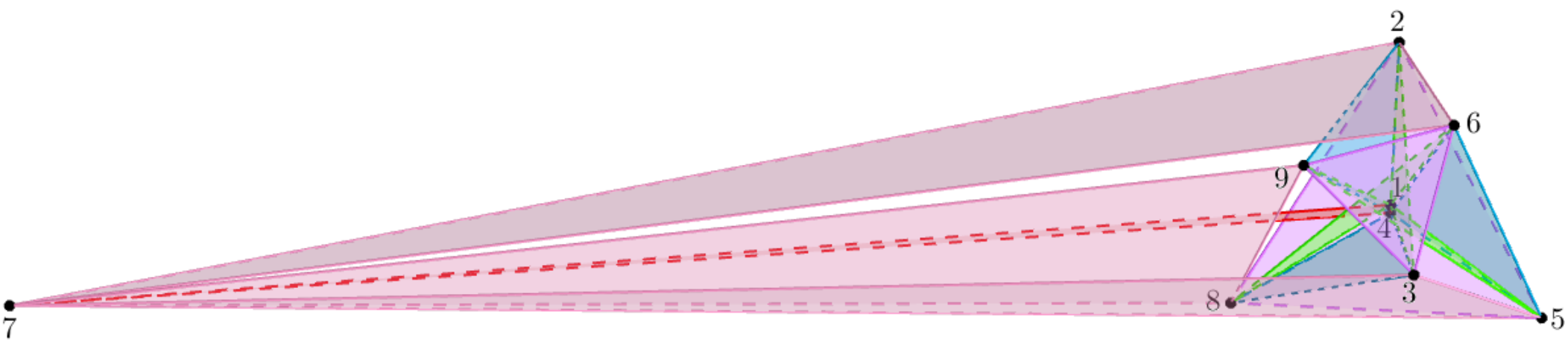


Contact Representations for Hypergraphs in 3D

The Steiner triple system $S(2, 3, 9)$:

1 2 3	1 5 9
4 5 6	2 6 7
7 8 9	3 4 8
1 4 7	1 6 8
2 5 8	2 4 9
3 6 9	3 5 7

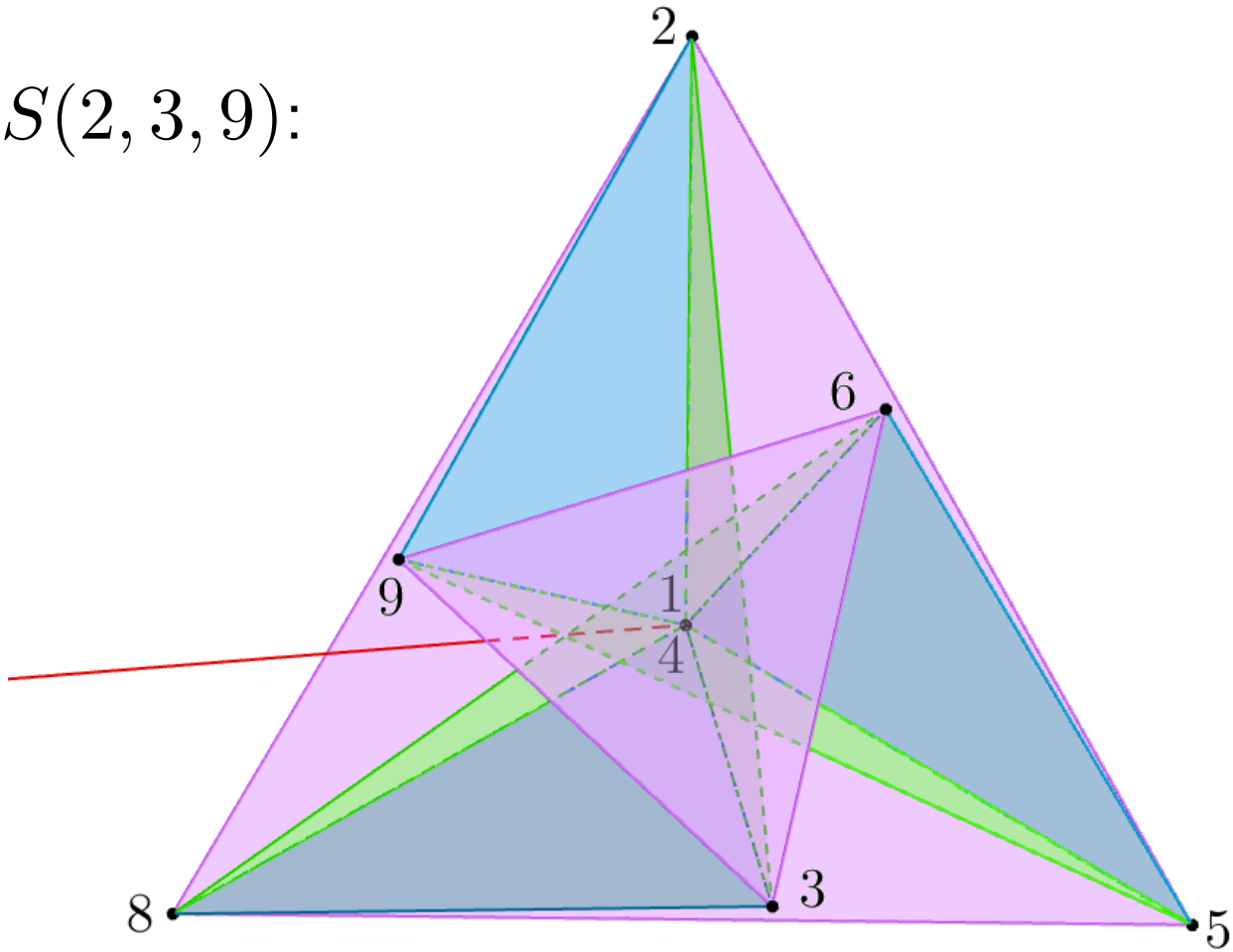




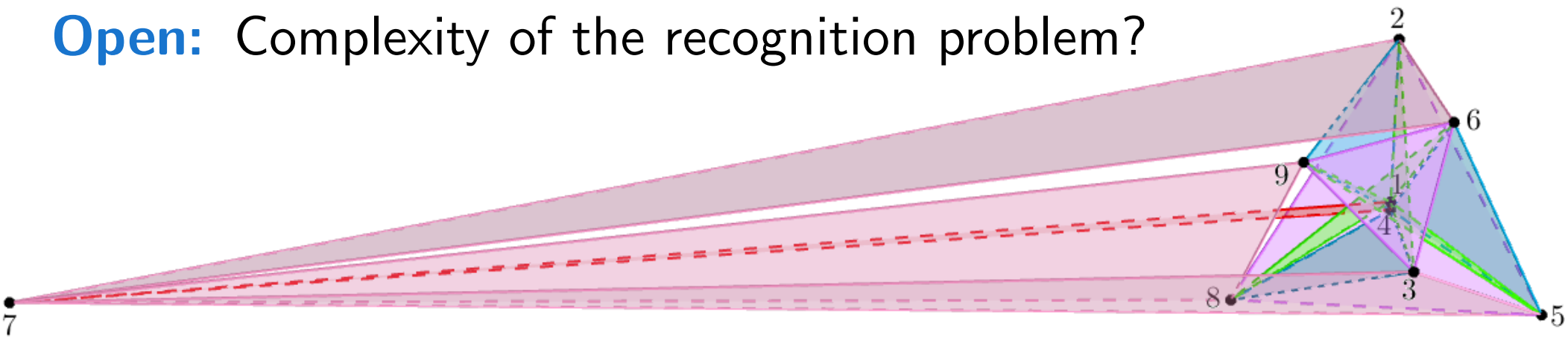
Contact Representations for Hypergraphs in 3D

The Steiner triple system $S(2, 3, 9)$:

1 2 3	1 5 9
4 5 6	2 6 7
7 8 9	3 4 8
1 4 7	1 6 8
2 5 8	2 4 9
3 6 9	3 5 7



Open: Complexity of the recognition problem?



Steiner Quadruple Systems

$S(3,4,8)$	
1 2 4 8	3 5 6 7
2 3 5 8	1 4 6 7
3 4 6 8	1 2 5 7
4 5 7 8	1 2 3 6
1 5 6 8	2 3 4 7
2 6 7 8	1 3 4 5
1 3 7 8	2 4 5 6

Steiner Quadruple Systems

$S(3,4,8)$	
1 2 4 8	3 5 6 7
2 3 5 8	1 4 6 7
3 4 6 8	1 2 5 7
4 5 7 8	1 2 3 6
1 5 6 8	2 3 4 7
2 6 7 8	1 3 4 5
1 3 7 8	2 4 5 6

Theorem. The Steiner quadruple system $S(3,4,8)$ does not admit a contact representation by quadrilaterals.

Steiner Quadruple Systems

$S(3,4,8)$	
1 2 4 8	3 5 6 7
2 3 5 8	1 4 6 7
3 4 6 8	1 2 5 7
4 5 7 8	1 2 3 6
1 5 6 8	2 3 4 7
2 6 7 8	1 3 4 5
1 3 7 8	2 4 5 6

$S(3,4,10)$		
1 2 4 5	1 2 3 7	1 3 5 8
2 3 5 6	2 3 4 8	2 4 6 9
3 4 6 7	3 4 5 9	3 5 7 0
4 5 7 8	4 5 6 0	1 4 6 8
5 6 8 9	1 5 6 7	2 5 7 9
6 7 9 0	2 6 7 8	3 6 8 0
1 7 8 0	3 7 8 9	1 4 7 9
1 2 8 9	4 8 9 0	2 5 8 0
2 3 9 0	1 5 9 0	1 3 6 9
1 3 4 0	1 2 6 0	2 4 7 0

Theorem. The Steiner quadruple system $S(3,4,8)$ does not admit a contact representation by quadrilaterals.

Steiner Quadruple Systems

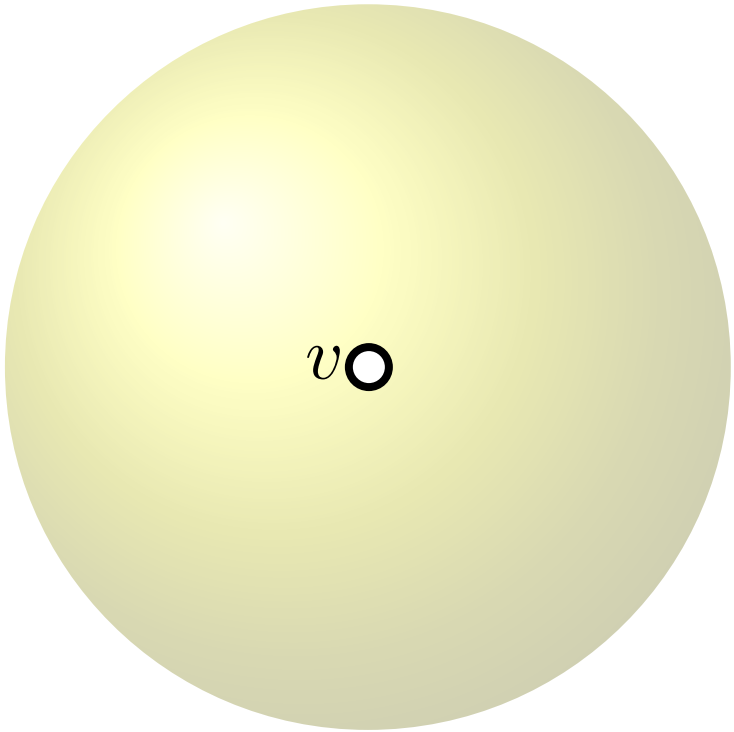
$S(3,4,8)$	
1 2 4 8	3 5 6 7
2 3 5 8	1 4 6 7
3 4 6 8	1 2 5 7
4 5 7 8	1 2 3 6
1 5 6 8	2 3 4 7
2 6 7 8	1 3 4 5
1 3 7 8	2 4 5 6

$S(3,4,10)$		
1 2 4 5	1 2 3 7	1 3 5 8
2 3 5 6	2 3 4 8	2 4 6 9
3 4 6 7	3 4 5 9	3 5 7 0
4 5 7 8	4 5 6 0	1 4 6 8
5 6 8 9	1 5 6 7	2 5 7 9
6 7 9 0	2 6 7 8	3 6 8 0
1 7 8 0	3 7 8 9	1 4 7 9
1 2 8 9	4 8 9 0	2 5 8 0
2 3 9 0	1 5 9 0	1 3 6 9
1 3 4 0	1 2 6 0	2 4 7 0

Theorem. The Steiner quadruple system $S(3,4,8)$ does not admit a contact representation by quadrilaterals.

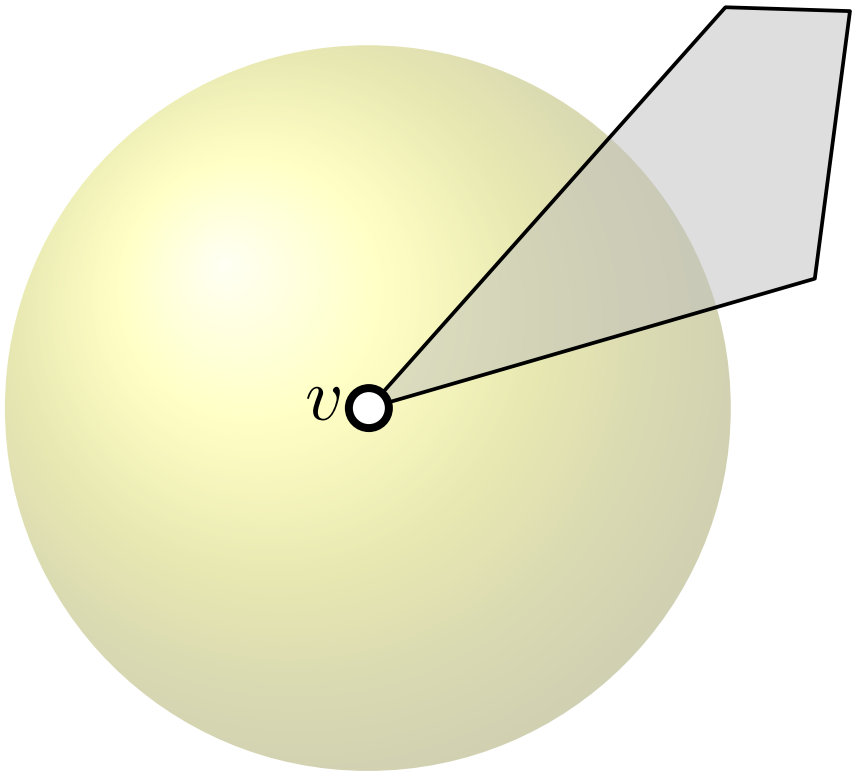
Theorem. No Steiner quadruple system admits a contact representation by *convex* quadrilaterals.

Steiner Quadruple Systems



Theorem. No Steiner quadruple system admits a contact representation by *convex* quadrilaterals.

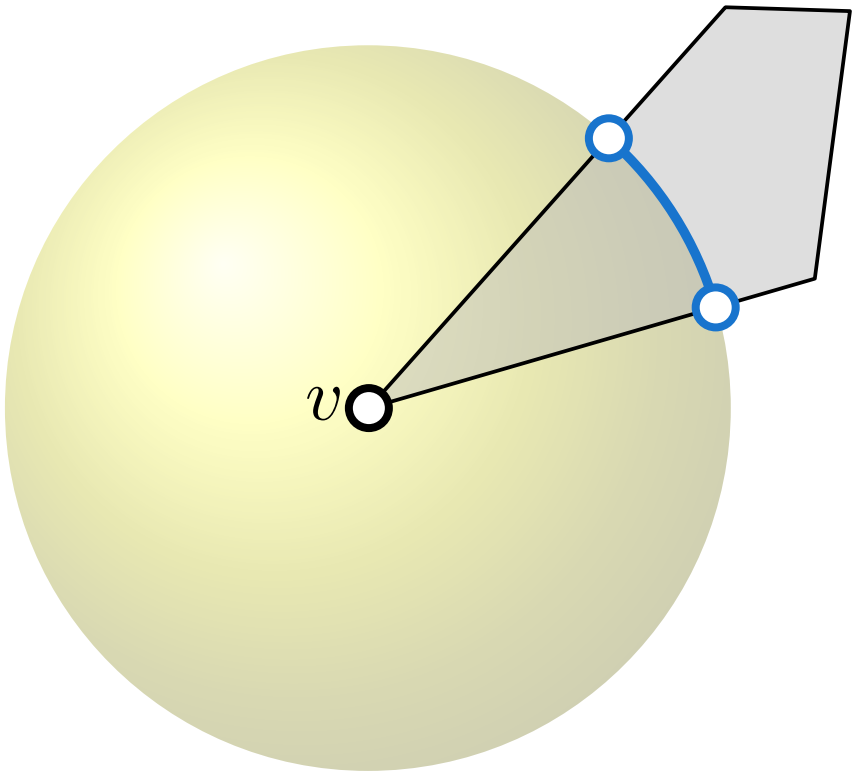
Steiner Quadruple Systems



Theorem. No Steiner quadruple system admits a contact representation by *convex* quadrilaterals.

Steiner Quadruple Systems

Dey and Edelsbrunner [DCG'94]

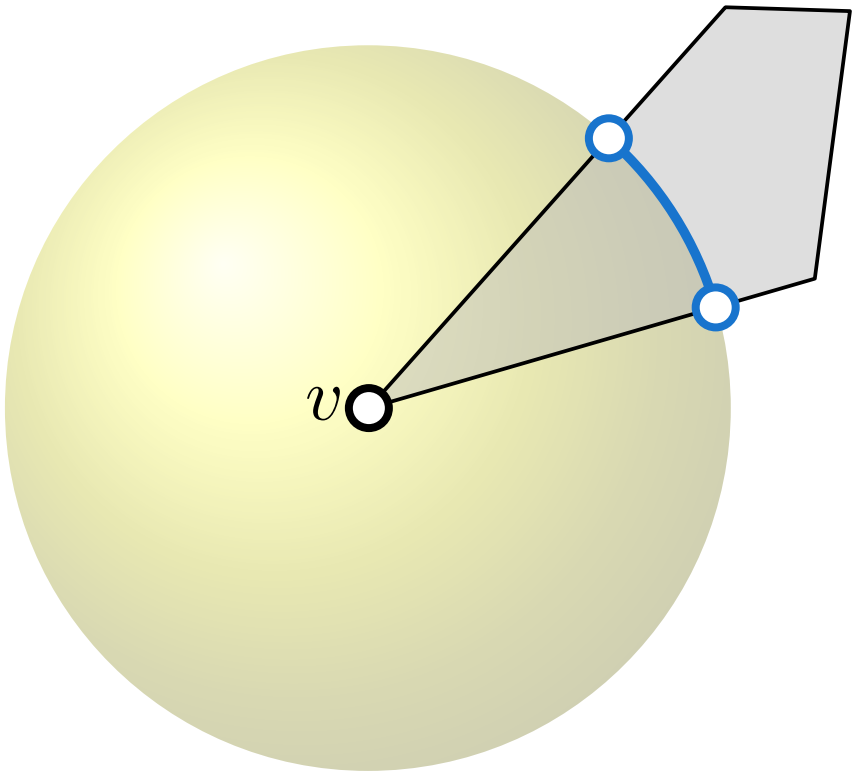


Link graph of v:

Theorem. No Steiner quadruple system admits a contact representation by *convex* quadrilaterals.

Steiner Quadruple Systems

Dey and Edelsbrunner [DCG'94]



Link graph of v:

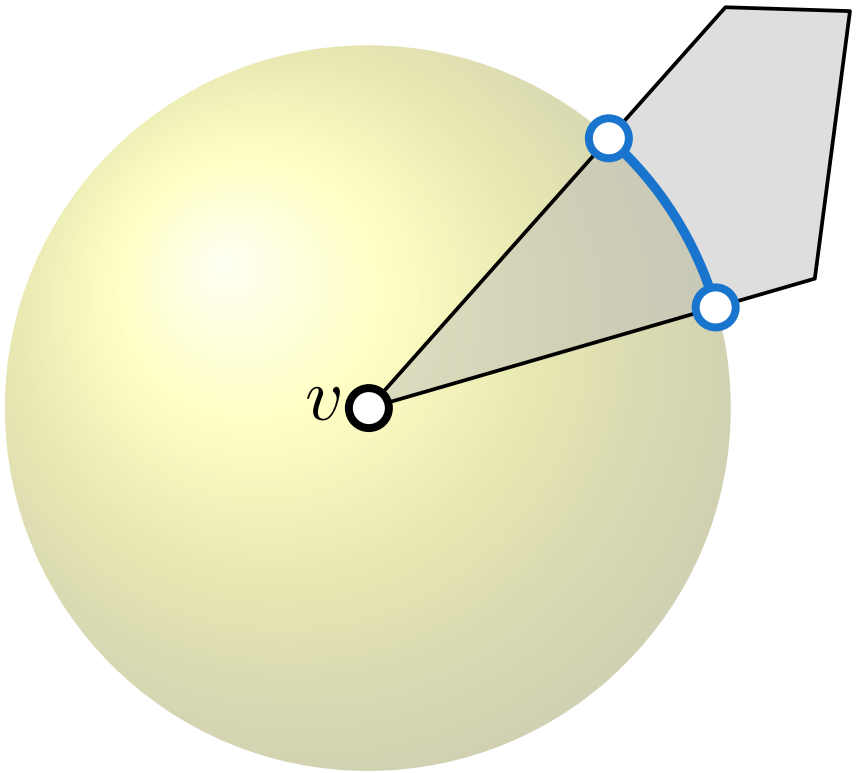
– edge for each hyperedge

– vertex for each vtx $\neq v$

Theorem. No Steiner quadruple system admits a contact representation by *convex* quadrilaterals.

Steiner Quadruple Systems

Dey and Edelsbrunner [DCG'94]



Link graph of v:

– edge for each hyperedge

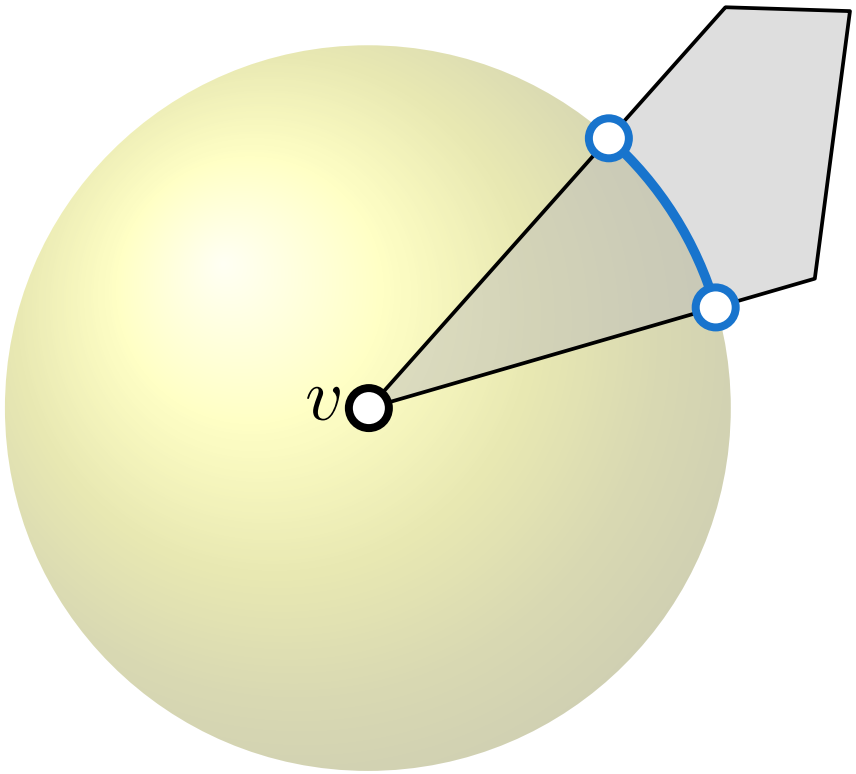
– vertex for each vtx $\neq v$

must be *planar*!

Theorem. No Steiner quadruple system admits a contact representation by *convex* quadrilaterals.

Steiner Quadruple Systems

Dey and Edelsbrunner [DCG'94]



Link graph of v:

– edge for each hyperedge

– vertex for each vtx $\neq v$

must be *planar*!

Here:

$\#$ vertices $= n - 1$

$\#$ edges $= (n-1)(n-2)/6$

Trick: split quadrilaterals!

$\Rightarrow$ $\#$ edges $= (n-1)(n-2)/3$

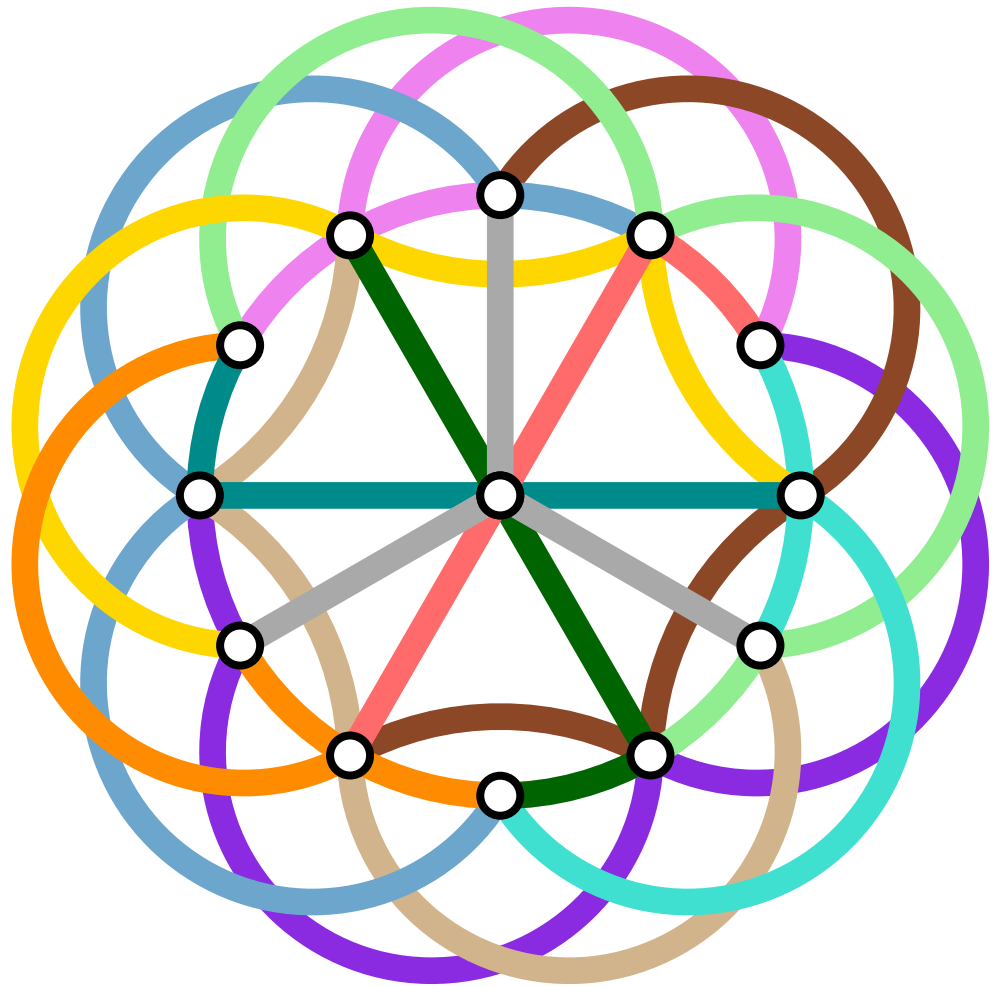
$\Rightarrow$ link graph not planar for $n > 8$.

Theorem. No Steiner quadruple system admits a contact representation by *convex* quadrilaterals.

Discrete Projective Plane

Open: Can the second smallest projective plane, $PG(3)$ $(= S(2, 4, 13))$, be represented by touching (convex) quadrilaterals?

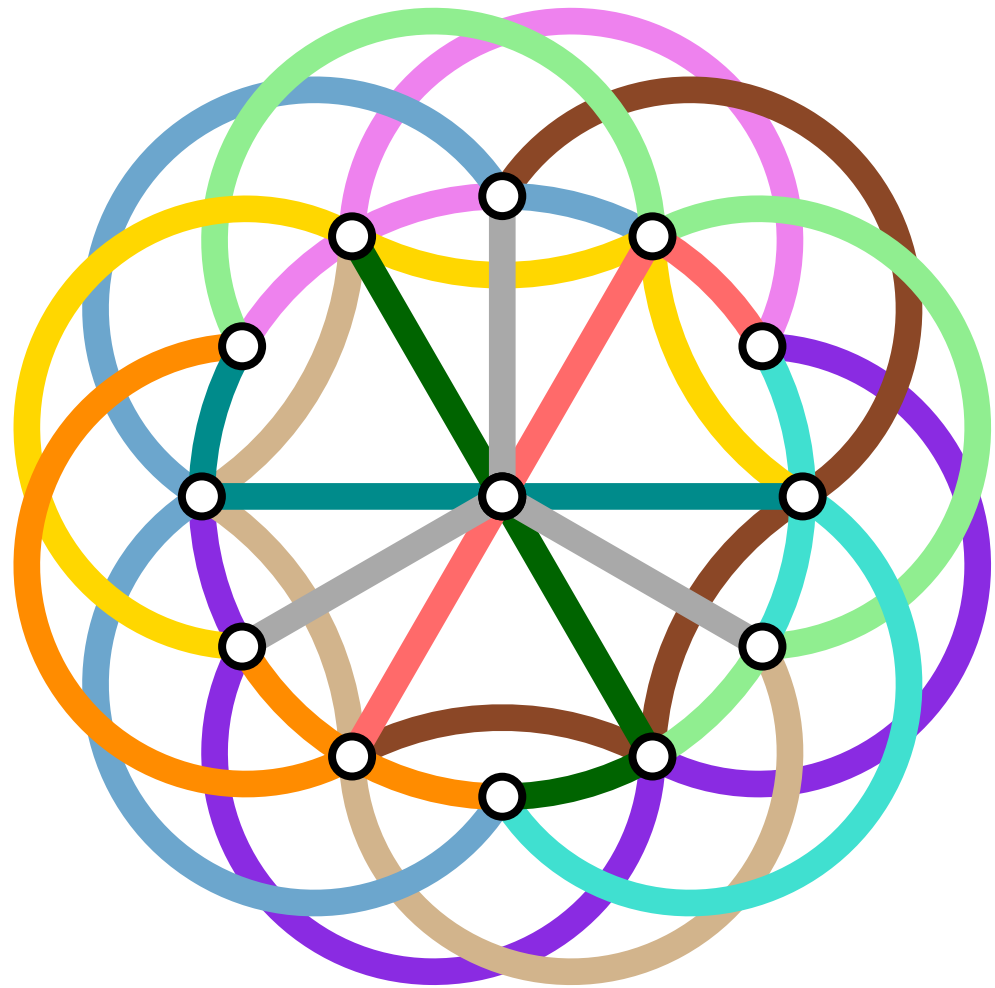
A	B	C	D
A	1	2	3
A	4	5	6
A	7	8	9
B	1	4	7
B	2	5	8
B	3	6	9
C	1	5	9
C	2	6	7
C	3	4	8
D	1	6	8
D	2	4	9
D	3	5	7



Discrete Projective Plane

Open: Can the second smallest projective plane, $PG(3)$ $(= S(2, 4, 13))$, be represented by touching
– (convex) quadrilaterals?
– tetrahedra?

A	B	C	D
A	1	2	3
A	4	5	6
A	7	8	9
B	1	4	7
B	2	5	8
B	3	6	9
C	1	5	9
C	2	6	7
C	3	4	8
D	1	6	8
D	2	4	9
D	3	5	7



Final words

Final words

- Many important topics had to be left out ...

Final words

- Many important topics had to be left out ...
  - crossing numbers
  - clustered planarity
  - labeling
  - beyond planar graphs
  - right-angle-crossing drawings
  - universal point sets
  - topological drawings
  - representation as contact/intersection graphs
  - layered drawings
  - bus drawings
  - more subdivision drawings for hypergraphs
  - ...

Final words

- Many important topics had to be left out ...
  - crossing numbers
  - clustered planarity
  - labeling
  - beyond planar graphs
  - right-angle-crossing drawings
  - universal point sets
  - topological drawings
  - representation as contact/intersection graphs
  - layered drawings
  - bus drawings
  - more subdivision drawings for hypergraphs
  - ...
- So I hope to talk to you more about Graph Drawing

Final words

- Many important topics had to be left out ...
  - crossing numbers
  - clustered planarity
  - labeling
  - beyond planar graphs
  - right-angle-crossing drawings
  - universal point sets
  - topological drawings
  - representation as contact/intersection graphs
  - layered drawings
  - bus drawings
  - more subdivision drawings for hypergraphs
  - ...

- So I hope to talk to you more about Graph Drawing – maybe at the Winter School in Teheran next year?