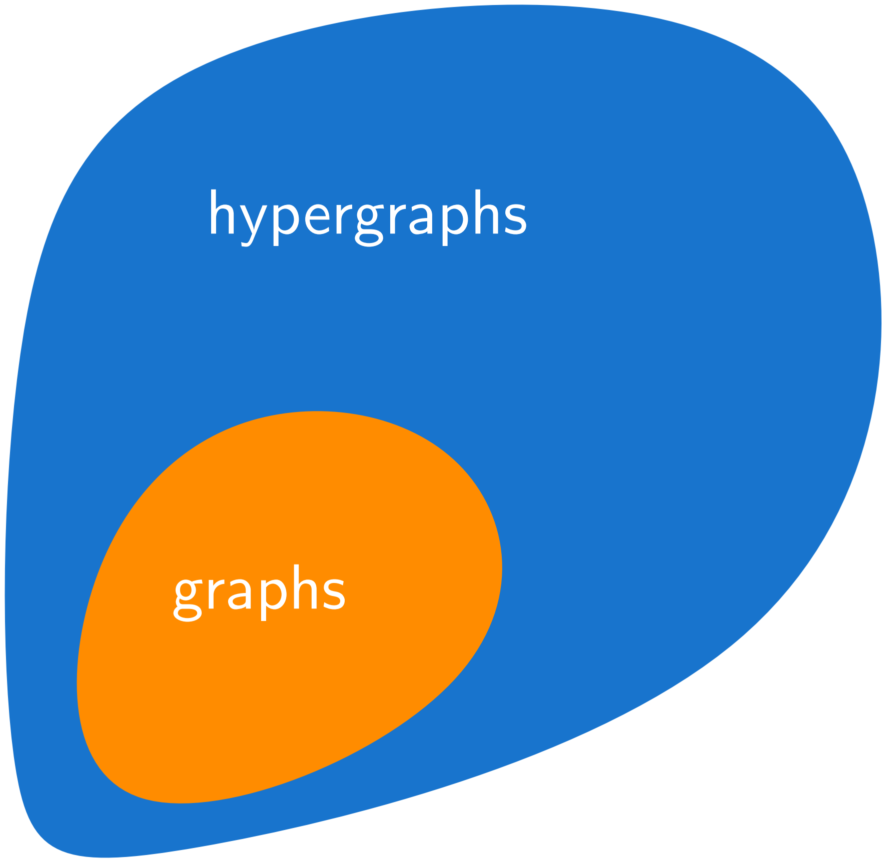
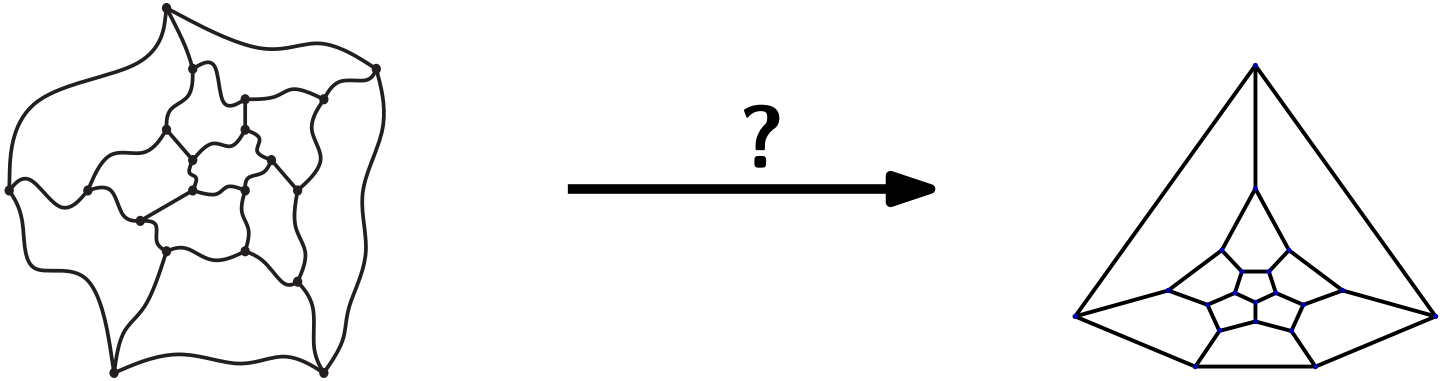


Drawing Graphs and Hypergraphs in 2D & 3D

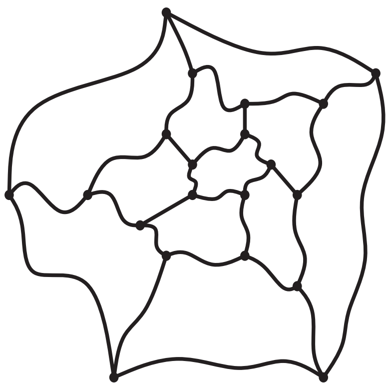
Alexander Wolff @ ICCG 2020

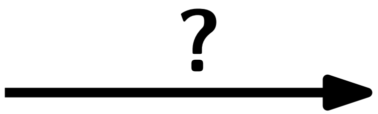
What is graph drawing?

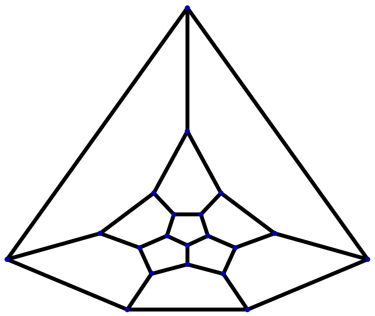
What is graph drawing?



What is graph drawing?



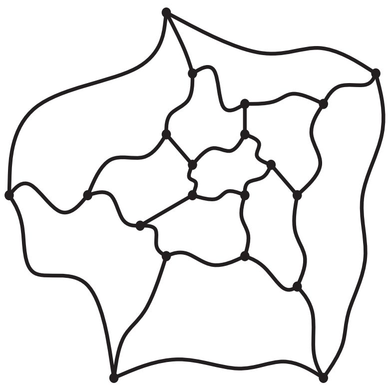




- abstract (combinatorial) graph

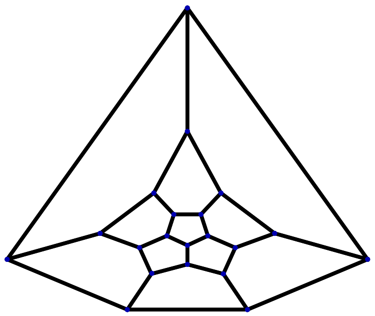
- drawing (e.g. node-link diagram)

What is graph drawing?



?

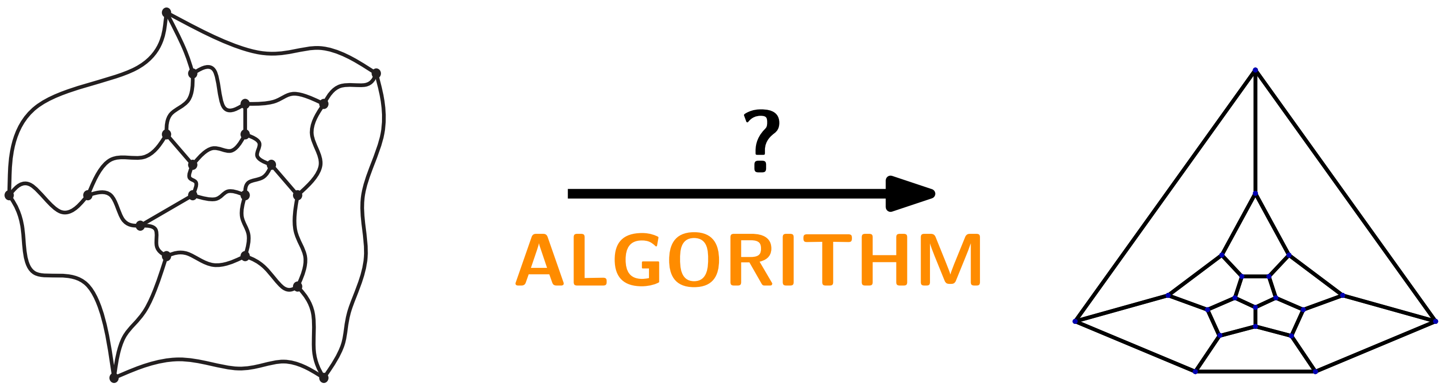
ALGORITHM



- abstract (combinatorial) graph

- drawing
  (e.g. node-link diagram)

What is graph drawing?

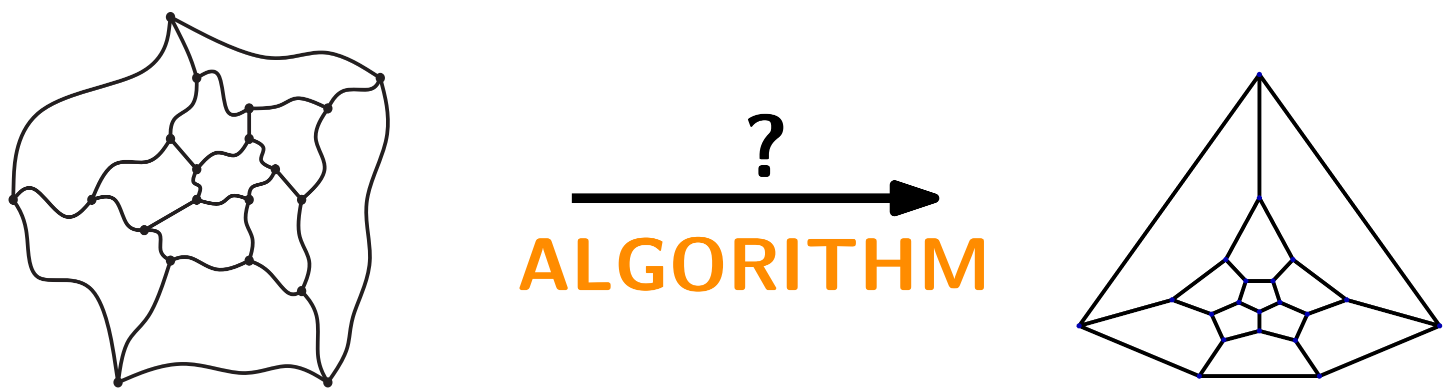


- abstract (combinatorial) graph

- drawing (e.g. node-link diagram)

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

What is graph drawing?



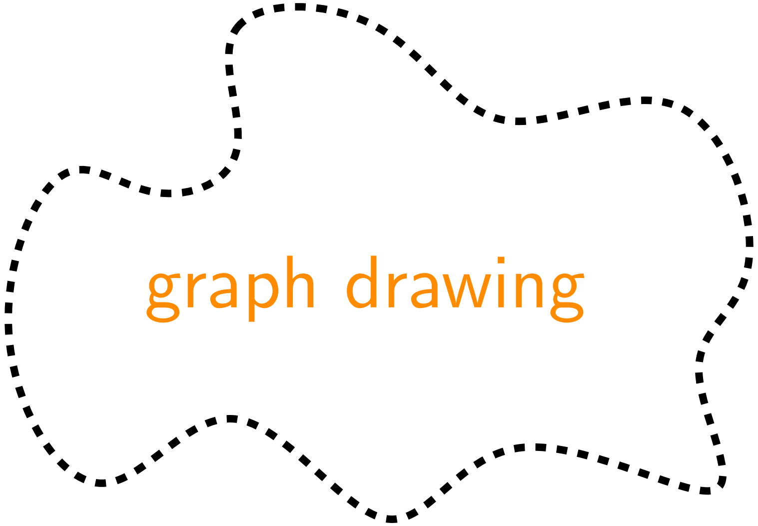
- abstract (combinatorial) graph

- drawing (e.g. node-link diagram)

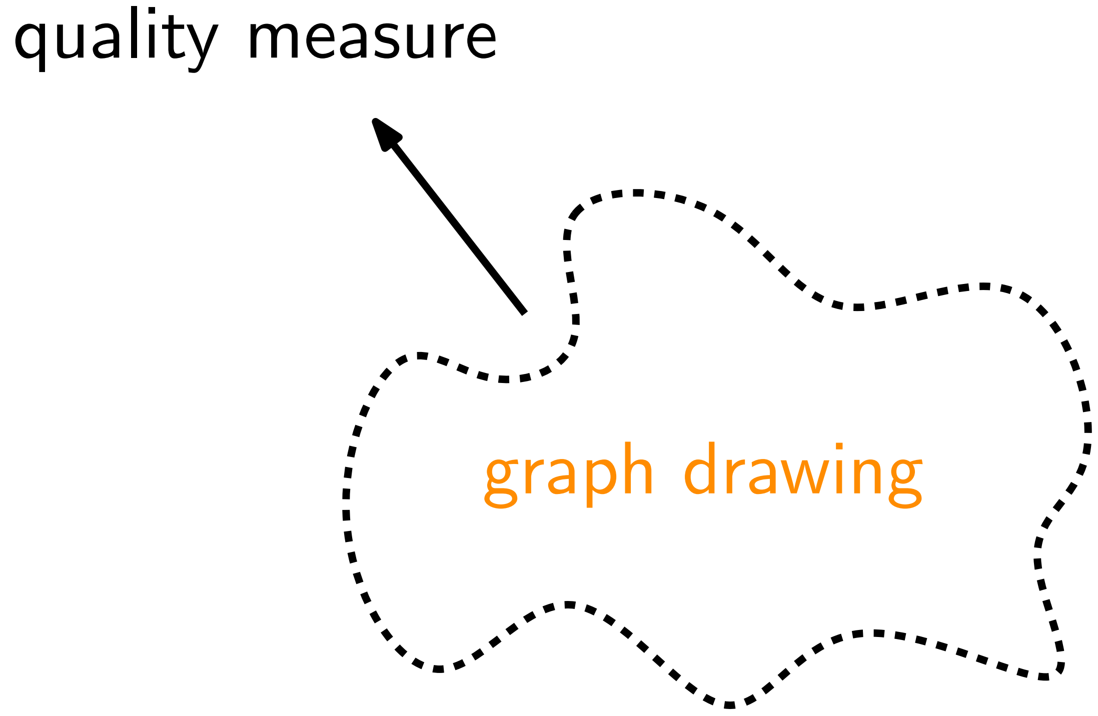
Goal: Algorithm guarantees a (provable) geometric quality measure in the worst case

Evaluation is not task-driven

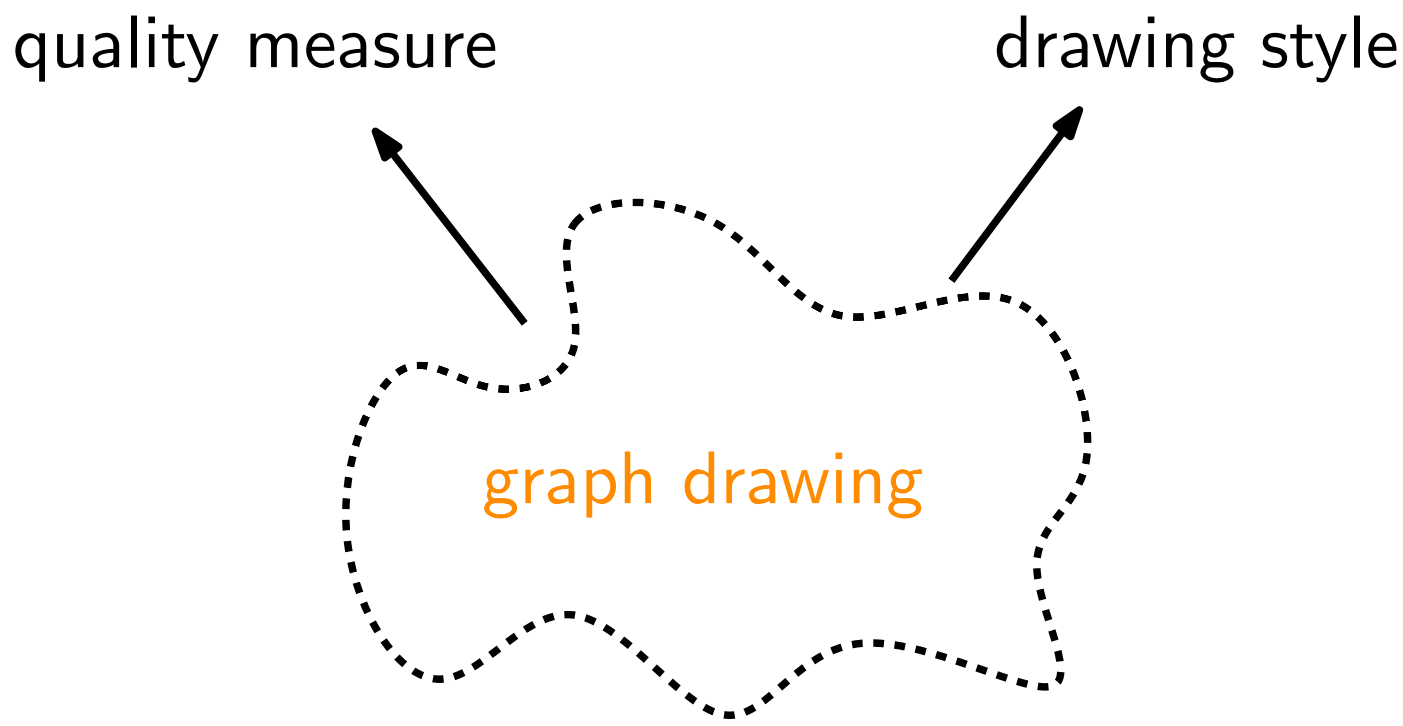
The many dimensions of graph drawing



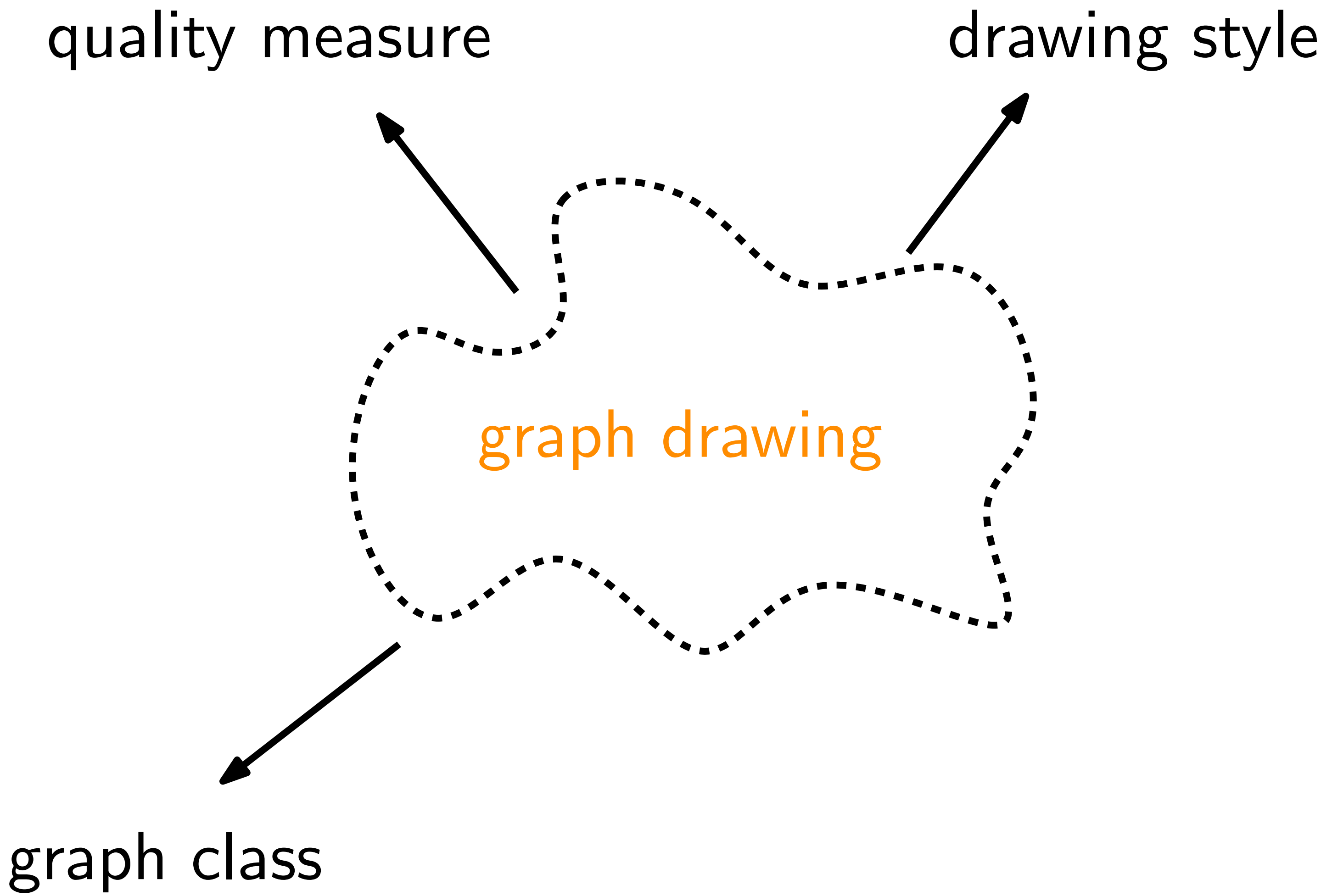
The many dimensions of graph drawing



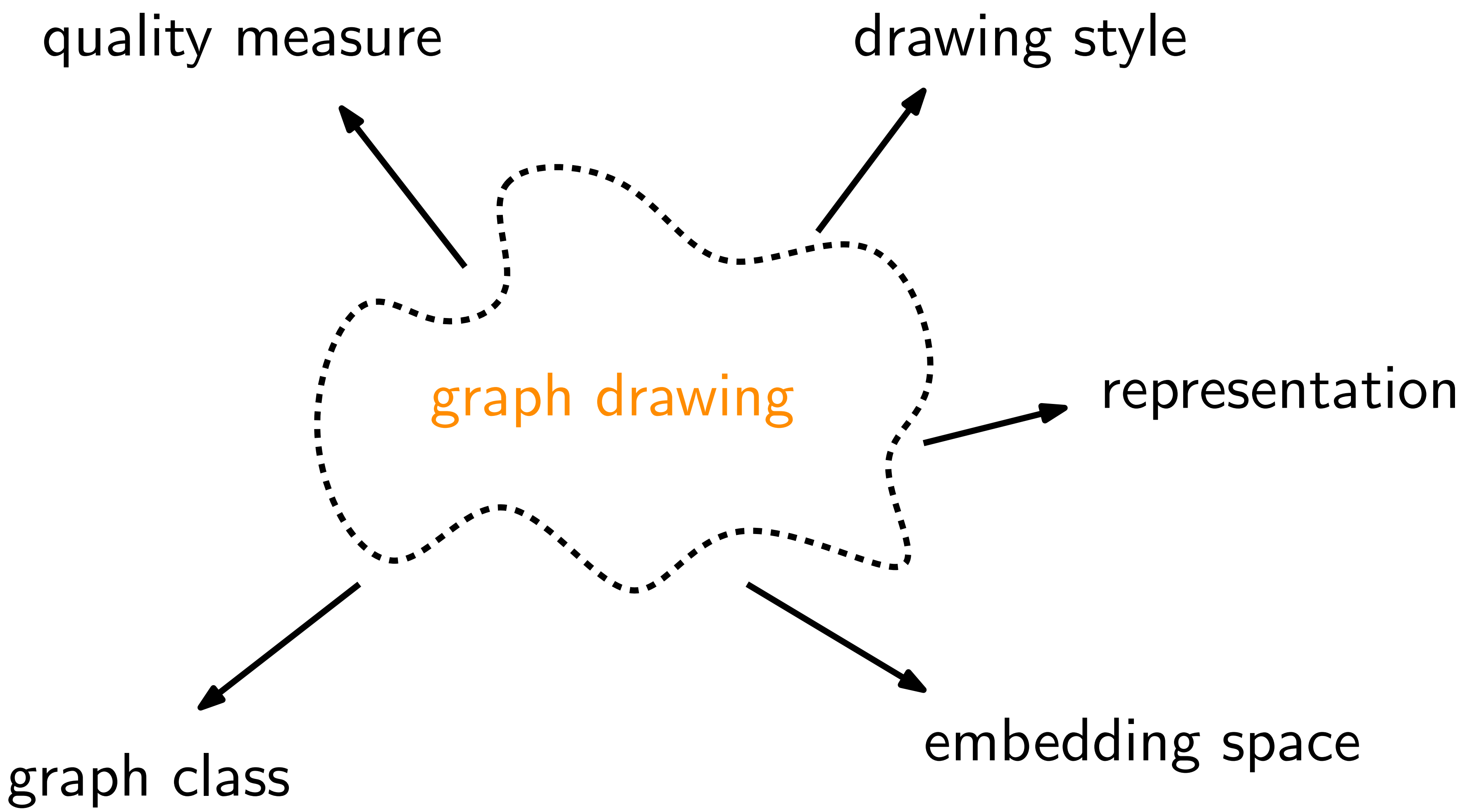
The many dimensions of graph drawing



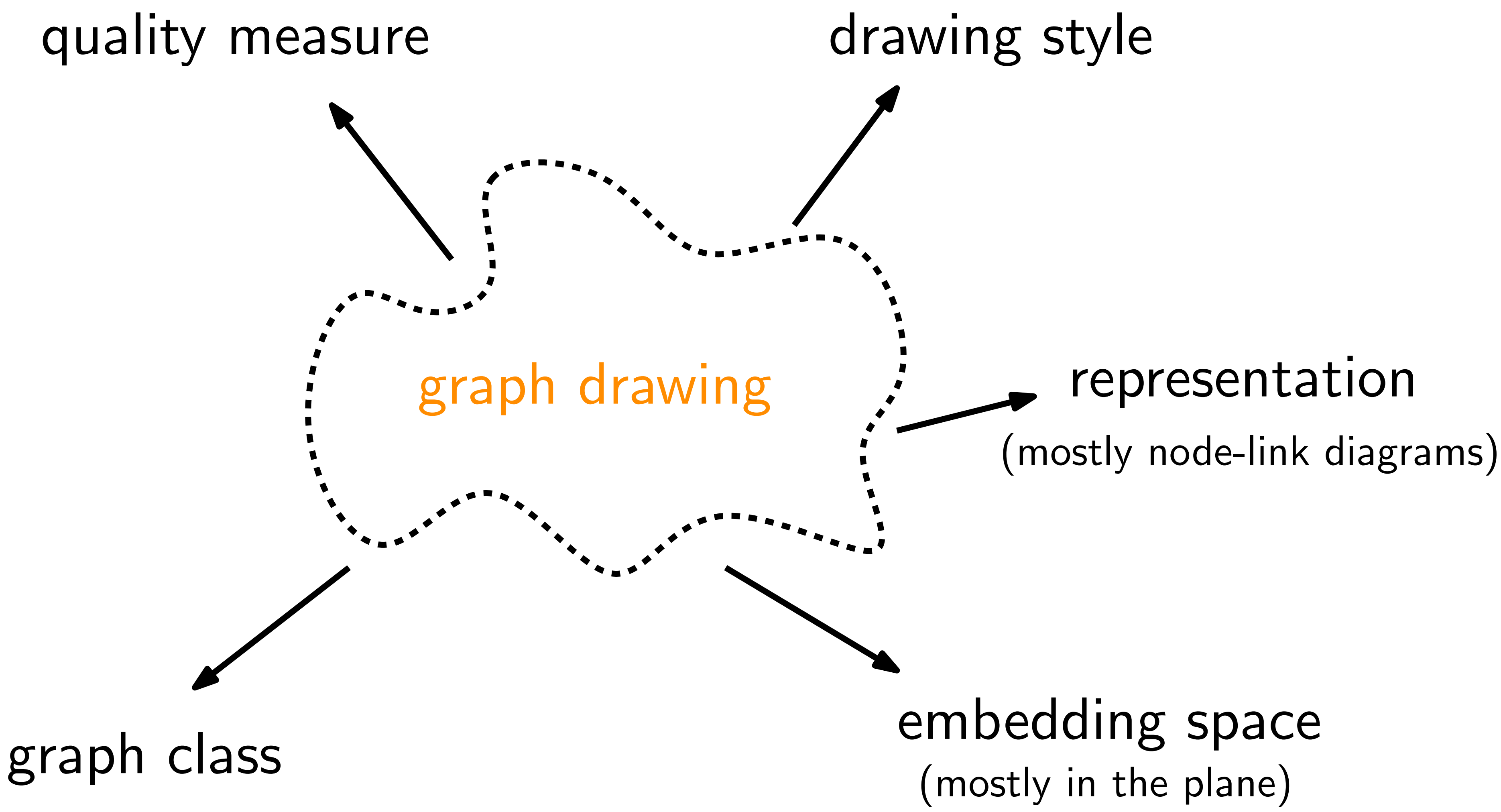
The many dimensions of graph drawing



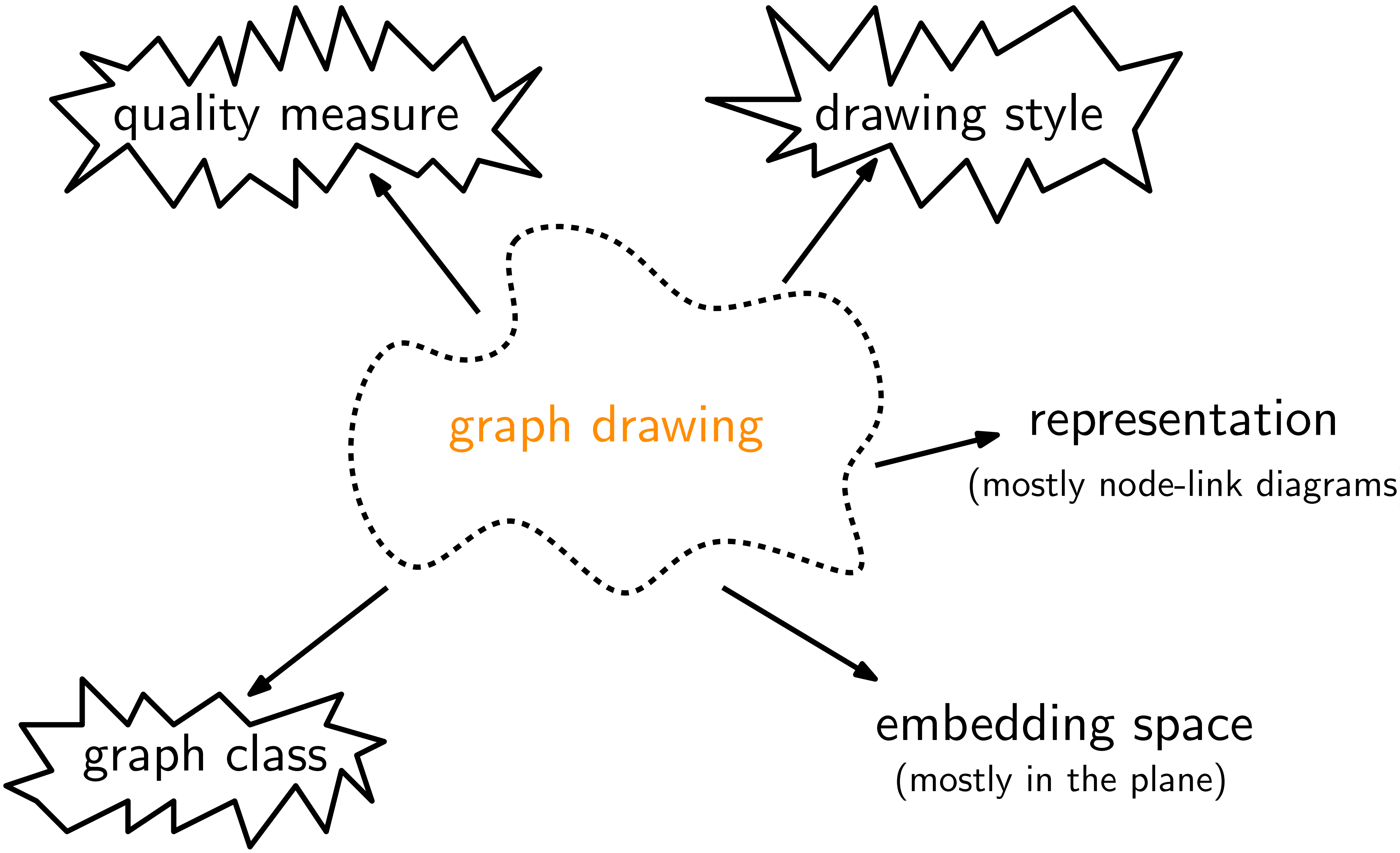
The many dimensions of graph drawing



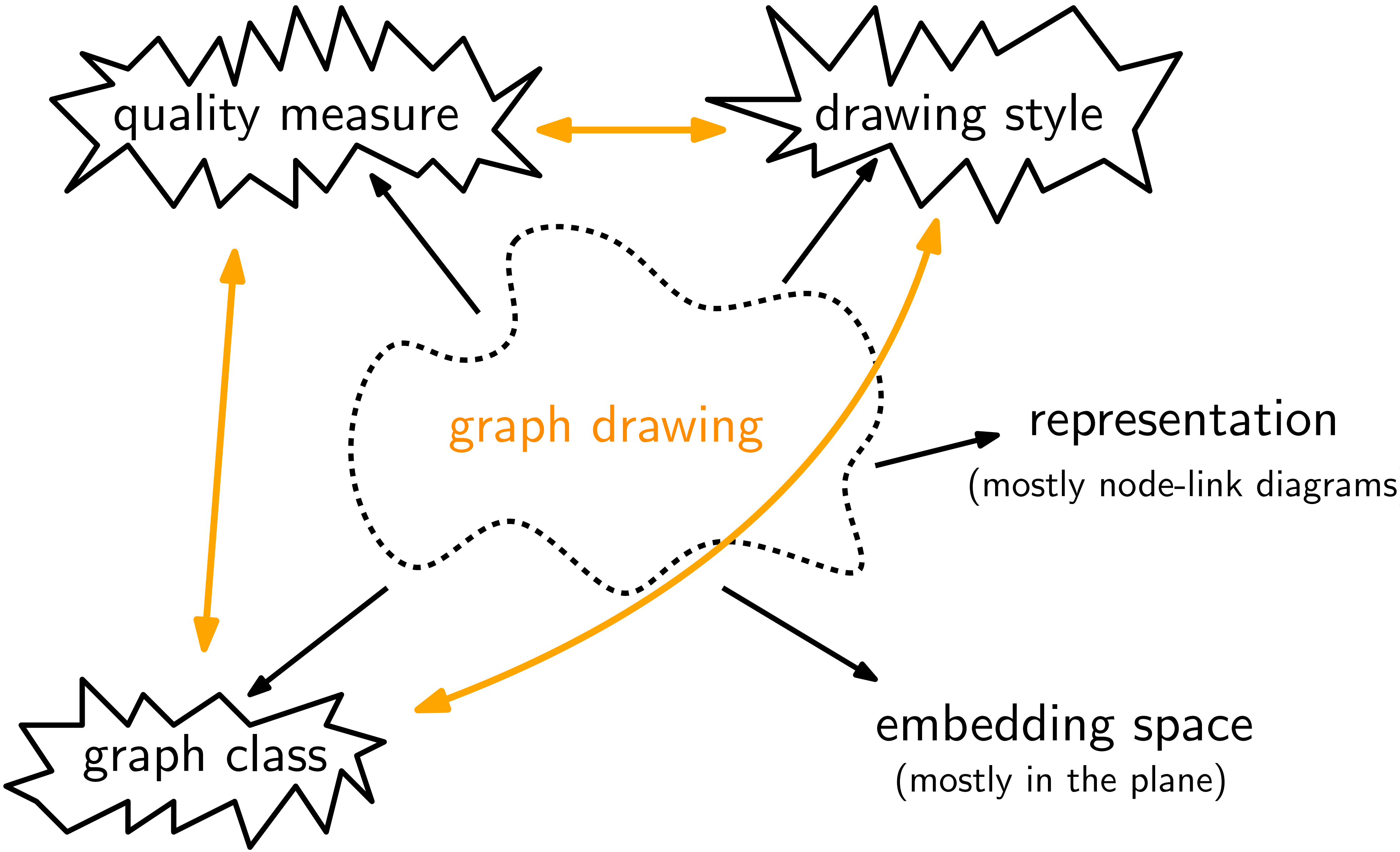
The many dimensions of graph drawing



The many dimensions of graph drawing

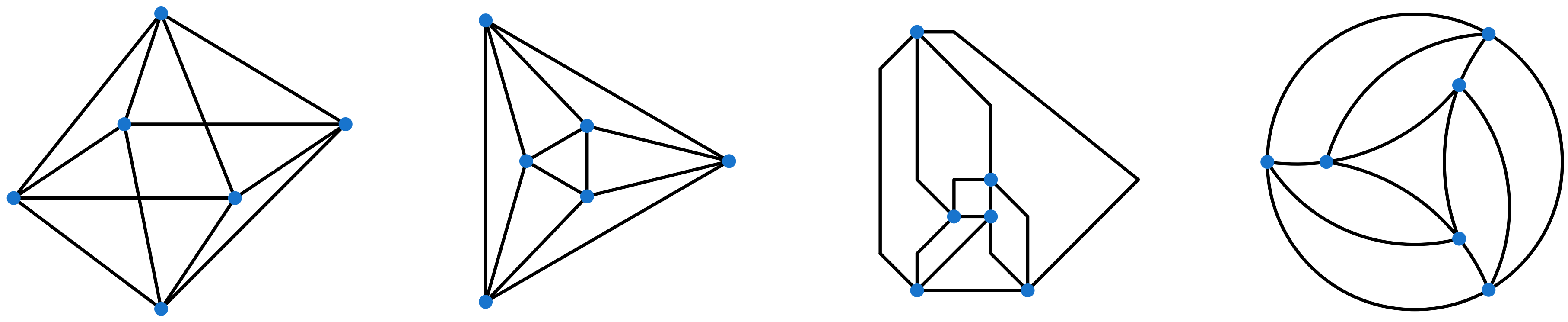


The many dimensions of graph drawing

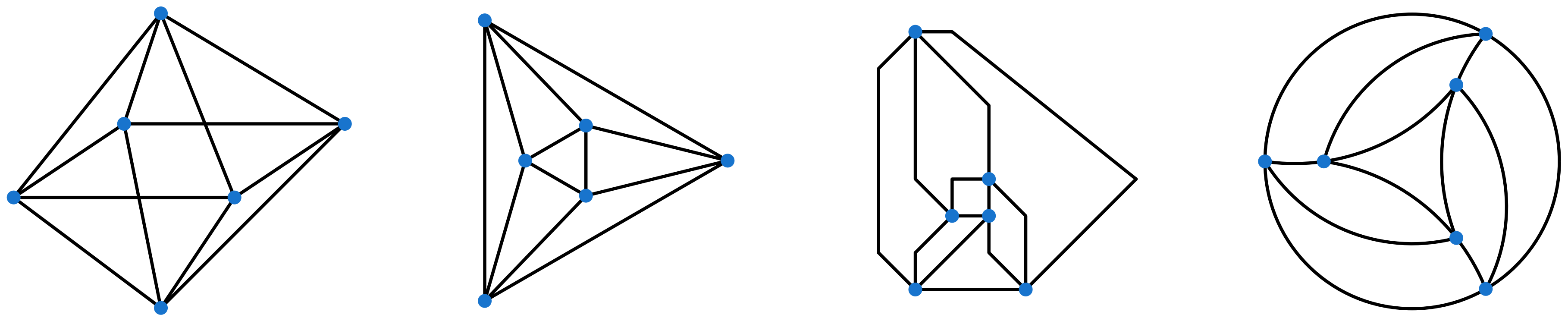


Drawing Styles

Drawing Styles

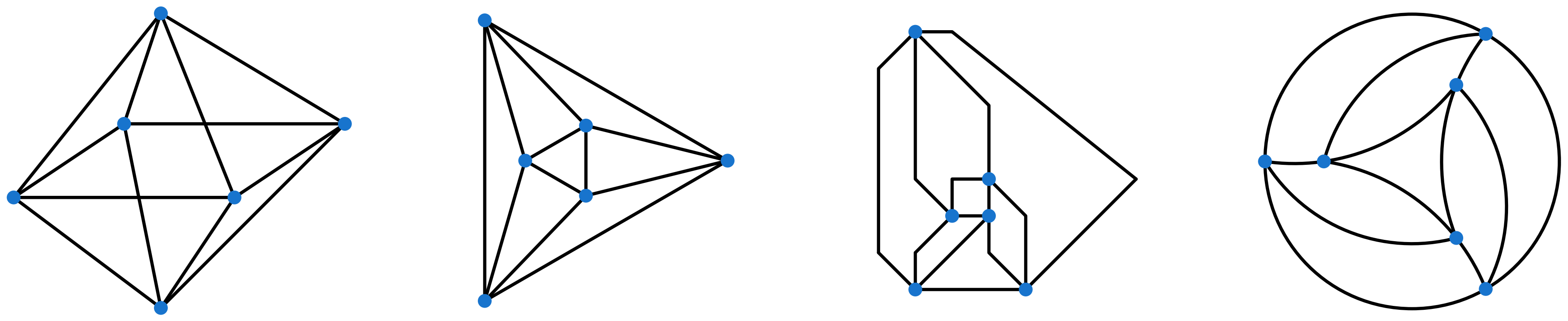


Drawing Styles



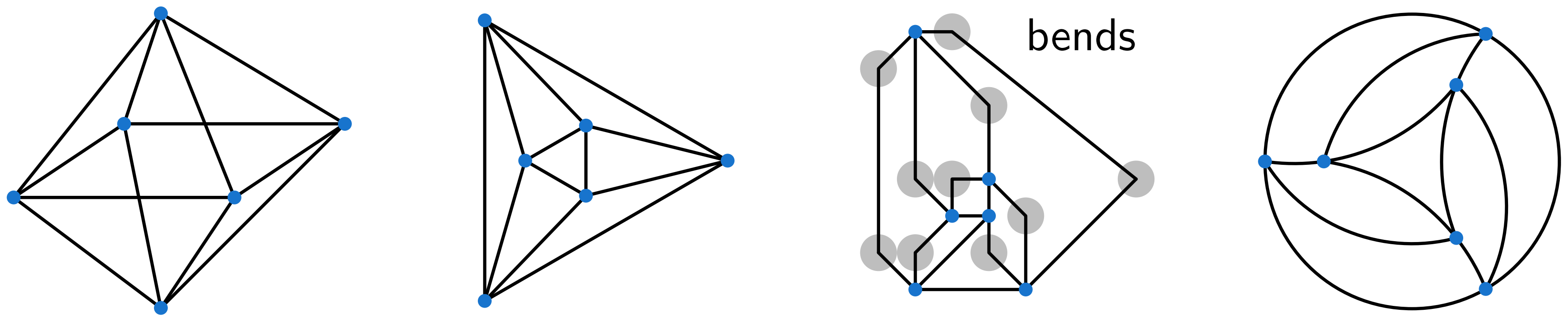
- straight-line vs. curved

Drawing Styles



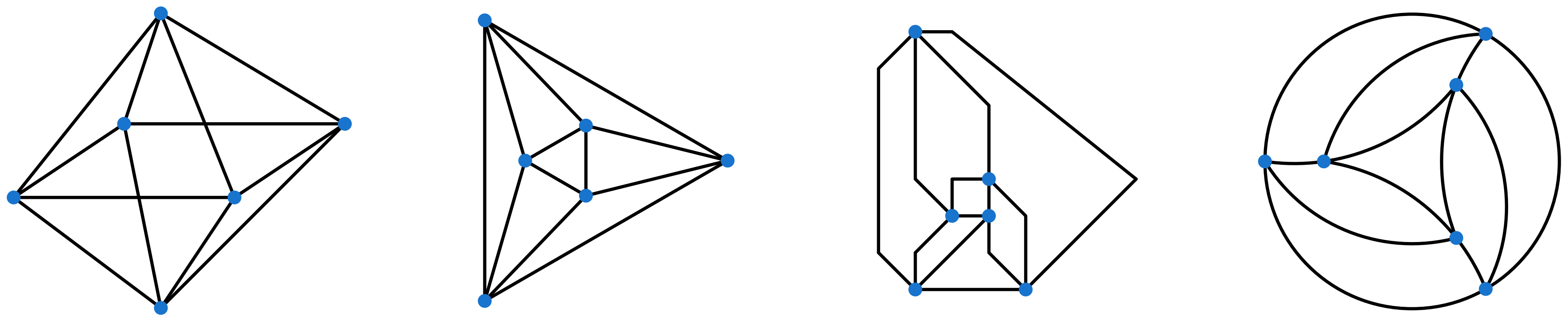
- straight-line vs. curved
- straight-line vs. polyline

Drawing Styles



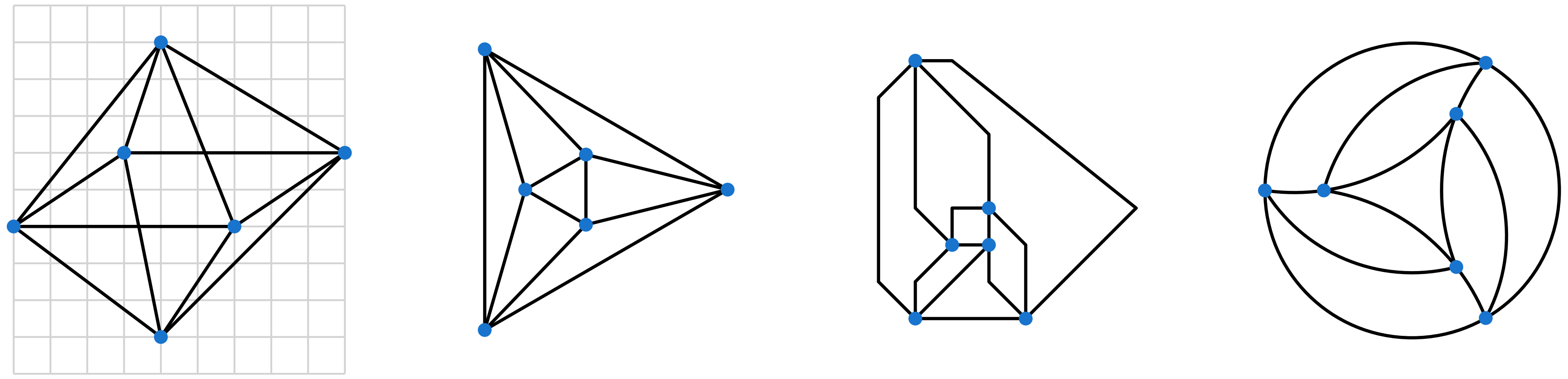
- straight-line vs. curved
- straight-line vs. polyline

Drawing Styles



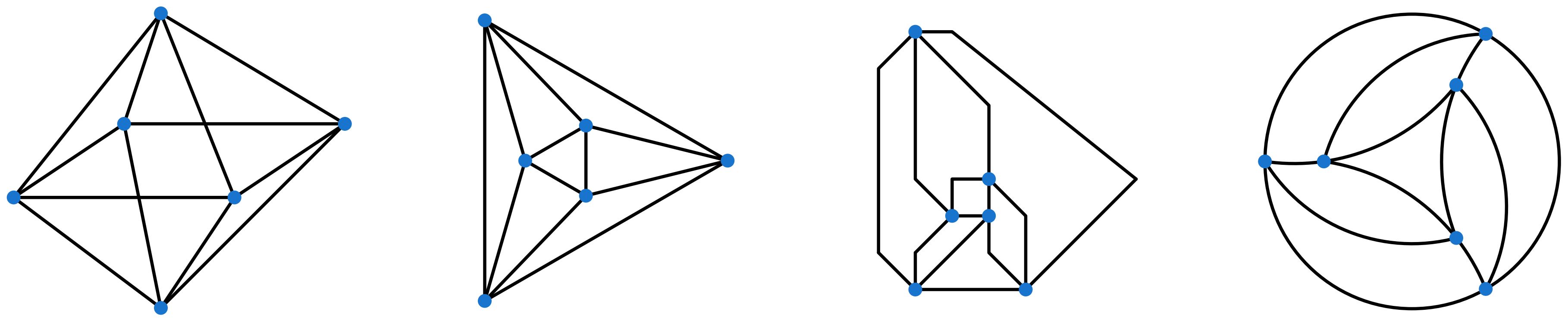
- straight-line vs. curved
- straight-line vs. polyline
- restricted slopes

Drawing Styles



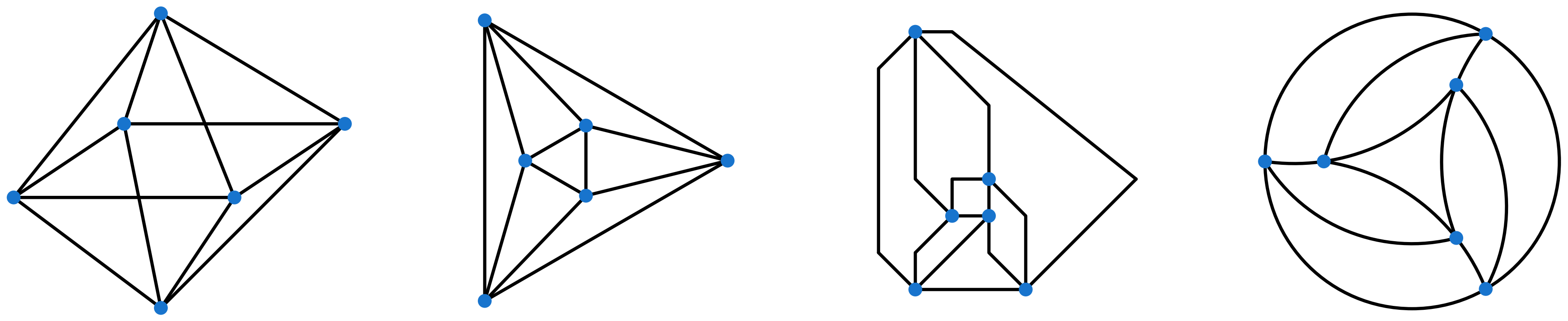
- straight-line vs. curved
- straight-line vs. polyline
- restricted slopes
- restricted to grid points

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, ...

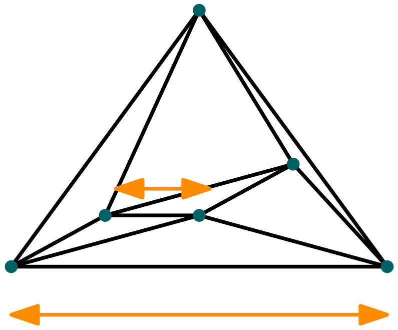
Classical Measures

Classical Measures

- vertex resolution

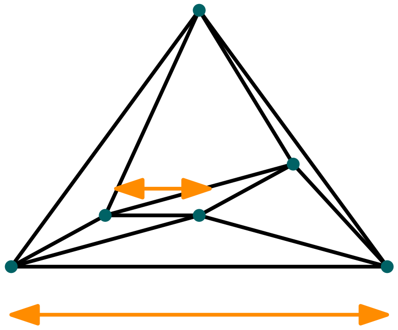
Classical Measures

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



Classical Measures

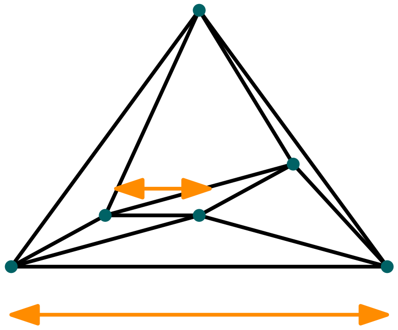
- vertex resolution $= \dfrac{\text{maximal distance between two vertices}}{\text{minimal distance between two vertices}}$



goal: small vertex resolution

Classical Measures

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

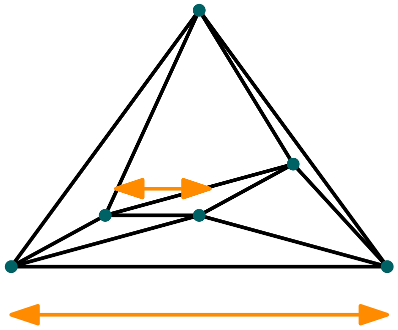


goal: small vertex resolution

- angular resolution

Classical Measures

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

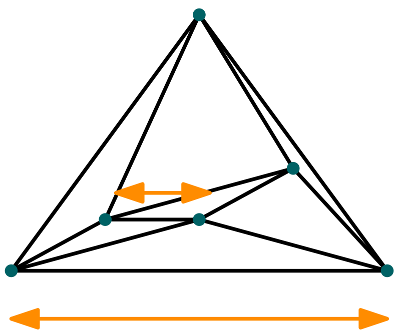


goal: small vertex resolution

- angular resolution $=$ size of the smallest angle

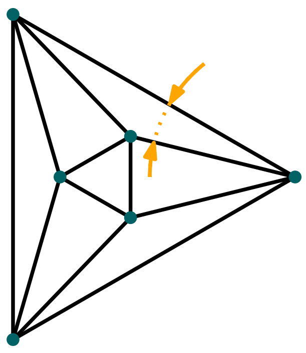
Classical Measures

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



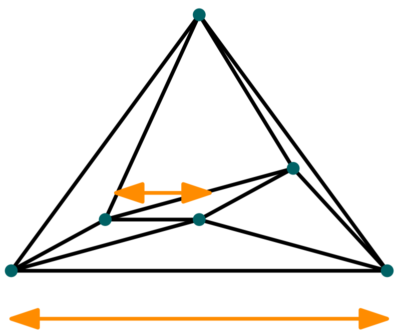
goal: small vertex resolution

- angular resolution $=$ size of the smallest angle



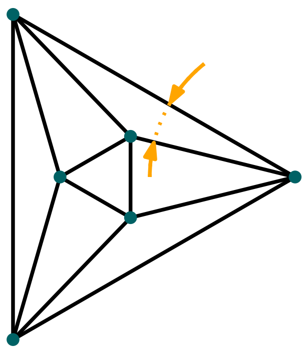
Classical Measures

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



goal: small vertex resolution

- angular resolution $=$ size of the smallest angle



goal: large vertex resolution

More Measures

More Measures

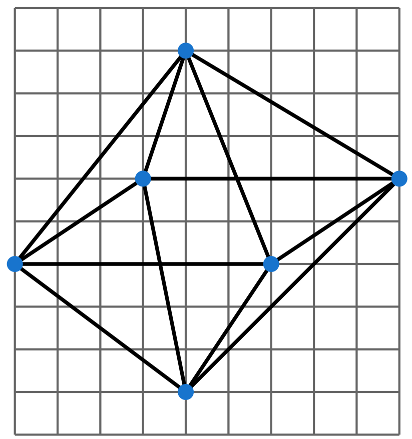
- grid size

More Measures

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

More Measures

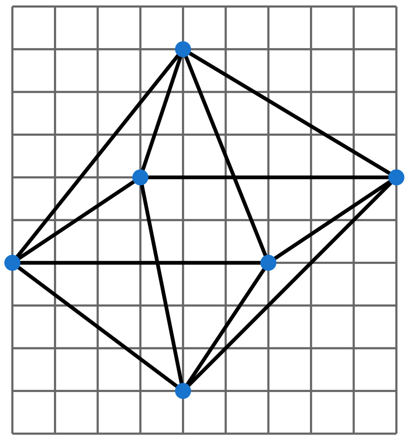
- grid size $=$ area of the drawing using integer grid points



goal: small grid size

More Measures

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

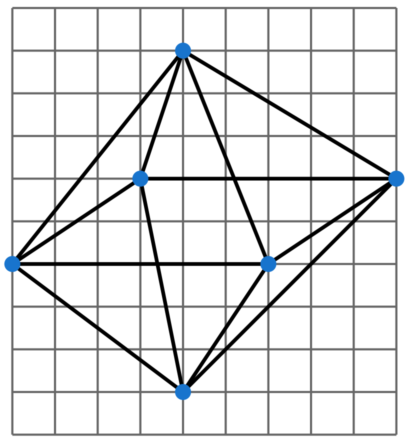


goal: small grid size

$\rightarrow$ implies good vertex and angular resolution

More Measures

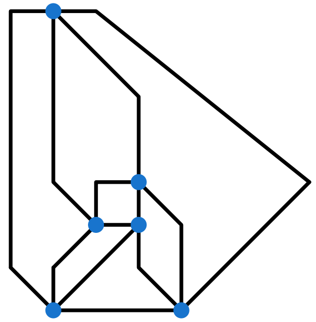
- grid size $=$ area of the drawing using integer grid points



goal: small grid size

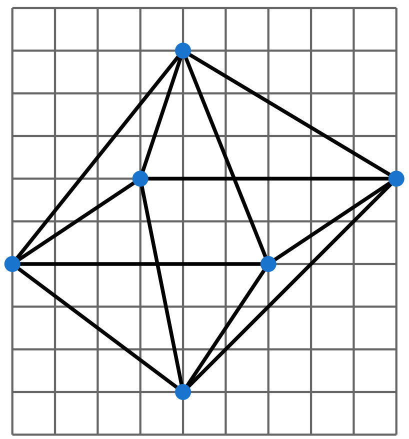
$\rightarrow$ implies good vertex and angular resolution

- number of bends



More Measures

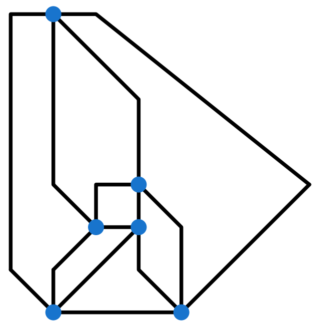
- grid size $=$ area of the drawing using integer grid points



goal: small grid size

$\rightarrow$ implies good vertex and angular resolution

- number of bends

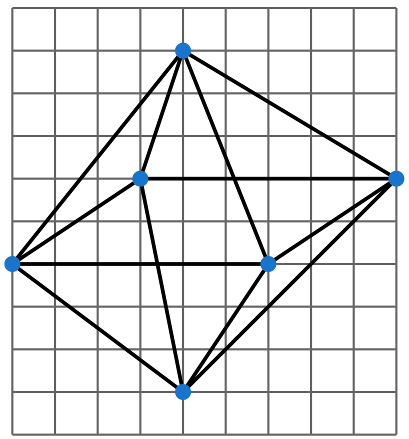


goal: minimize the number of total bends

goal: minimize the maximal number of bends per edge

More Measures

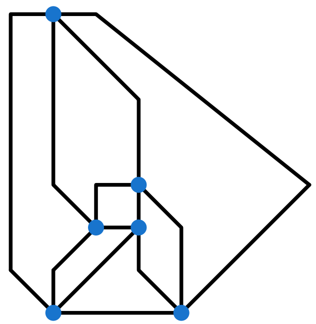
- grid size $=$ area of the drawing using integer grid points



goal: small grid size

$\rightarrow$ implies good vertex and angular resolution

- number of bends



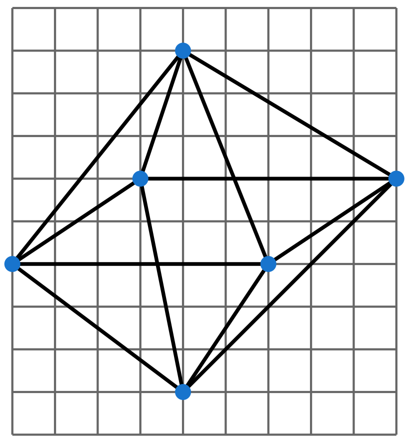
goal: minimize the number of total bends

goal: minimize the maximal number of bends per edge

- number of edge crossings

More Measures

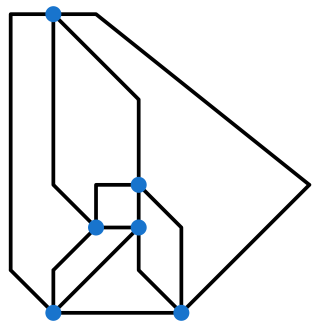
- grid size $=$ area of the drawing using integer grid points



goal: small grid size

$\rightarrow$ implies good vertex and angular resolution

- number of bends



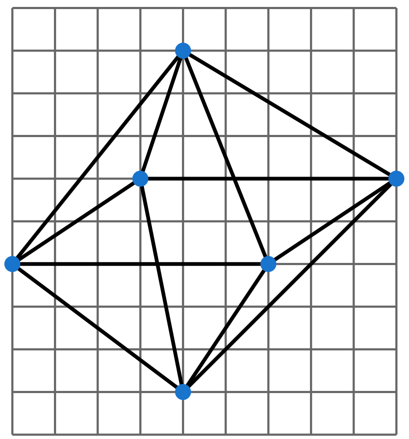
goal: minimize the number of total bends

goal: minimize the maximal number of bends per edge

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

More Measures

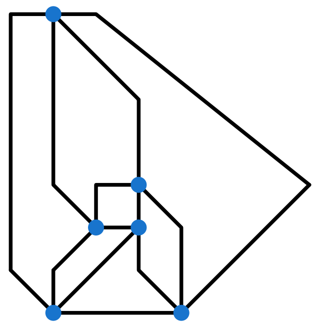
- grid size = area of the drawing using integer grid points



goal: small grid size

$\rightarrow$ implies good vertex and angular resolution

- number of bends



goal: minimize the number of total bends

goal: minimize the maximal 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:

- trees (connected, no cycles)

Graph classes

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

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

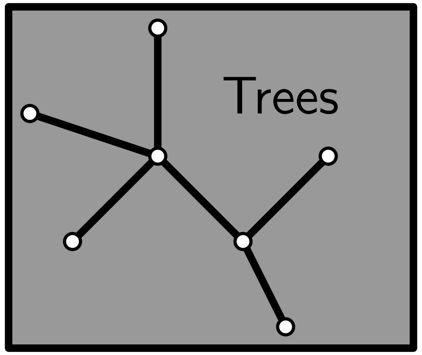
Graph classes

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

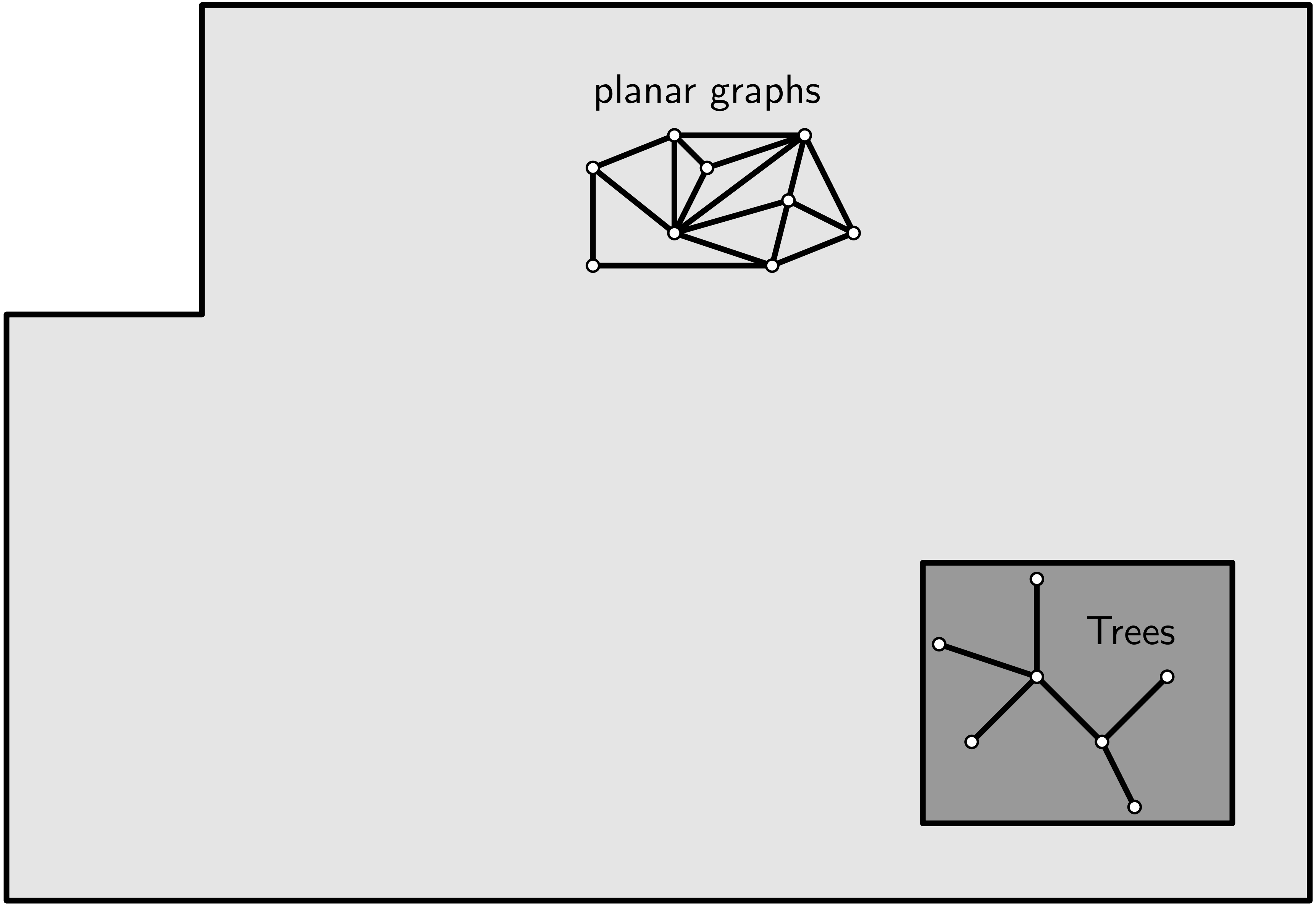
- trees (connected, no cycles)
- planar graphs (can be drawn without crossings)
- triangulations (maximal planar)
- planar 3-trees
- outerplanar graphs
- serial-parallel graphs
- k-connected
- ...

Prominent graph classes by example

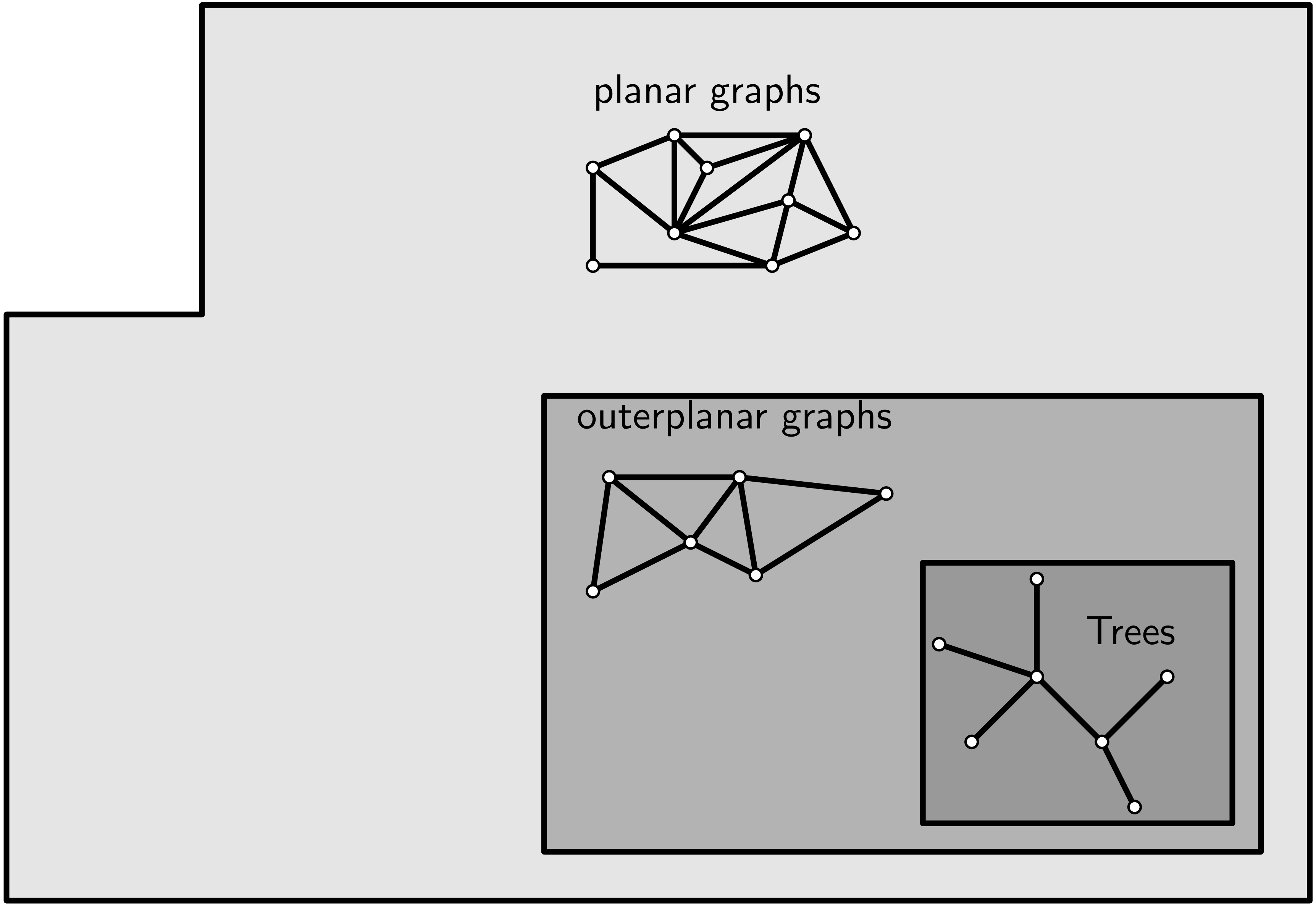
Prominent graph classes by example



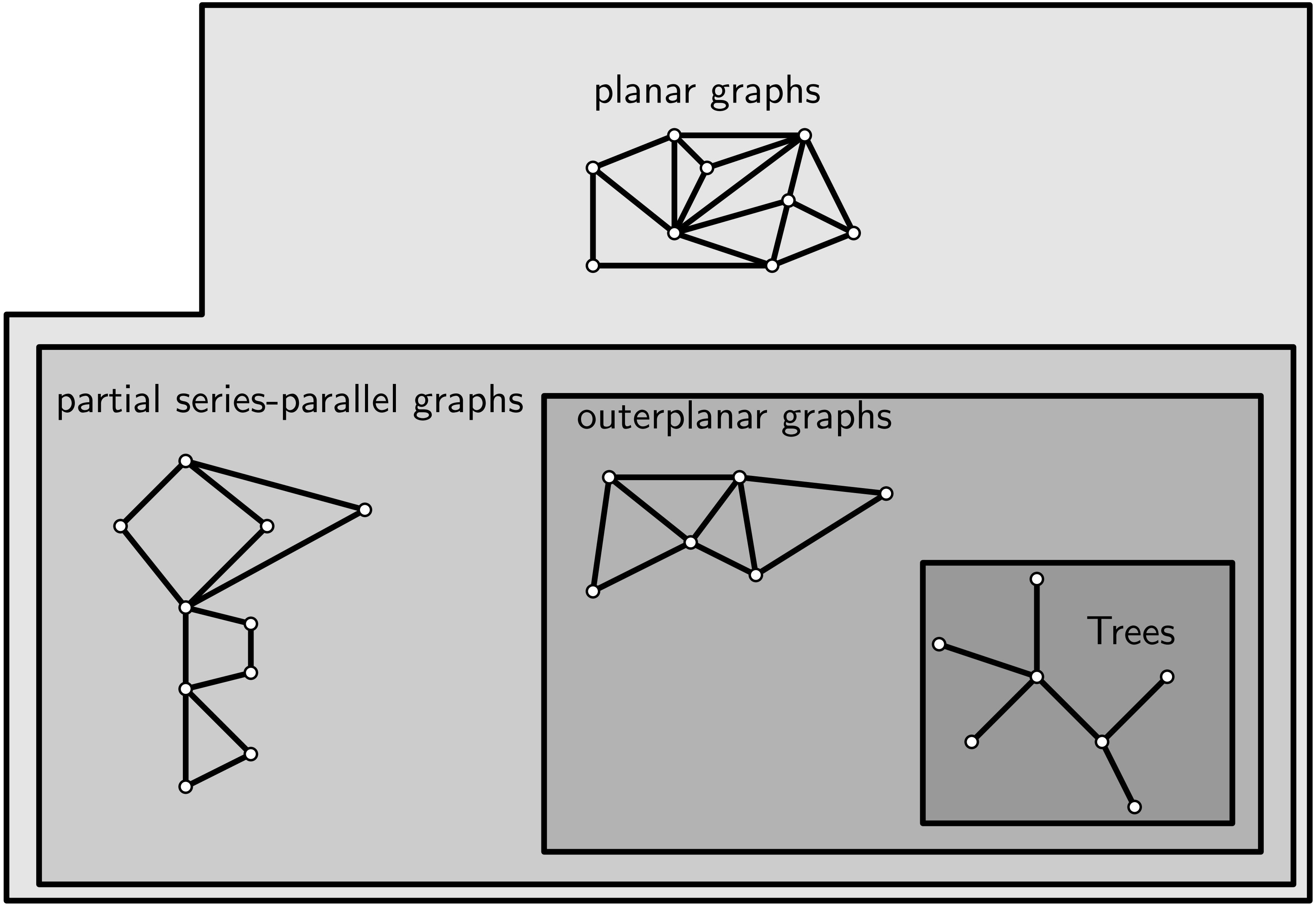
Prominent graph classes by example



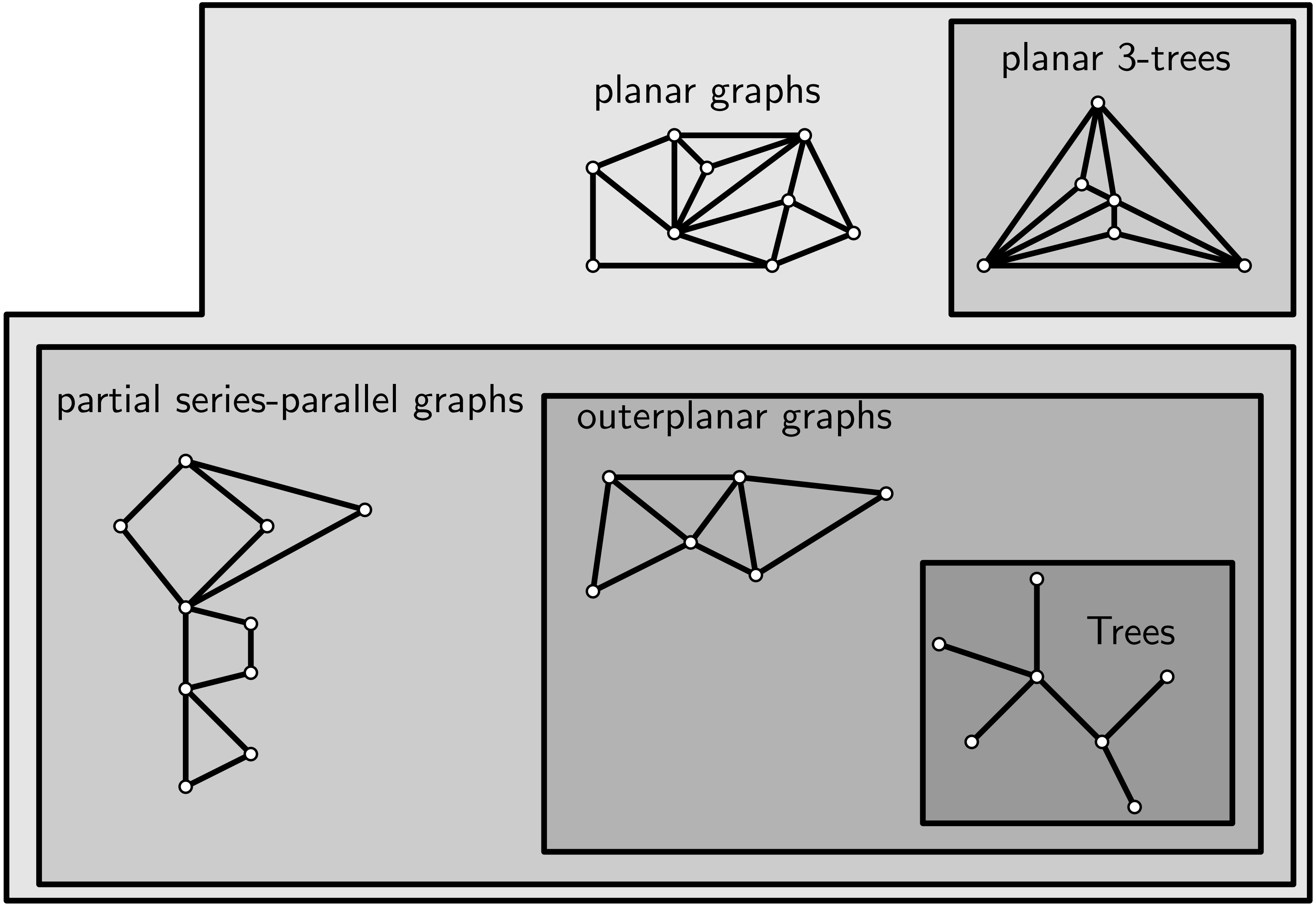
Prominent graph classes by example



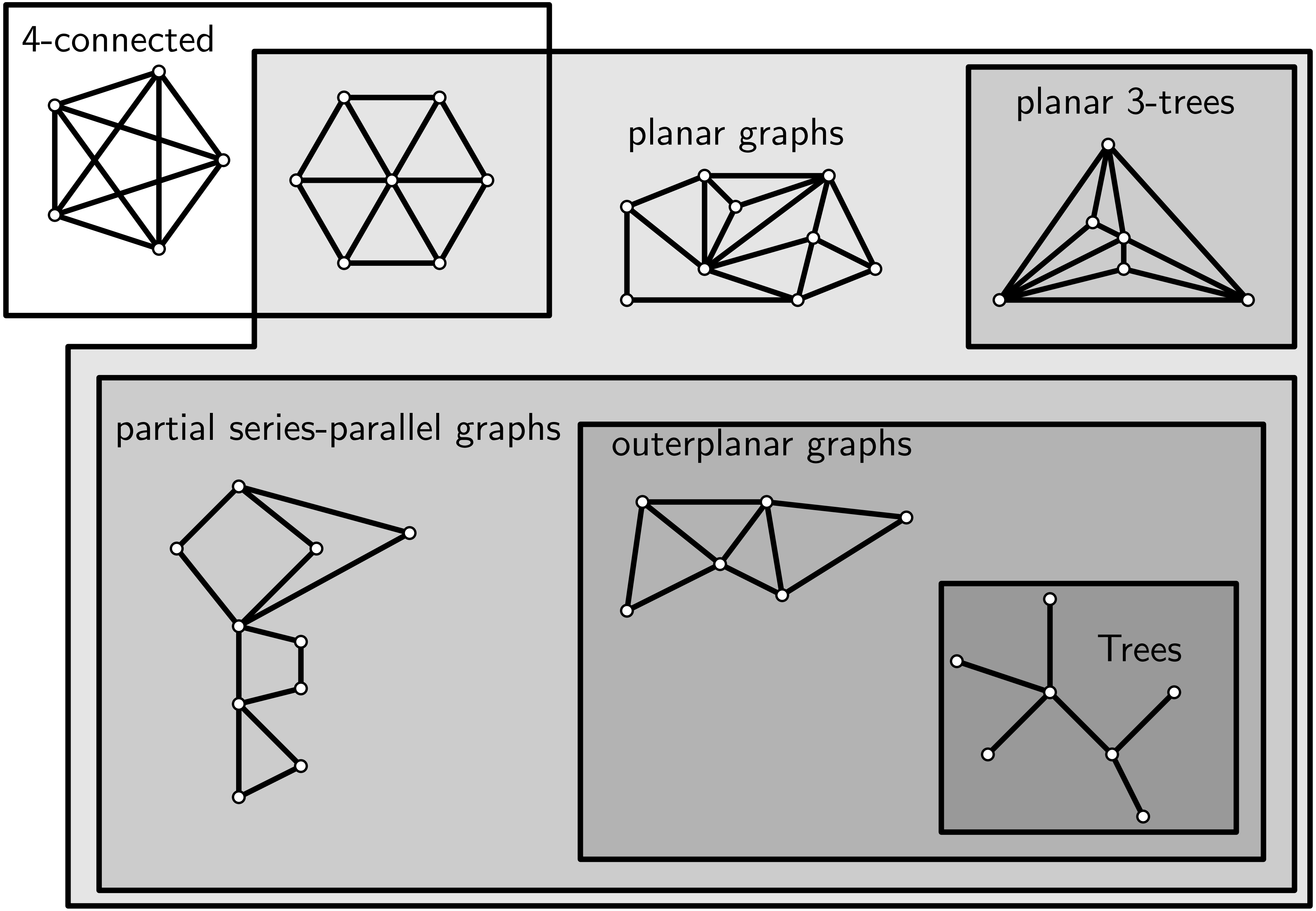
Prominent graph classes by example



Prominent graph classes by example



Prominent graph classes by example



Standard techniques I

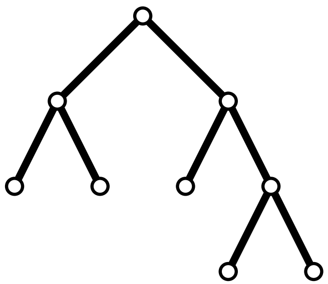
Standard techniques I

- 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.

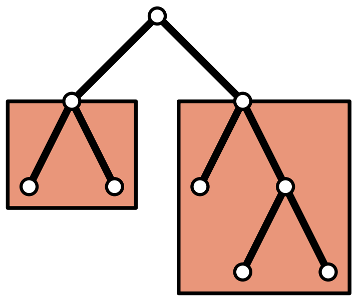
Binary trees:



Standard techniques I

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

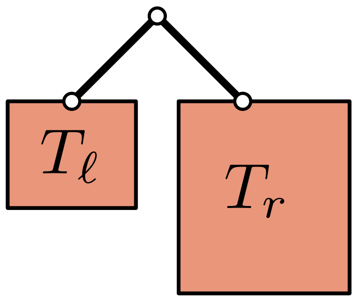
Binary trees:



Standard techniques I

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

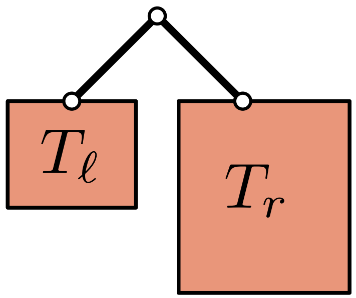
Binary trees:

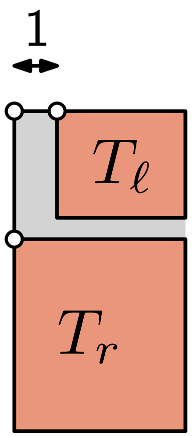


Standard techniques I

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

Binary trees:



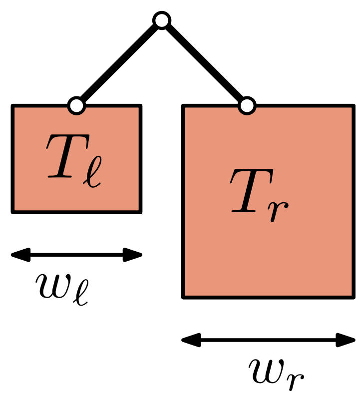


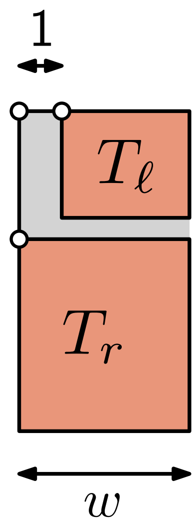
w.l.o.g. $|T_\ell| \le |T_r|$

Standard techniques I

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

Binary trees:





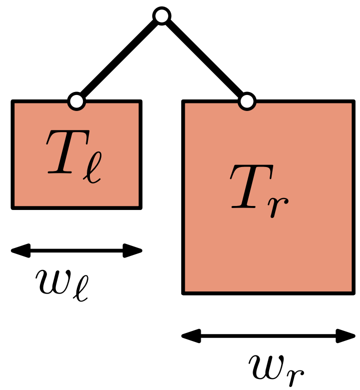
w.l.o.g. $|T_\ell| \leq |T_r|$

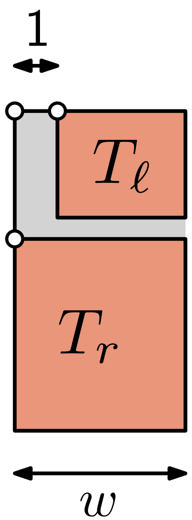
$$w = \max\{w_l + 1, w_r\}$$

Standard techniques I

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

Binary trees:





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

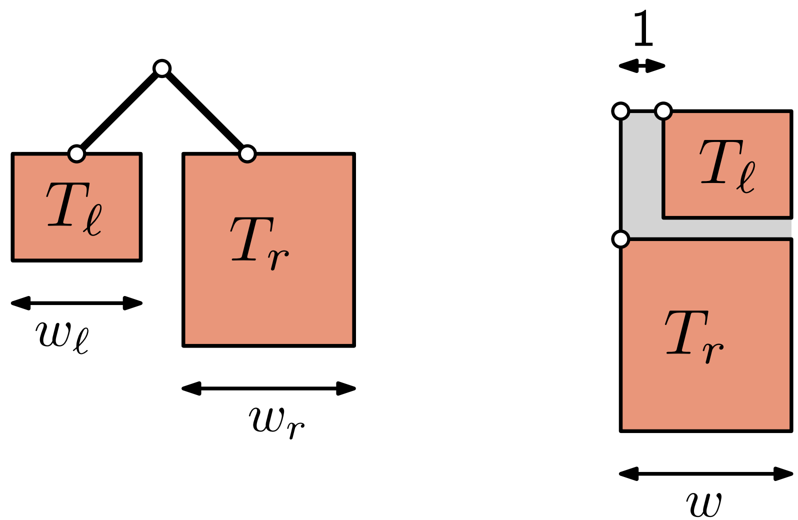
$w = \max\{w_l + 1, w_r\}$

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

Standard techniques I

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

Binary trees:



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

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

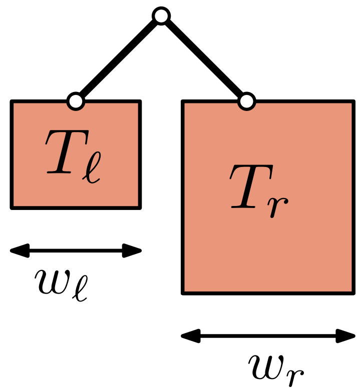
$W(n) \leq W(n/2) + 1$
$W(n) = O(\log n)$

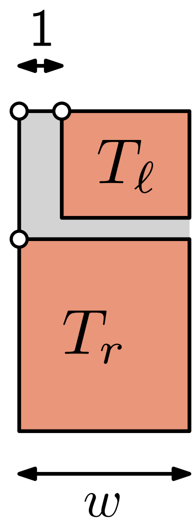
No row without vertex: $H(n) = O(n)$

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: $H(n) = O(n)$

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

[Crescenzi, Di Battista, Piperno '92]

Standard techniques II

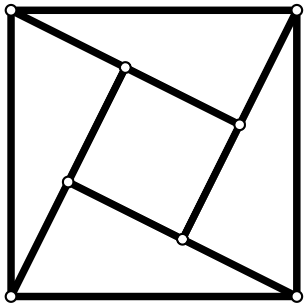
Standard techniques II

- 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.

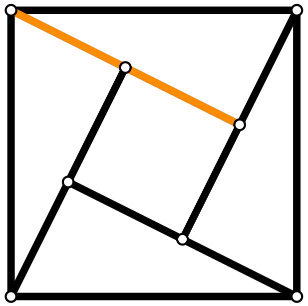
  Goal: Draw a graph with few segments.



Standard techniques II

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

  Goal: Draw a graph with few segments.

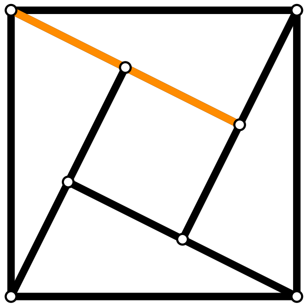


8 vertices
12 edges
8 segments

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

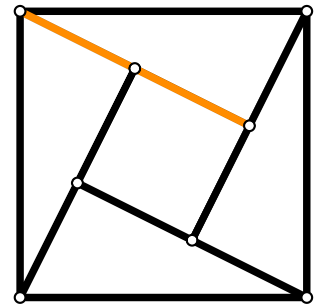
Planar 3-trees can be drawn with $2n-4$ segments.

[Dujmović et al. '05]

Standard techniques II

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

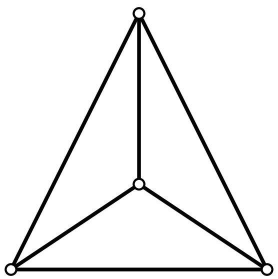
Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

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

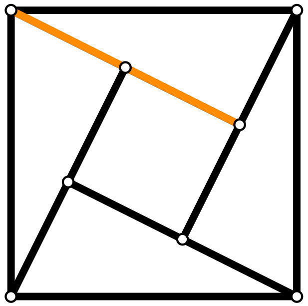
[Dujmović et al. '05]



Standard techniques II

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

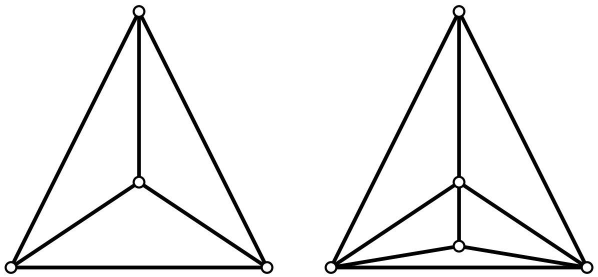
Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

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

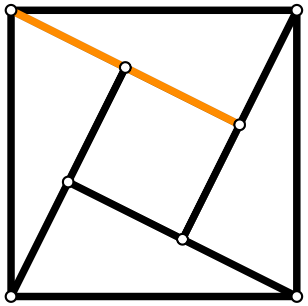
[Dujmović et al. '05]



Standard techniques II

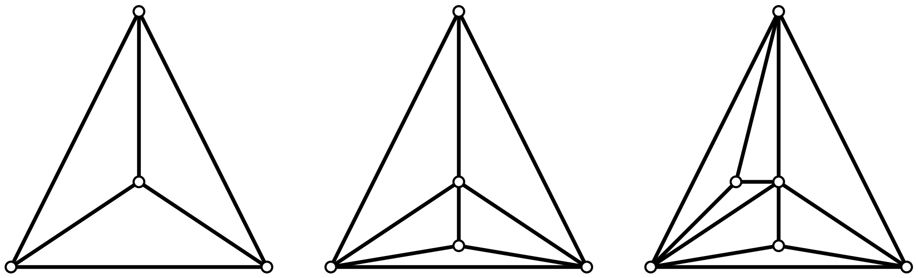
- If your graph class has an inductive construction, build the graph drawing inductively.

  Goal: Draw a graph with few segments.



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

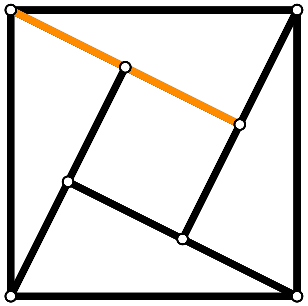
[Dujmović et al. '05]



Standard techniques II

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

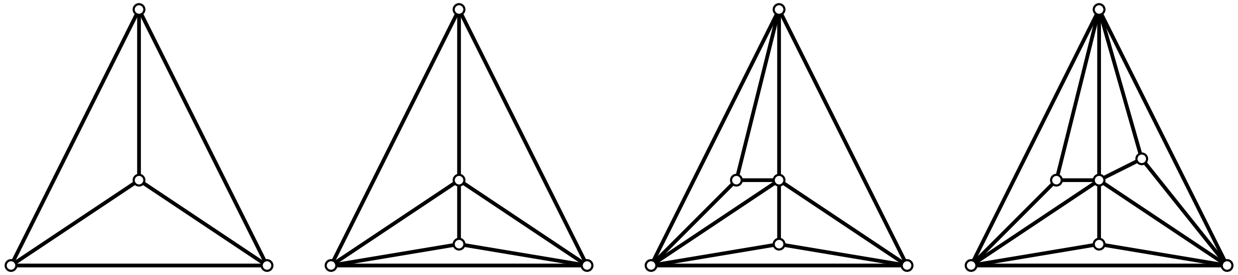
Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

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

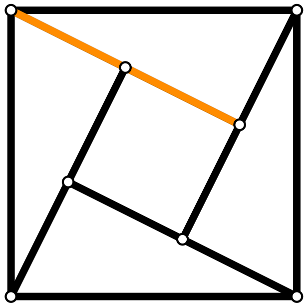
[Dujmović et al. '05]



Standard techniques II

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

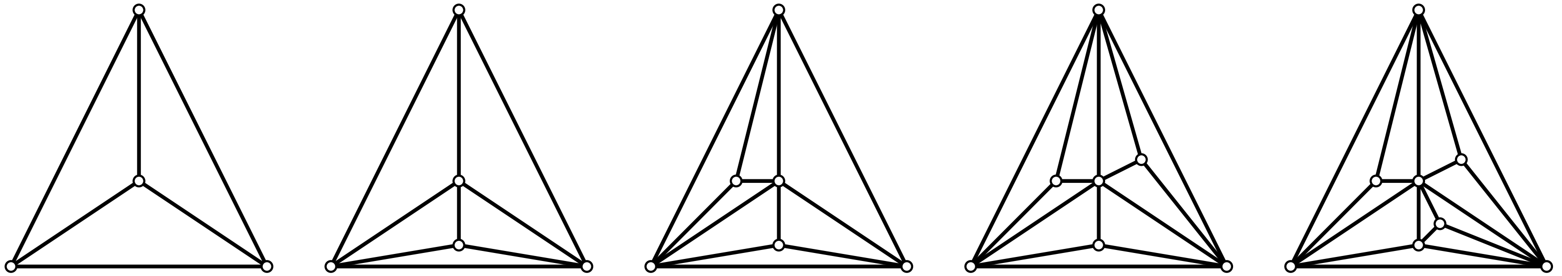
Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

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

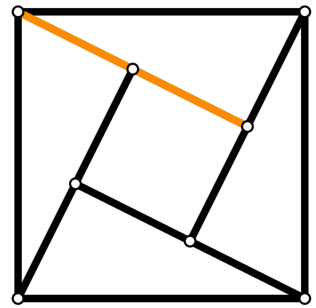
[Dujmović et al. '05]



Standard techniques II

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

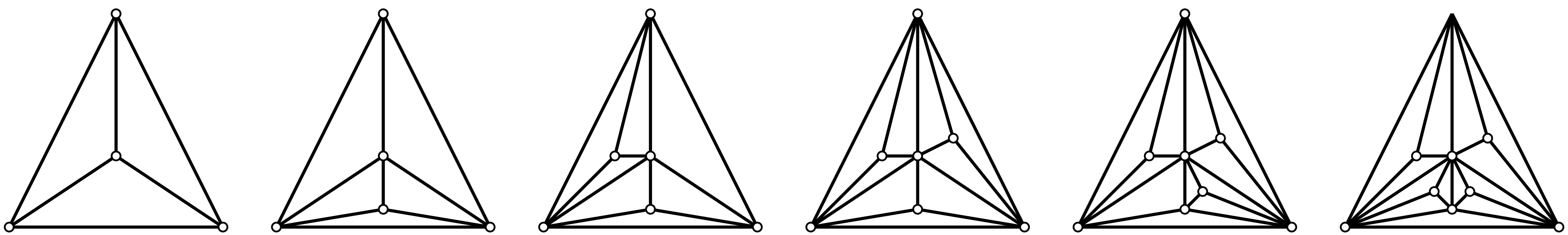
Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

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

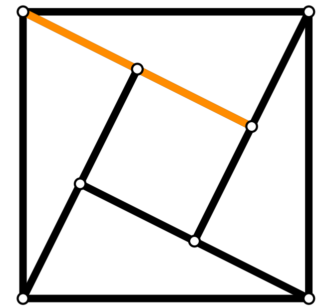
[Dujmović et al. '05]



Standard techniques II

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

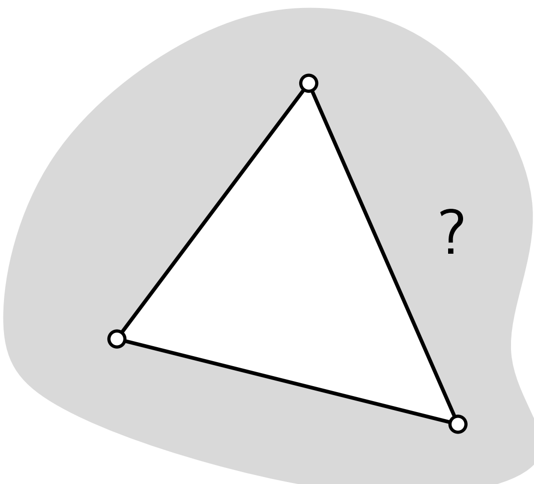
Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

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

[Dujmović et al. '05]

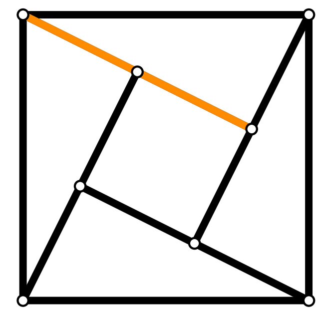


3-tree before last addition
$\leq 2(n-1)-4$ segments

Standard techniques II

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

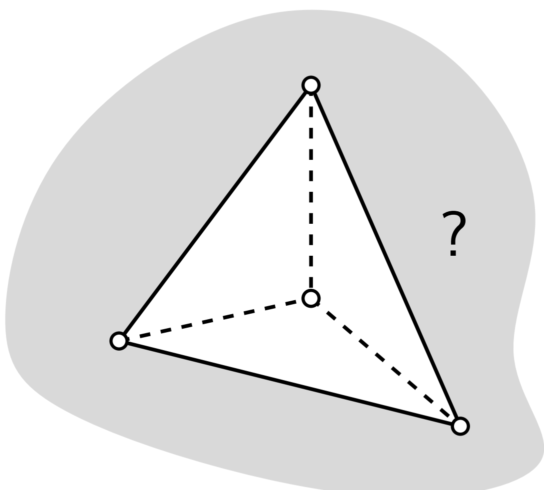
Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

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

[Dujmović et al. '05]

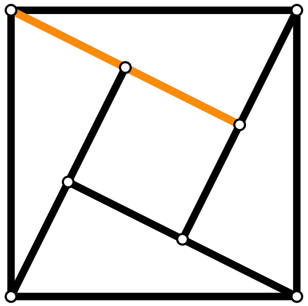


3-tree before last addition
$\leq 2(n-1) - 4$ segments

Standard techniques II

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

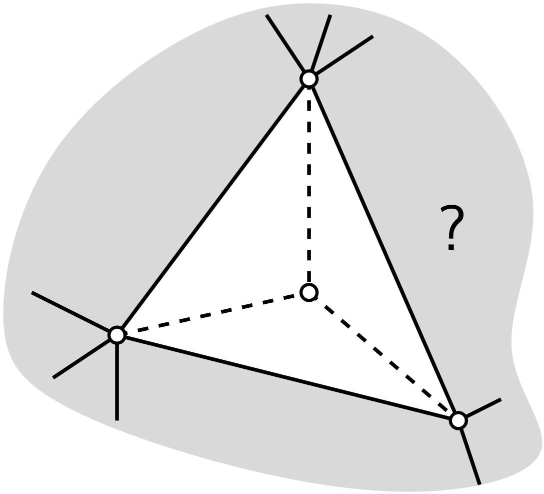
Goal: Draw a graph with few segments.



8 vertices
12 edges
8 segments

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

[Dujmović et al. '05]

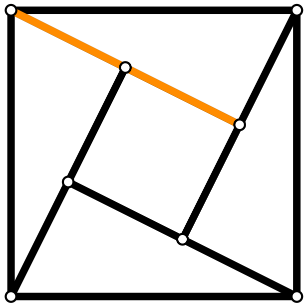


3-tree before last addition
$\leq 2(n-1) - 4$ segments

Standard techniques II

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

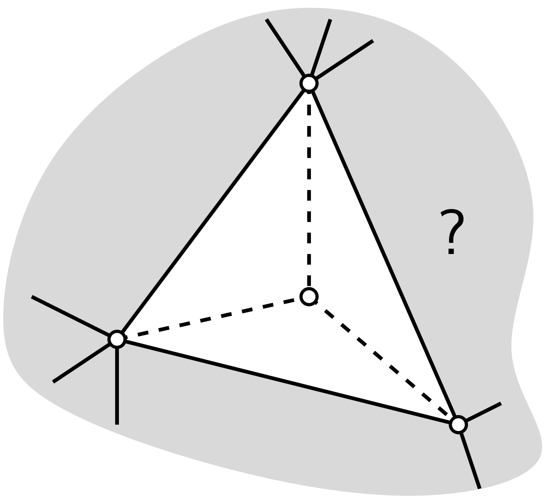
Goal: Draw a graph with few segments.



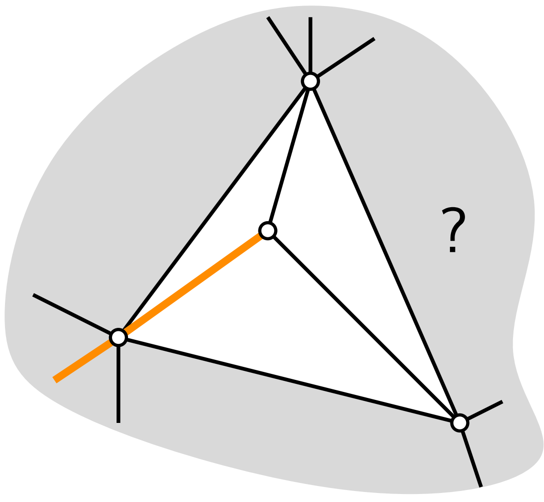
8 vertices
12 edges
8 segments

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

[Dujmović et al. '05]



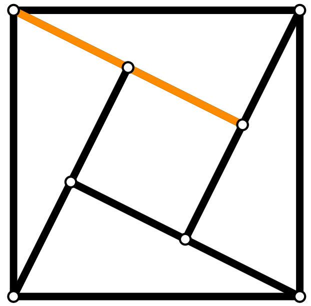
3-tree before last addition
$\leq 2(n-1)-4$ segments



Standard techniques II

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

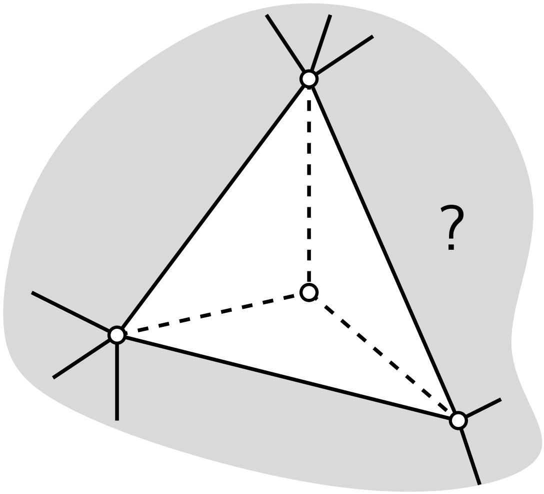
Goal: Draw a graph with few segments.



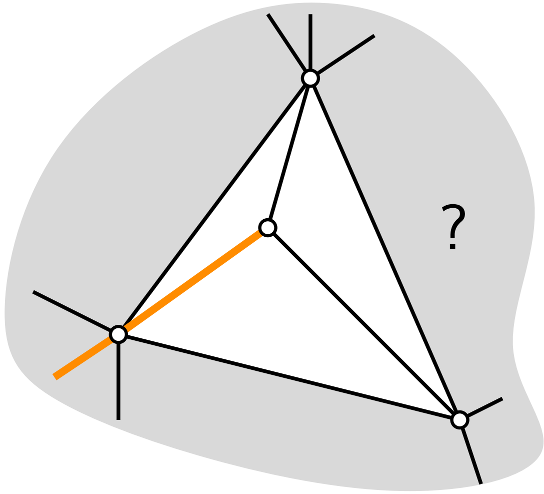
8 vertices
12 edges
8 segments

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

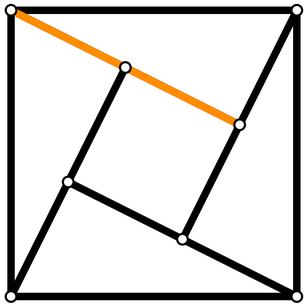


$\leq 2(n-1)-4+2 = 2n-4$ segments

Standard techniques II

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

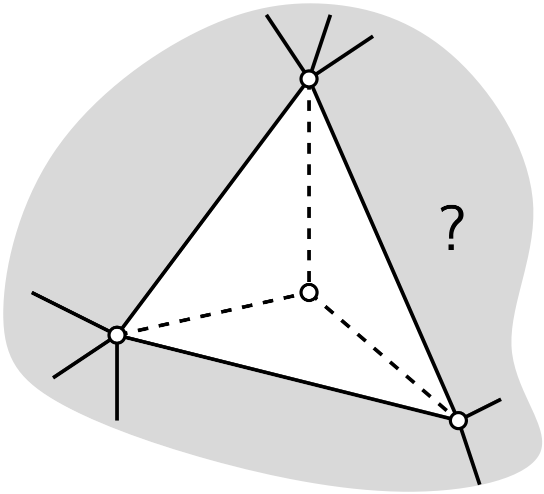
Goal: Draw a graph with few segments.



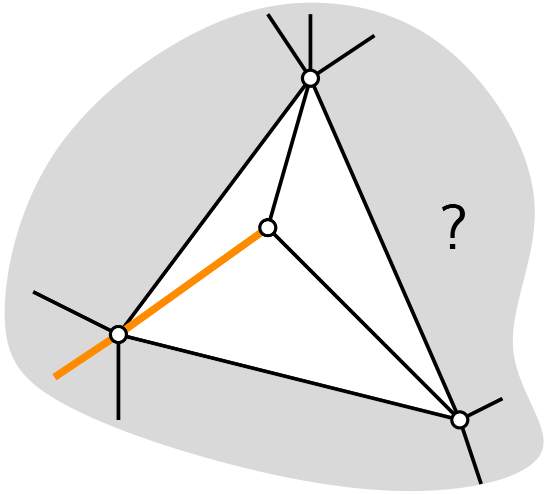
8 vertices
12 edges
8 segments

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



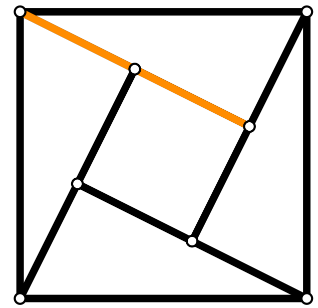
$\leq 2(n-1)-4+2=2n-4$ segments

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.

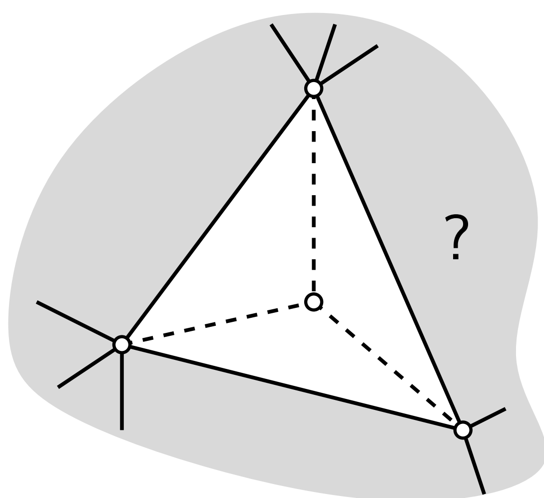
Goal: Draw a graph with few segments.



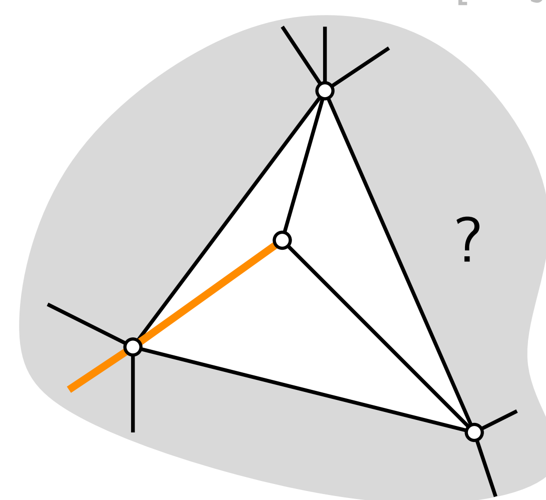
8 vertices
12 edges
8 segments

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



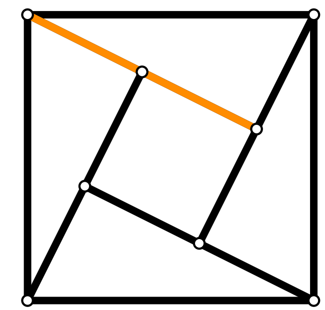
$\leq 2(n-1) - 4 + 2 = 2n - 4$ segments

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.

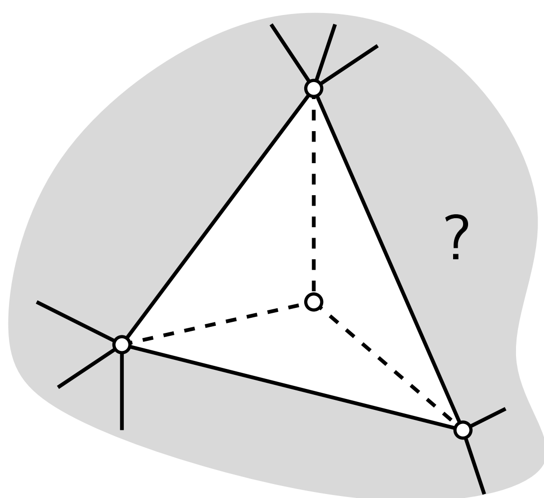
Goal: Draw a graph with few segments.



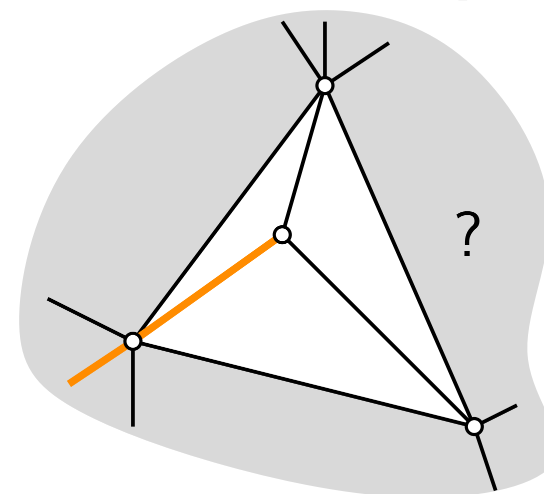
8 vertices
12 edges
8 segments

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 works best in practice is the spring embedder.

More ideas

- The approach that 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 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 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 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)

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

2. adjacent vertices attract

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

(like a spring)

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

Tutte '60

Eades '84

More ideas

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

1. all vertices repel
2. adjacent vertices attract

Tutte '60

$F_{ij} = 0$

(but pin a face)

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

(like a spring)

Eades '84

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

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

Fruchterman Reingold '92

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

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

Simultaneous embedding

Simultaneous embedding

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

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

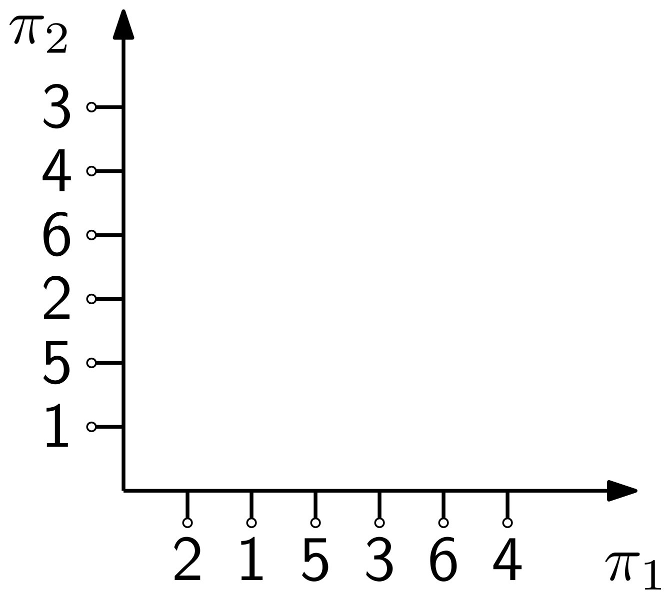
Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$.

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

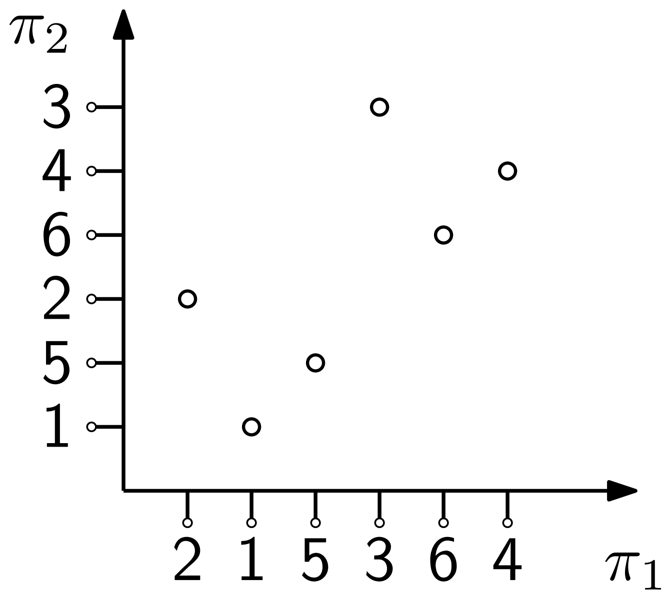
Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$.

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

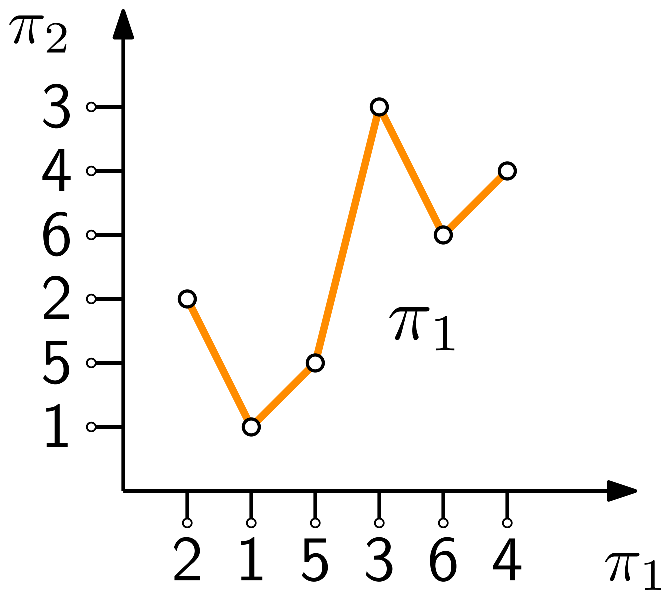
Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$.

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

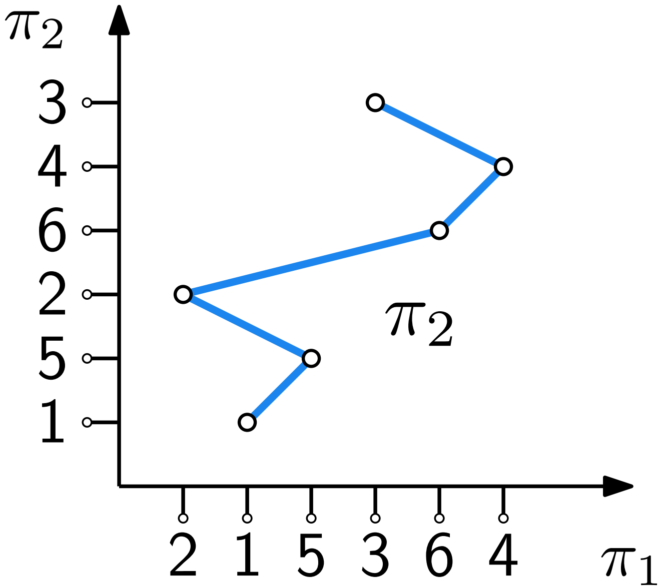
Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$.

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

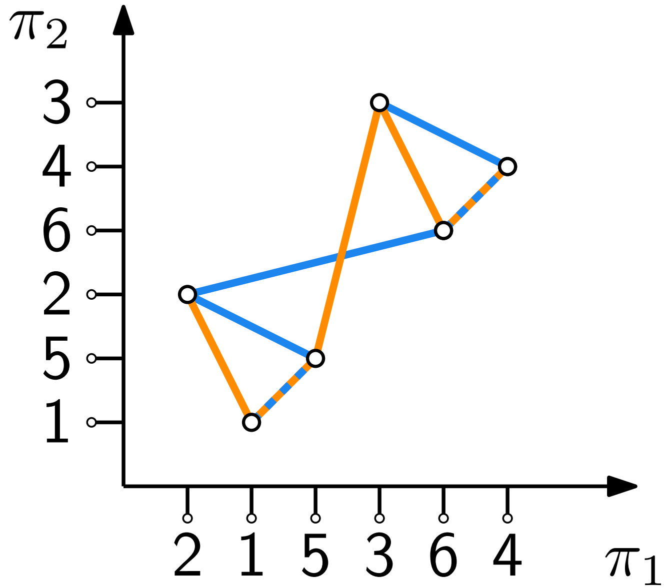
Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$.

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

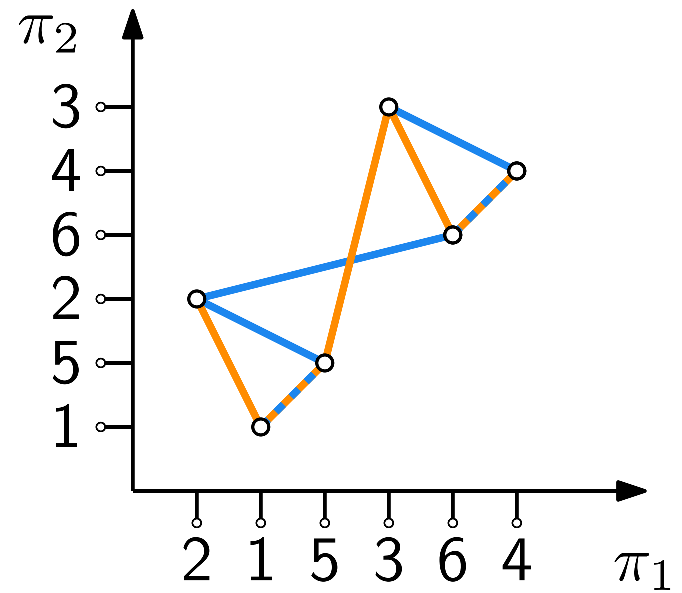
Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$.

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

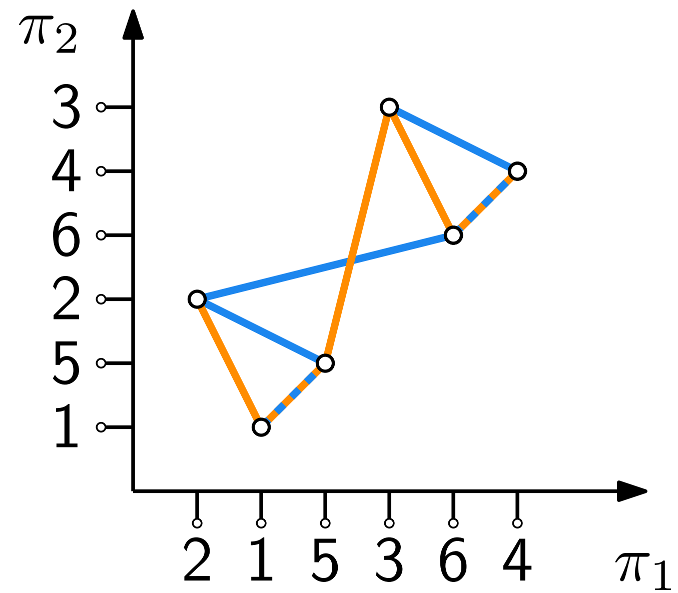
Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$.

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

Two paths: YES

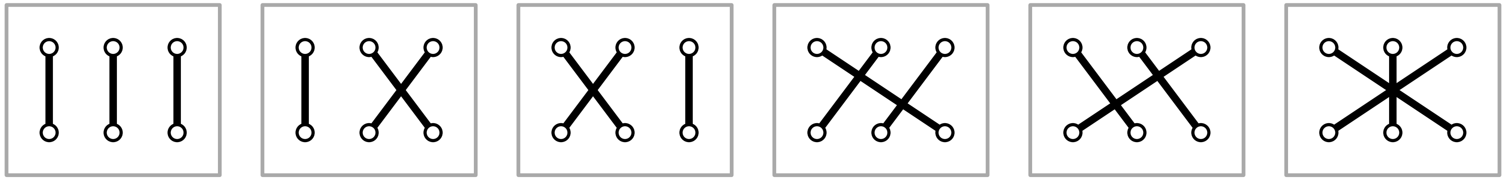
[Brass et al. '07]

$\pi_1 = (v_2, v_1, v_5, v_3, v_6, v_4)$
$\pi_2 = (v_1, v_5, v_2, v_6, v_4, v_3)$



Six Matchings: NO

[Cabello et al. '07]



Simultaneous embedding

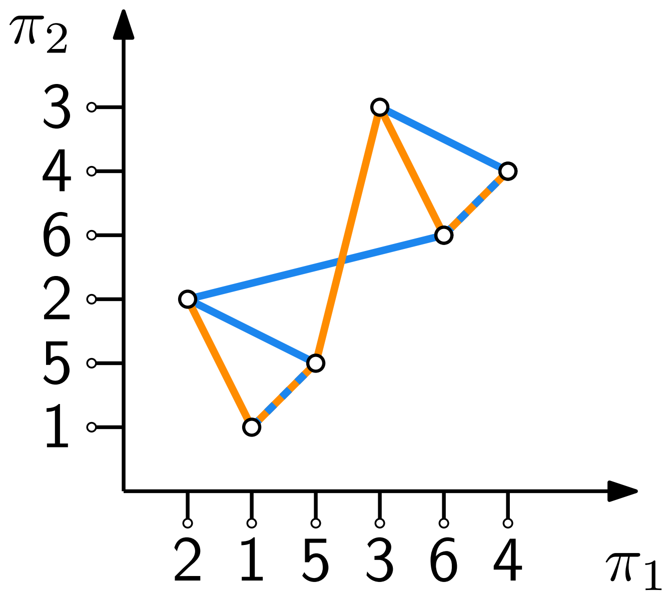
Given: Graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots$.

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

Two paths: YES

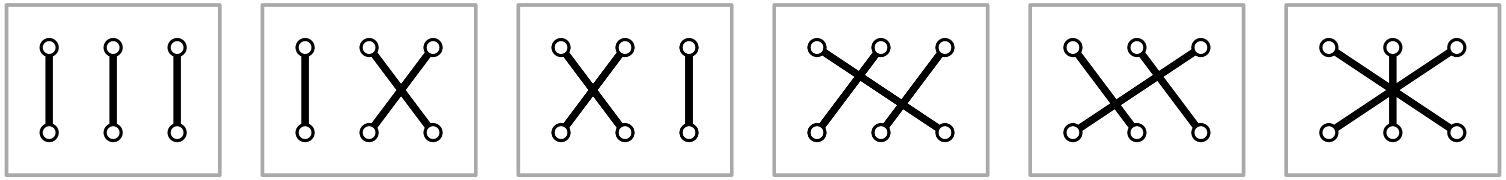
[Brass et al. '07]

$$\pi_1 = (v_2, v_1, v_5, v_3, v_6, v_4)$$
$$\pi_2 = (v_1, v_5, v_2, v_6, v_4, v_3)$$



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.

Morphing

Morphing

Not ONE (geometric) graph with vertices in different positions?

Given: Drawings of planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$.

Question: Can we continuously deform G_1 to G_2 without introducing crossings?

Morphing

Not ONE (geometric) graph with vertices in different positions?

Given: Drawings of planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$.

Question: Can we continuously deform G_1 to G_2 without introducing crossings?

Solution: [Floater & Gotsman '99]:

- Compute (asymmetric) spring weights for the drawings of G_1/G_2.
- Interpolate between weights and compute spring embedding.

Morphing

Not ONE (geometric) graph with vertices in different positions?

Given: Drawings of planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$.

Question: Can we continuously deform G_1 to G_2 without introducing crossings?

Solution: [Floater & Gotsman '99]:

- Compute (asymmetric) spring weights for the drawings of G_1/G_2.
- Interpolate between weights and compute spring embedding.

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

Morphing

Not ONE (geometric) graph with vertices in different positions?

Given: Drawings of planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$.

Question: Can we continuously deform G_1 to G_2 without introducing crossings?

Solution: [Floater & Gotsman '99]:

- Compute (asymmetric) spring weights for the drawings of G_1/G_2.
- 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.

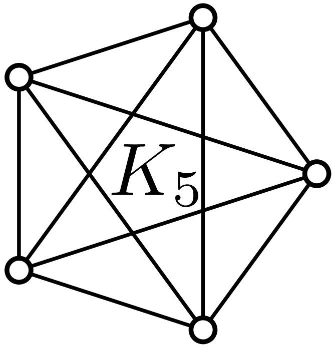
3D

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 $\rho_3^2(G)$.

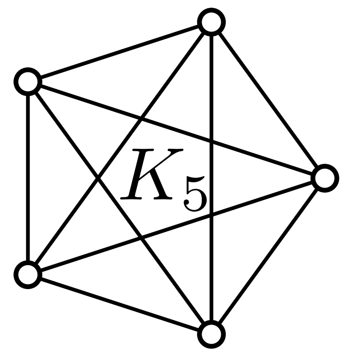
$\rho_3^2(K_5) = ?$

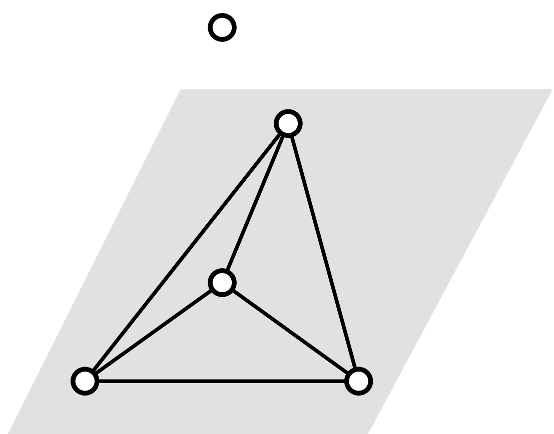


3D

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 $\rho_3^2(G)$.

$$\rho_3^2(K_5) = ?$$

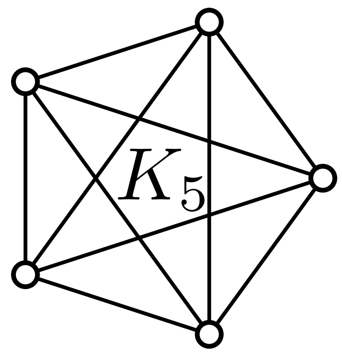


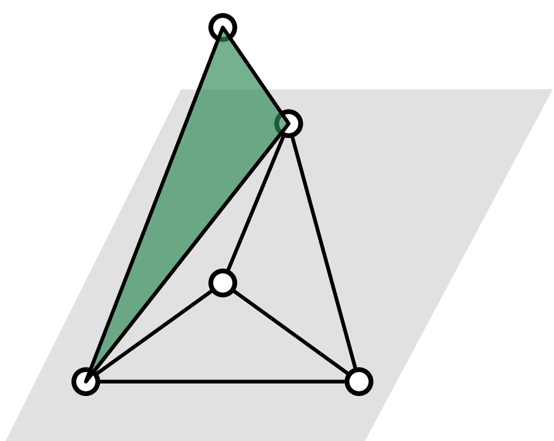


3D

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 $\rho_3^2(G)$.

$\rho_3^2(K_5) = ?$

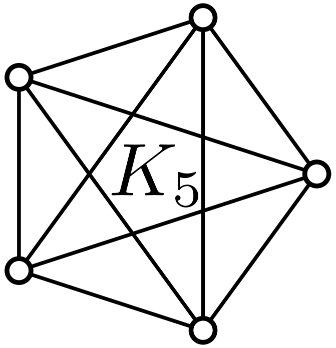


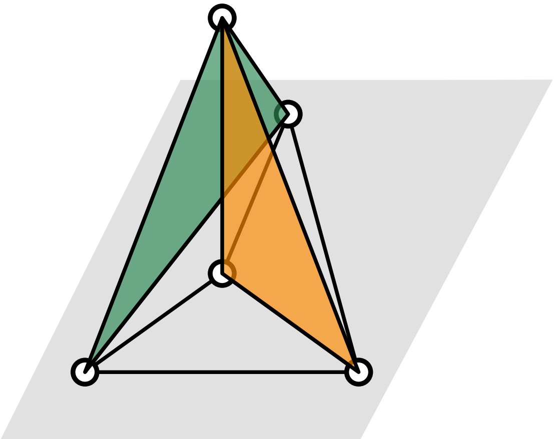


3D

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 $\rho_3^2(G)$.

$\rho_3^2(K_5) = ?$

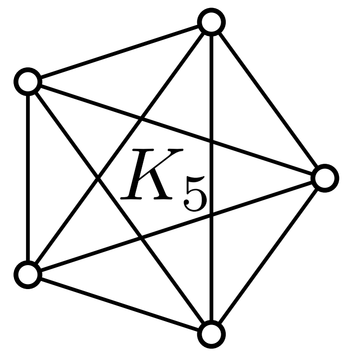


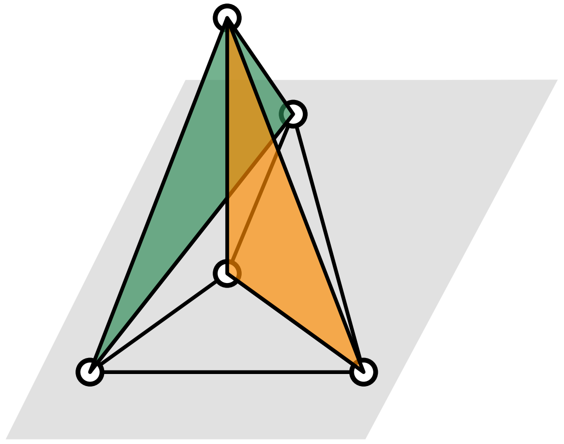


3D

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

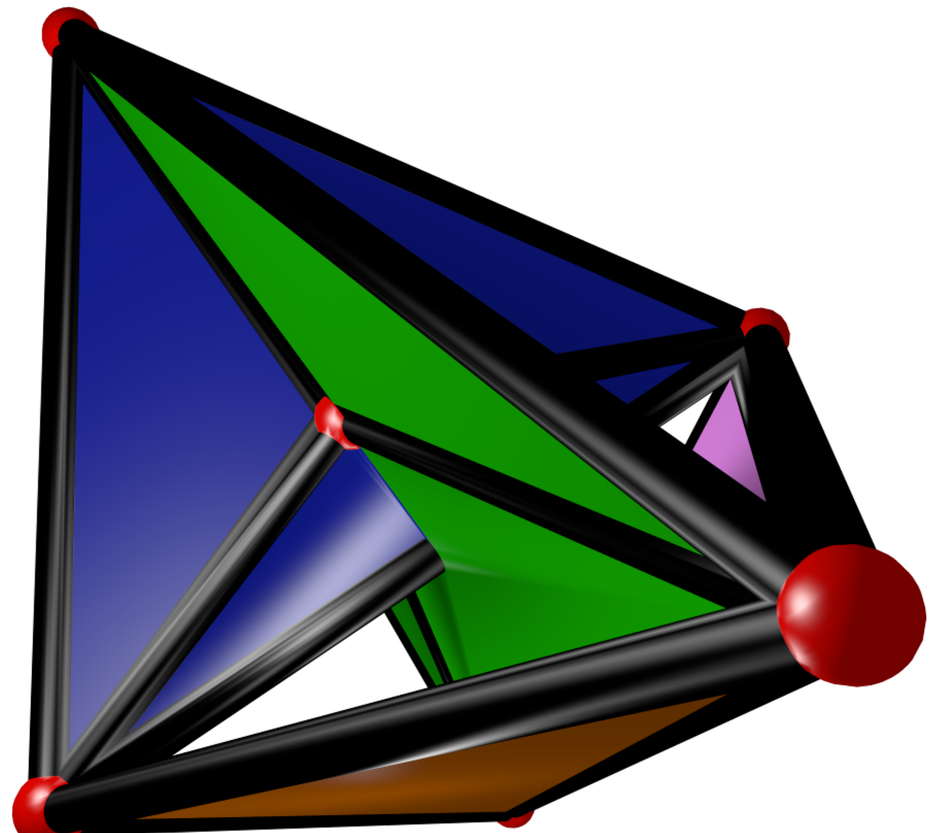
Given a graph G, find a set of planes in 3-space such that there is a *crossing-free straight-line drawing* of G with all vertices and edges drawn on these planes.
Call the minimum number of planes needed $\rho_3^2(G)$.

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

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 $\rho_3^2(G)$.

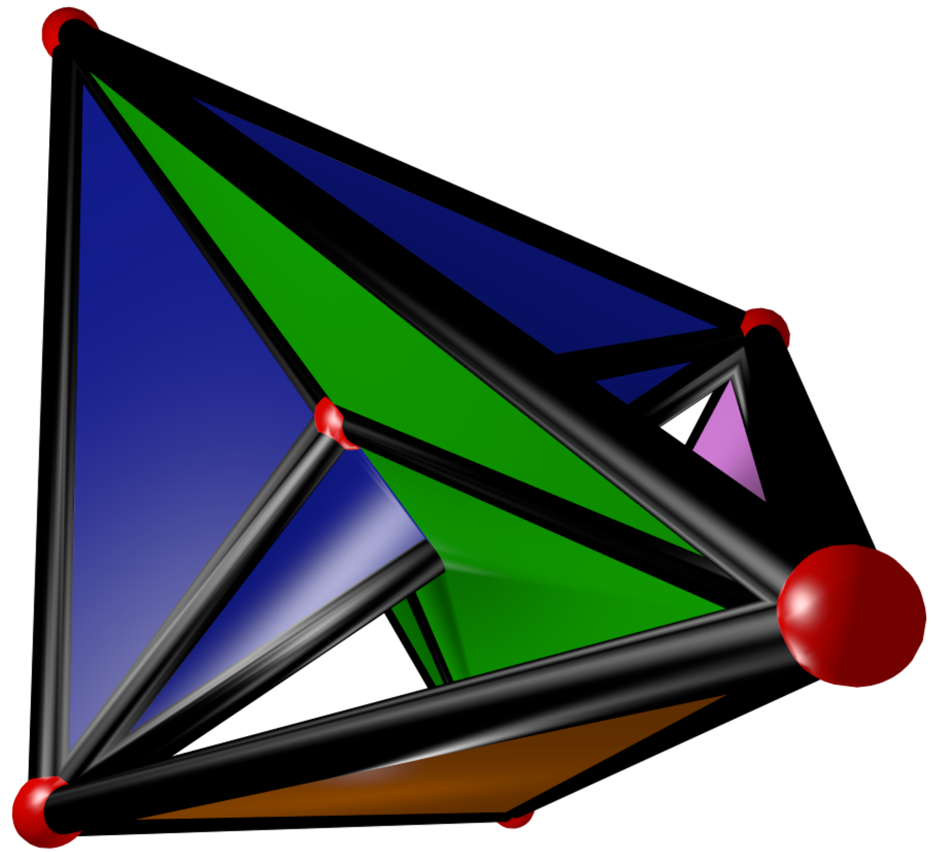
$\rho_3^2(K_6) = ?$



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 $\rho_3^2(G)$.

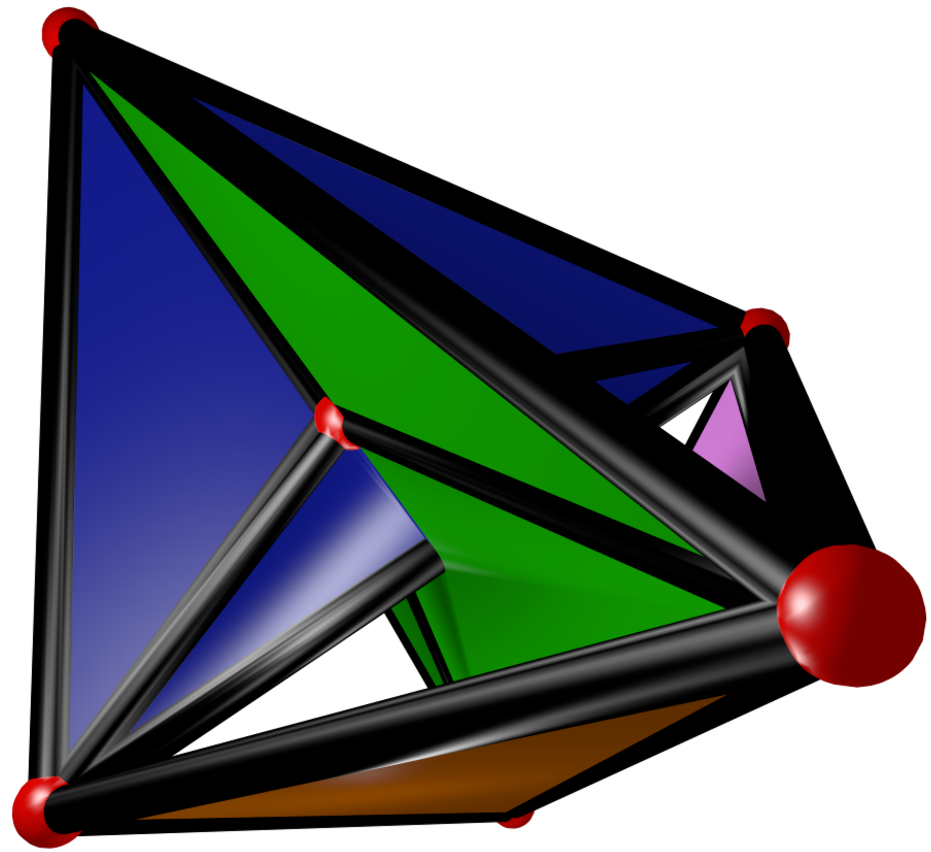
$$\rho_3^2(K_6) = 4$$



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 $\rho_3^2(G)$.

$\rho_3^2(K_6) = 4$

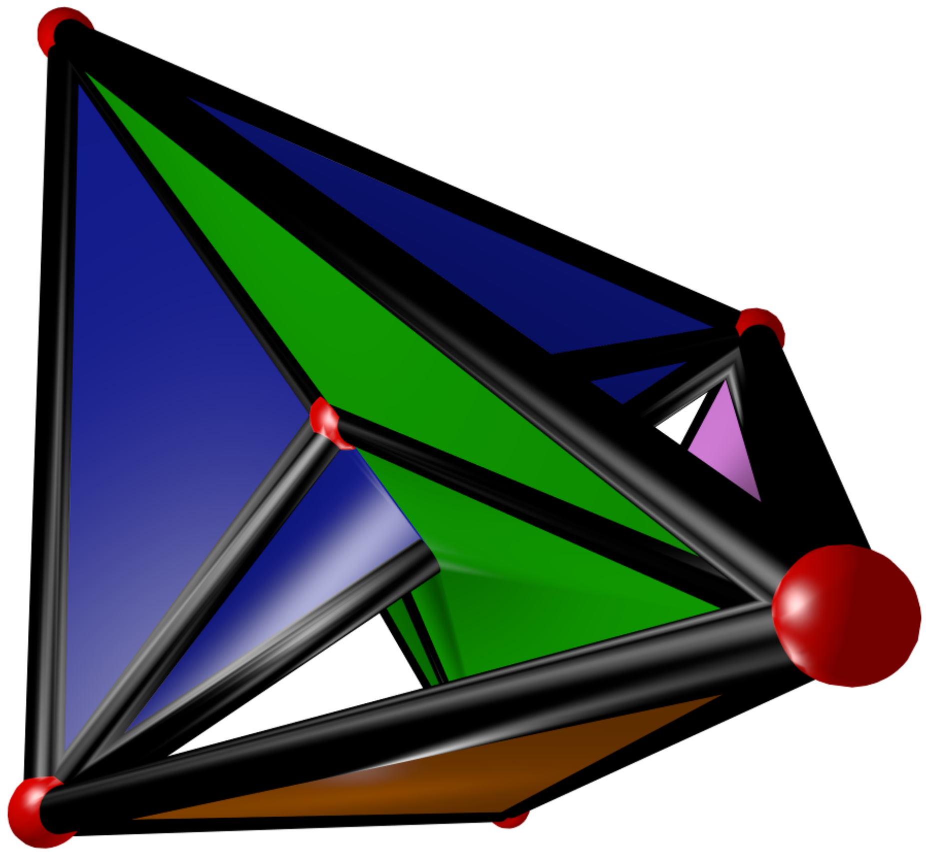


For any planar graph G, clearly $\rho_3^2(G) = 1$.

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 $m \geq 3$

$\rho_d^m = \rho_3^m$ 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 $d \geq 3$

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

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

Interesting cases

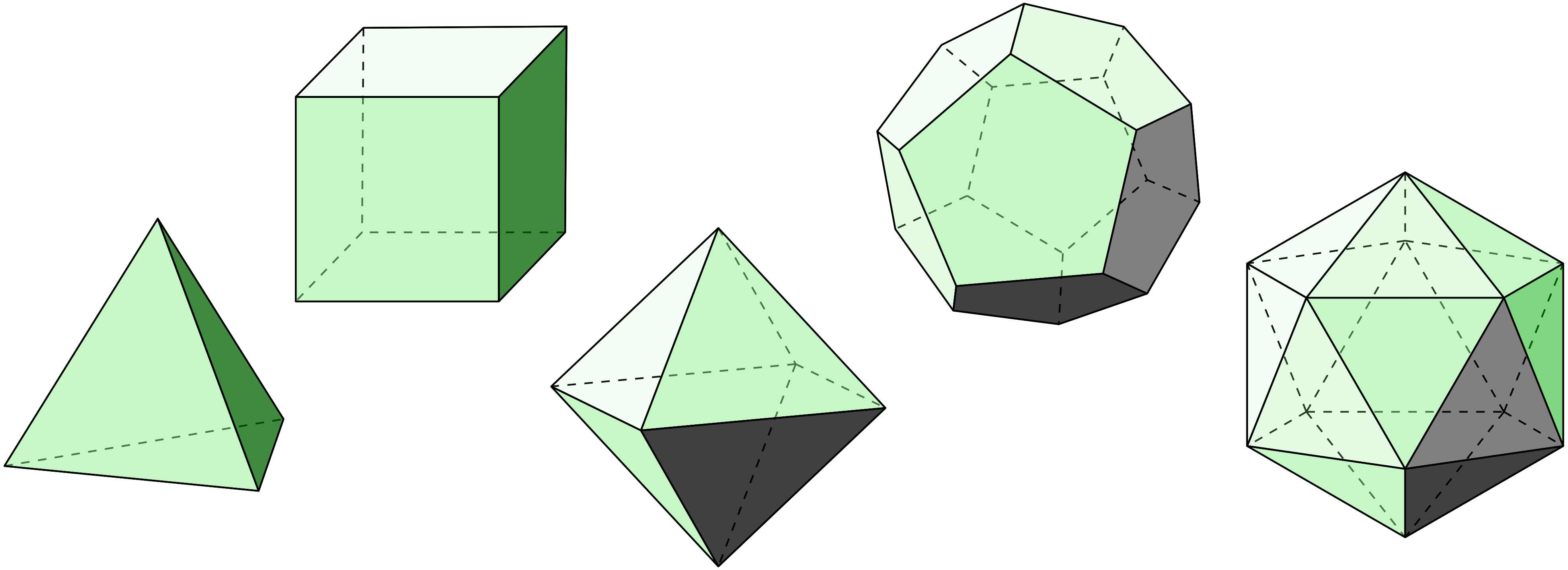
- Line cover numbers in 2D and 3D: ρ_2^1, ρ_3^1, π_2^1, π_3^1
- Plane cover numbers in 3D: ρ_3^2, π_3^2

[WADS'17 & GD'19]

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

Line Cover Numbers of the Platonic Solids

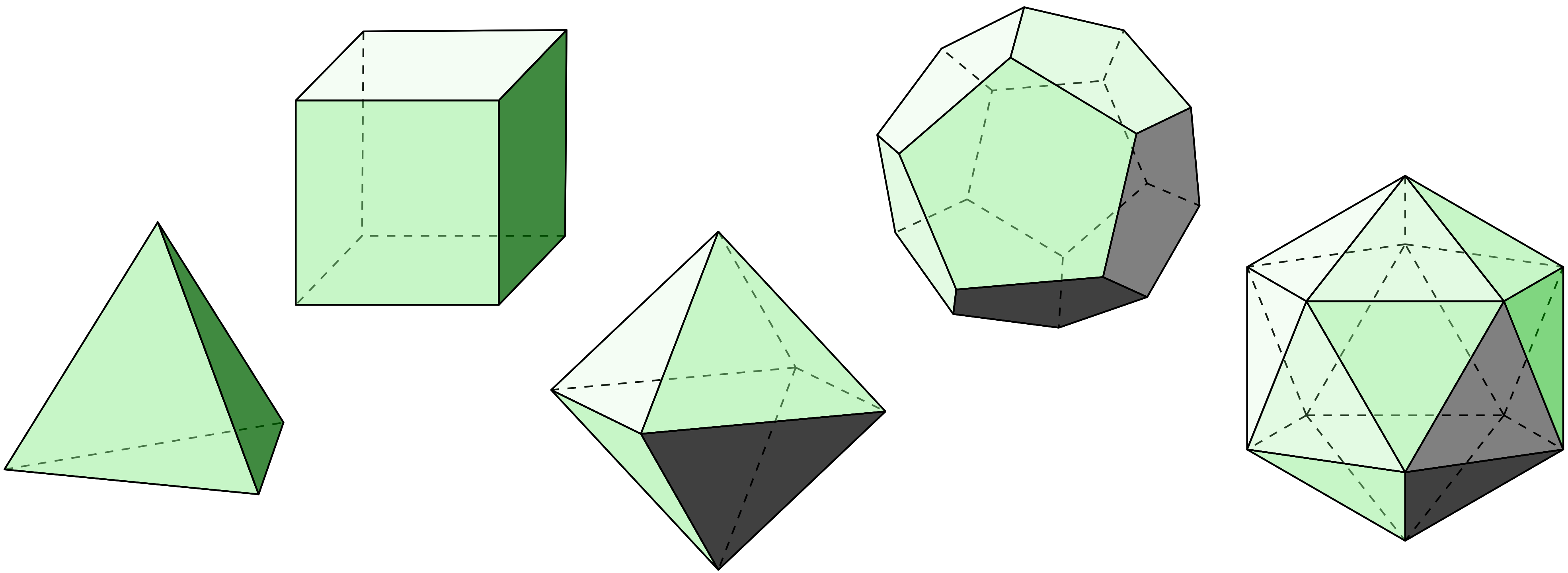
$G = (V, E)$	$\lvert V \rvert$	$\lvert E \rvert$	$\lvert F \rvert$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4				
cube	8	12	6				
octahedron	6	12	8				
dodecahedron	20	30	12				
icosahedron	12	30	20				



Line Cover Numbers of the Platonic Solids

[Kryven et al., CALDAM'18]

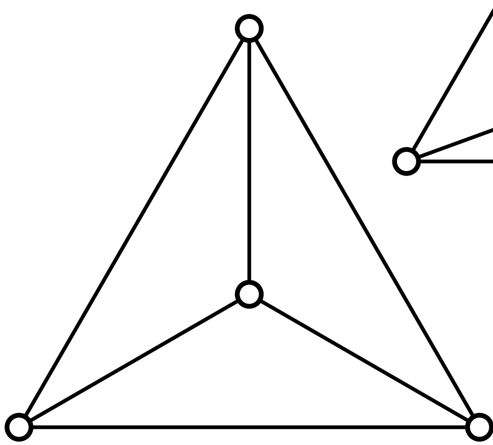
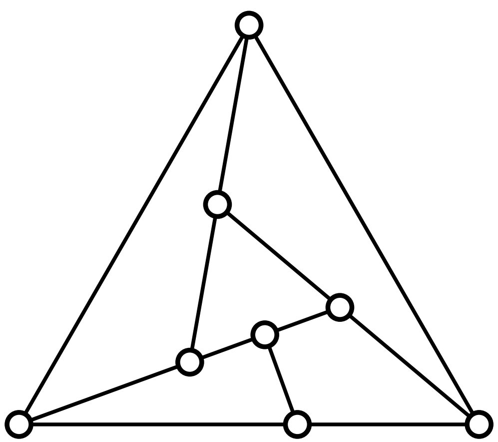
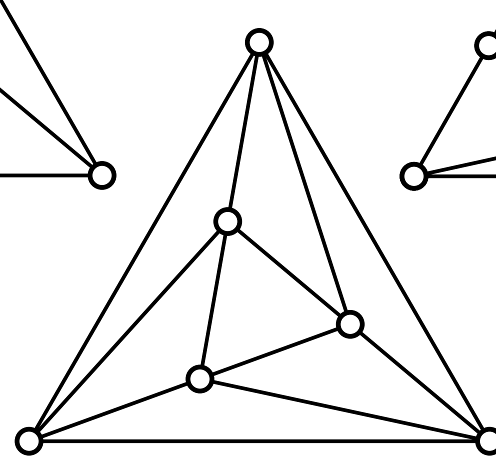
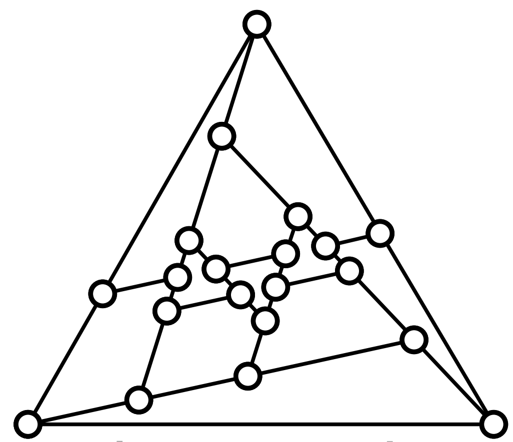
$G = (V, E)$	$\|V\|$	$\|E\|$	$\|F\|$	$\rho^1_2(G)$	$\rho^1_3(G)$	$\pi^1_2(G)$	$\pi^1_3(G)$
tetrahedron	4	6	4				
cube	8	12	6				
octahedron	6	12	8				
dodecahedron	20	30	12				
icosahedron	12	30	20				



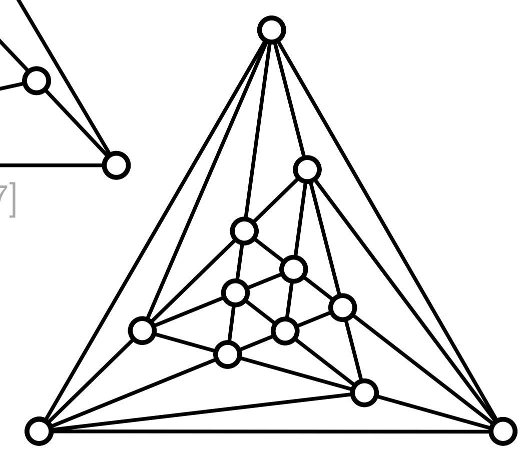
Line Cover Numbers of the Platonic Solids

[Kryven et al., CALDAM'18]

$G = (V, E)$	$\|V\|$	$\|E\|$	$\|F\|$	$\rho^1_2(G)$	$\rho^1_3(G)$	$\pi^1_2(G)$	$\pi^1_3(G)$
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		

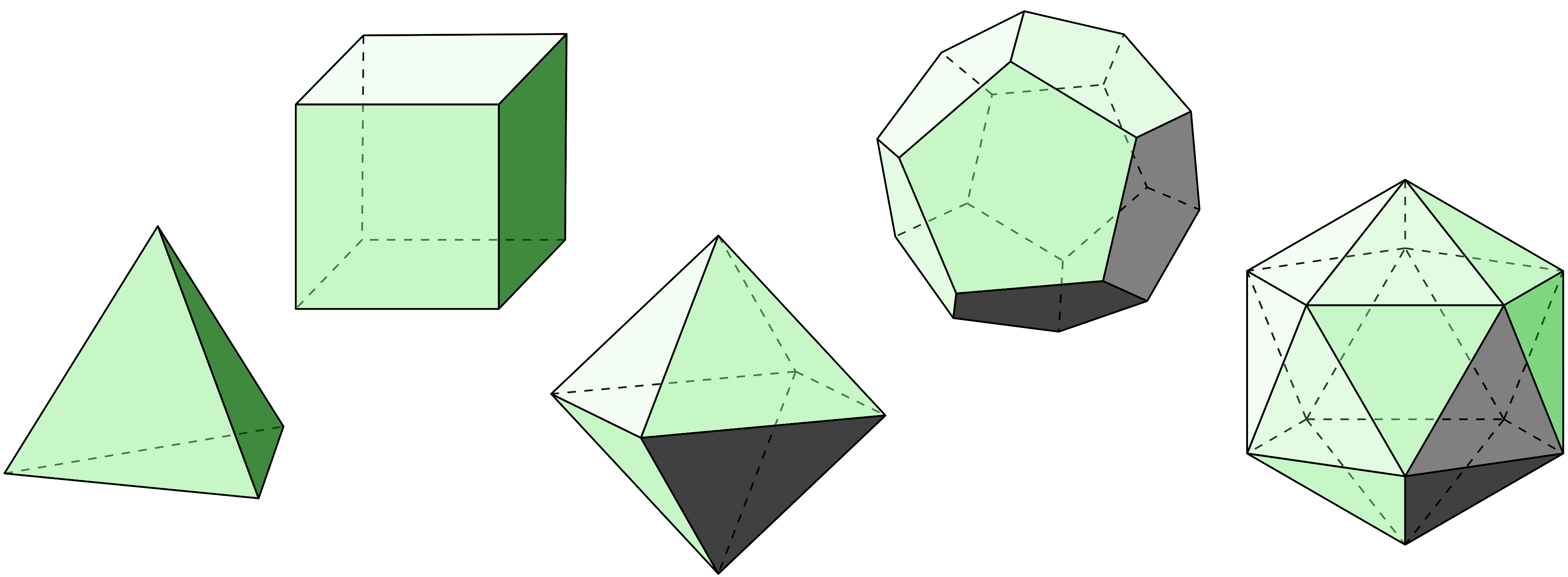





[Scherm, B.Th.'17]



Line Cover Numbers of the Platonic Solids

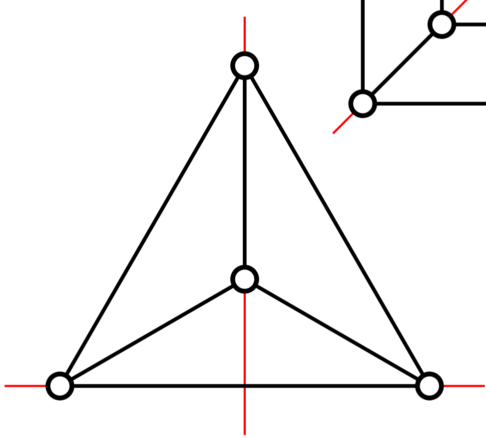
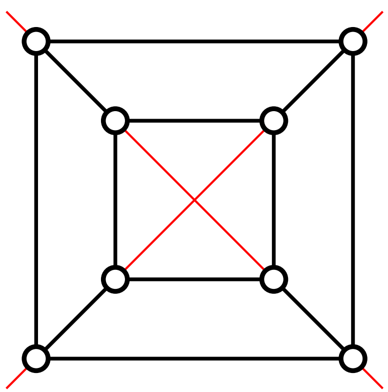
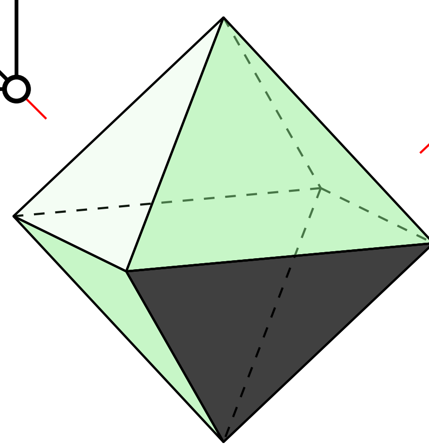
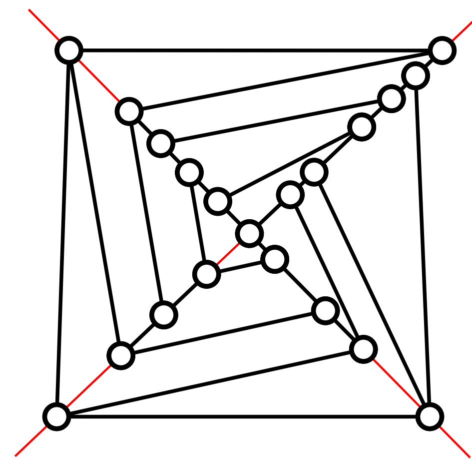
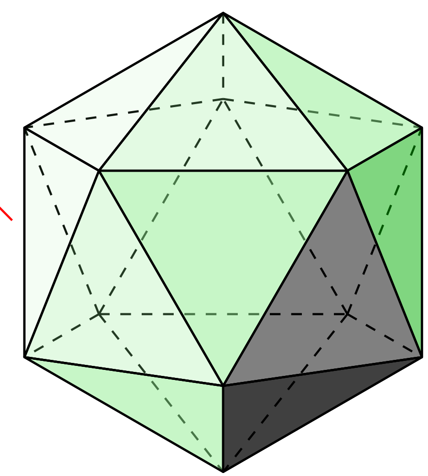
$G=(V,E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho_2^1(G)$ [Kryven et al., CALDAM'18]	$\rho_3^1(G)$ [Kryven et al., CALDAM'18]	$\pi_2^1(G)$ [Firman, MTh. '17]	$\pi_3^1(G)$ [Firman, MTh. '17]
tetrahedron	4	6	4	6	6		
cube	8	12	6	7	7		
octahedron	6	12	8	9	9		
dodecahedron	20	30	12	9...10	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]

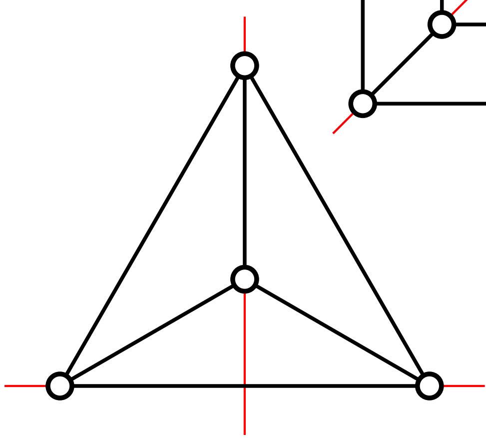
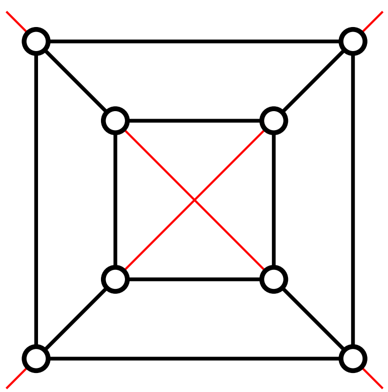
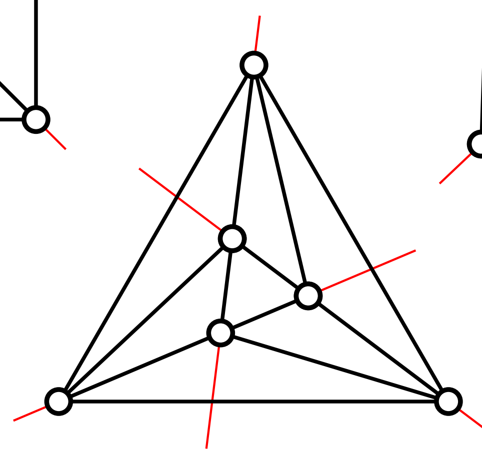
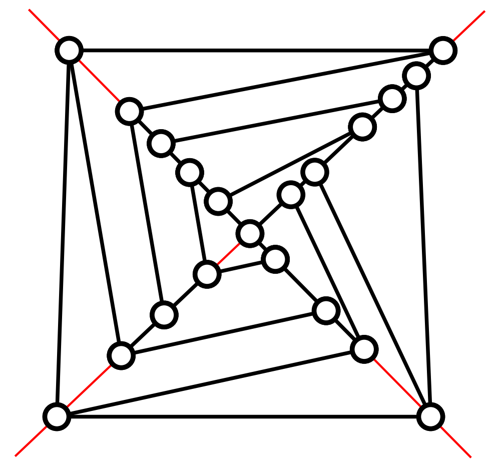
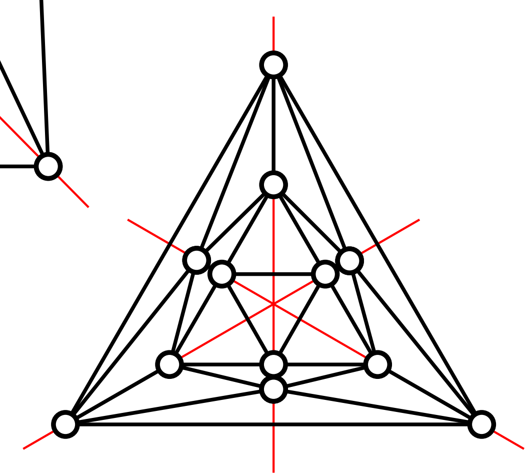
$G = (V, E)$	$\|V\|$	$\|E\|$	$\|F\|$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4	6	6	2	
cube	8	12	6	7	7	2	
octahedron	6	12	8	9	9		
dodecahedron	20	30	12	9...10	9...10	2	
icosahedron	12	30	20	13...15	13...15		

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]

$G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert F\rvert$	$\rho^1_2(G)$	$\rho^1_3(G)$	$\pi^1_2(G)$	$\pi^1_3(G)$
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	$9 \ldots 10$	$9 \ldots 10$	2	2
icosahedron	12	30	20	$13 \ldots 15$	$13 \ldots 15$	3	

Line Cover Numbers of the Platonic Solids

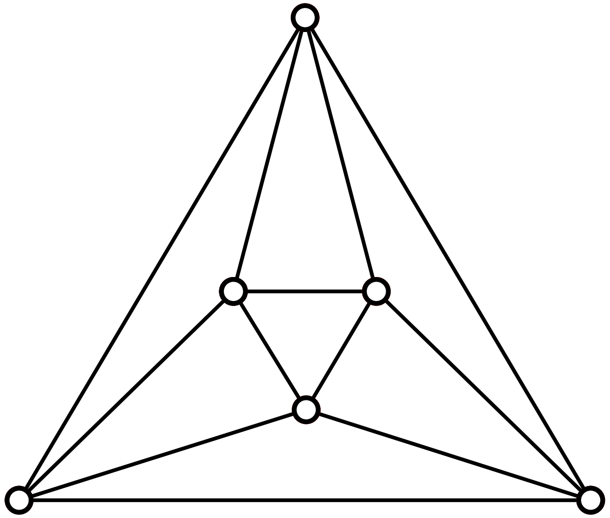
[Kryven et al., CALDAM'18] [Firman, MTh. '17]

$G = (V, E)$	$\|V\|$	$\|E\|$	$\|F\|$	$\rho^1_2(G)$	$\rho^1_3(G)$	$\pi^1_2(G)$	$\pi^1_3(G)$
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]

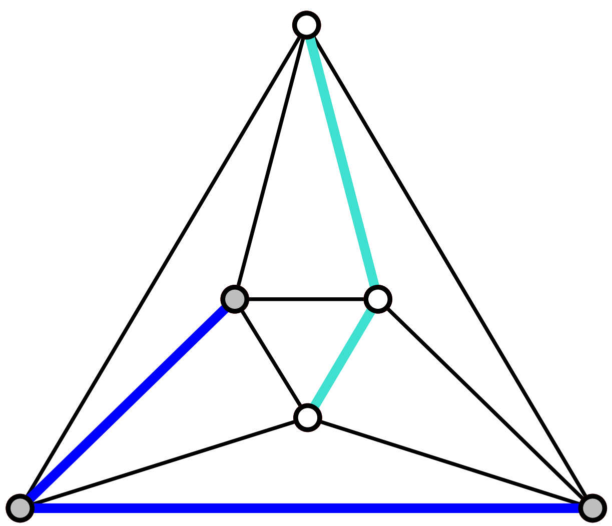
$G = (V, E)$	$\lvert V \rvert$	$\lvert E \rvert$	$\lvert F \rvert$	$\rho^1_2(G)$	$\rho^1_3(G)$	$\pi^1_2(G)$	$\pi^1_3(G)$
tetrahedron	4	6	4	6	6	2	2
cube	8	12	6	7	7	2	2
octahedron	6	12	8	9	9	3	
dodecahedron	20	30	12	9...10	9...10	2	2
icosahedron	12	30	20	13...15	13...15	3	



Line Cover Numbers of the Platonic Solids

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

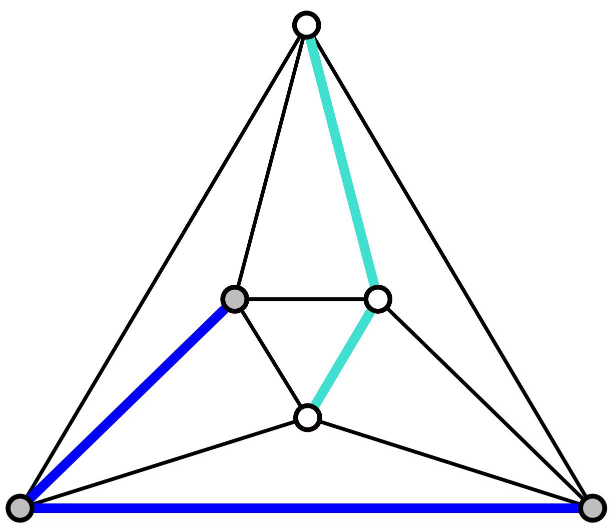
$G = (V, E)$	$\|V\|$	$\|E\|$	$\|F\|$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4	6	6	2	2
cube	8	12	6	7	7	2	2
octahedron	6	12	8	9	9	3	2
dodecahedron	20	30	12	9...10	9...10	2	2
icosahedron	12	30	20	13...15	13...15	3	

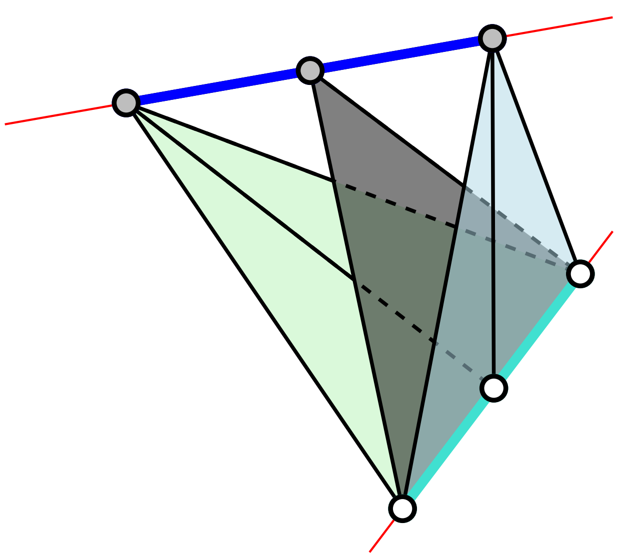


Line Cover Numbers of the Platonic Solids

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

$G = (V, E)$	$\lvert V \rvert$	$\lvert E \rvert$	$\lvert F \rvert$	$\rho^1_2(G)$	$\rho^1_3(G)$	$\pi^1_2(G)$	$\pi^1_3(G)$
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	

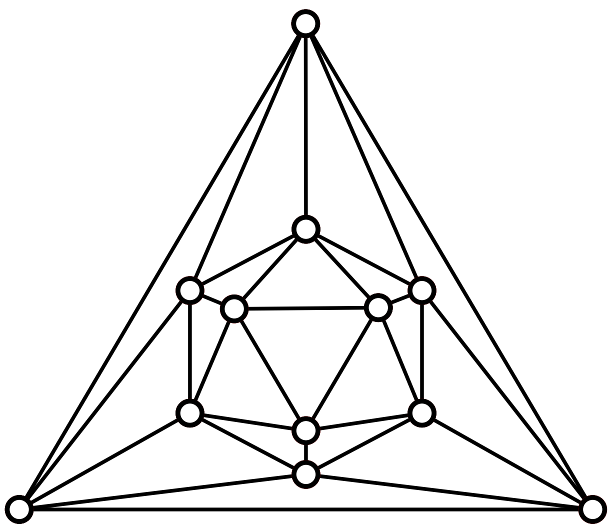




Line Cover Numbers of the Platonic Solids

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

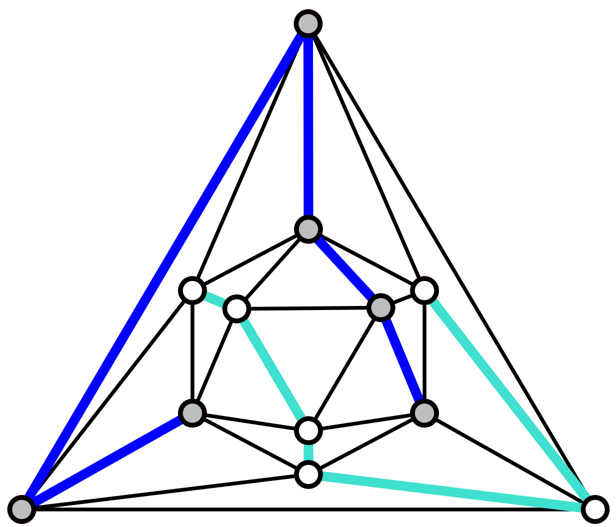
$G = (V, E)$	$\|V\|$	$\|E\|$	$\|F\|$	$\rho^1_2(G)$	$\rho^1_3(G)$	$\pi^1_2(G)$	$\pi^1_3(G)$
tetrahedron	4	6	4	6	6	2	2
cube	8	12	6	7	7	2	2
octahedron	6	12	8	9	9	3	2
dodecahedron	20	30	12	9...10	9...10	2	2
icosahedron	12	30	20	13...15	13...15	3	



Line Cover Numbers of the Platonic Solids

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

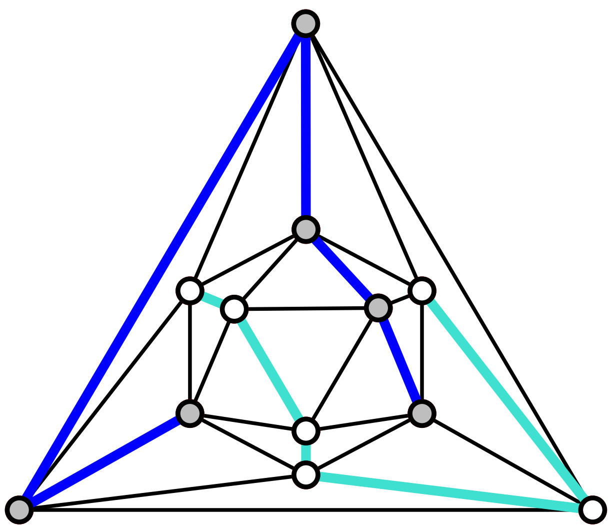
$G = (V, E)$	$\|V\|$	$\|E\|$	$\|F\|$	$\rho_2^1(G)$	$\rho_3^1(G)$	$\pi_2^1(G)$	$\pi_3^1(G)$
tetrahedron	4	6	4	6	6	2	2
cube	8	12	6	7	7	2	2
octahedron	6	12	8	9	9	3	2
dodecahedron	20	30	12	$9 \dots 10$	$9 \dots 10$	2	2
icosahedron	12	30	20	$13 \dots 15$	$13 \dots 15$	3	2

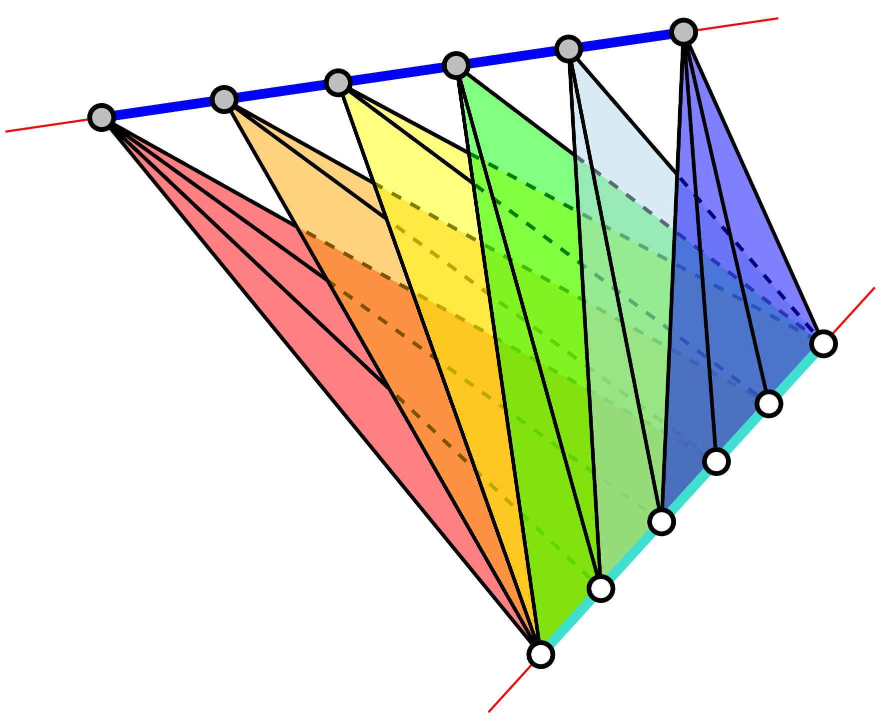


Line Cover Numbers of the Platonic Solids

[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





Graphs vs. Sets

Graphs vs. Sets

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

Graphs vs. Sets

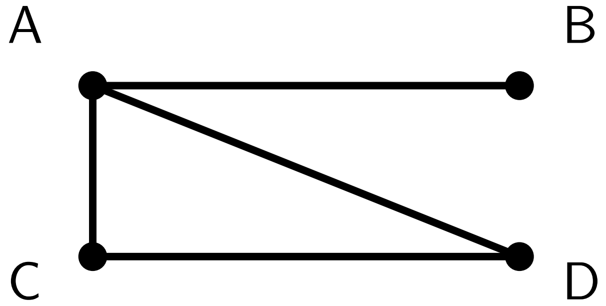
- Graphs are defined by a set of edges, which are sets of two elements.

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

Graphs vs. Sets

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

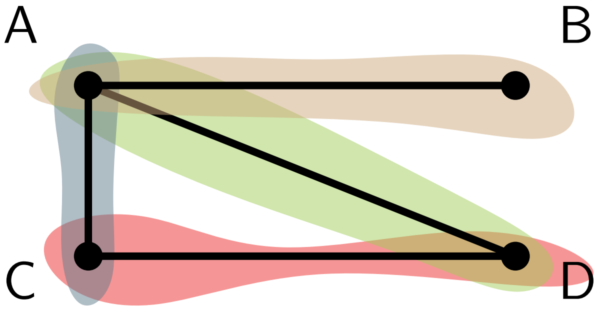
$$\{A, B\}, \{C, D\},$$
$$\{D, A\}, \{A, C\}$$



Graphs vs. Sets

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

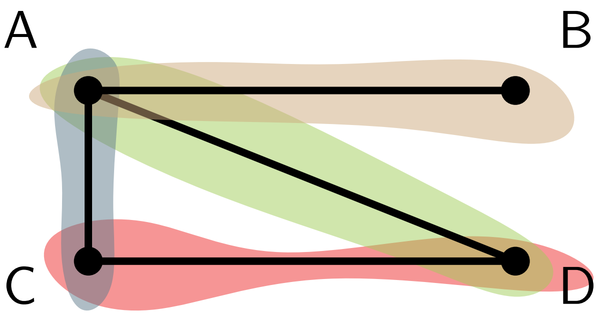
$$\{A, B\}, \{C, D\},$$
$$\{D, A\}, \{A, C\}$$



Graphs vs. Sets

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

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

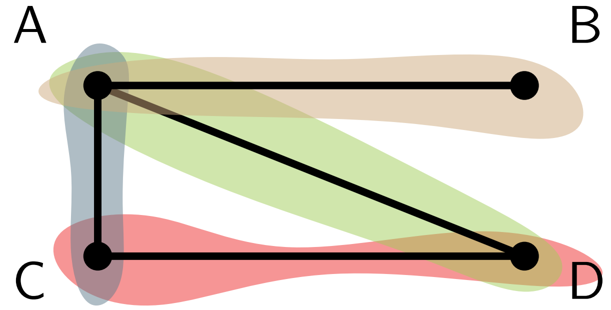


- Hierarchical data can be describes by a tree

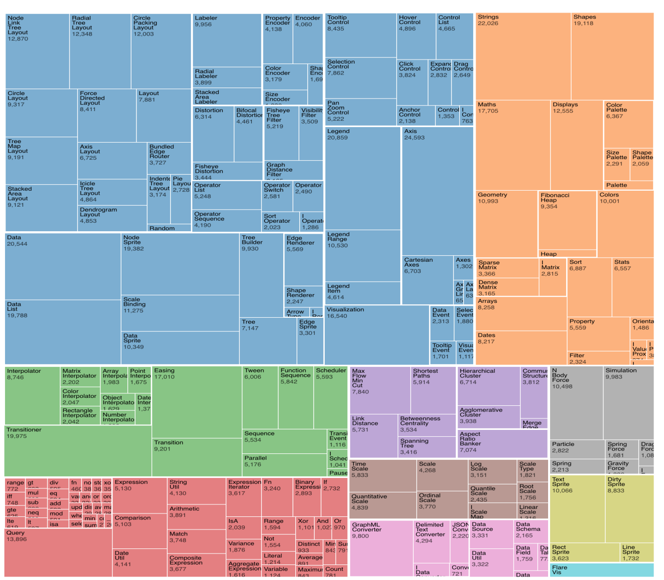
Graphs vs. Sets

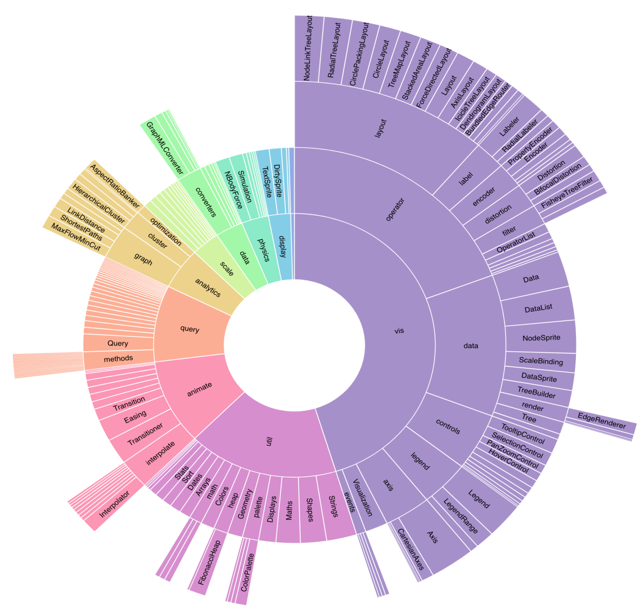
- Graphs are defined by a set of edges, which are sets of two elements.

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



- Hierarchical data can be describes by a tree

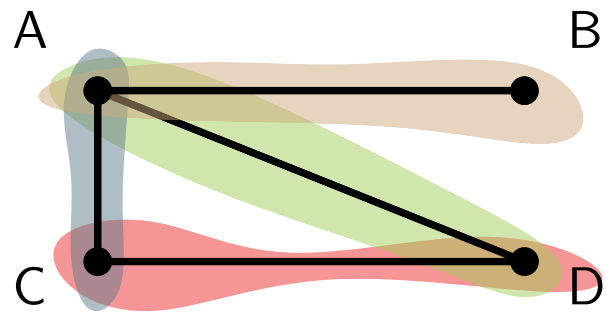




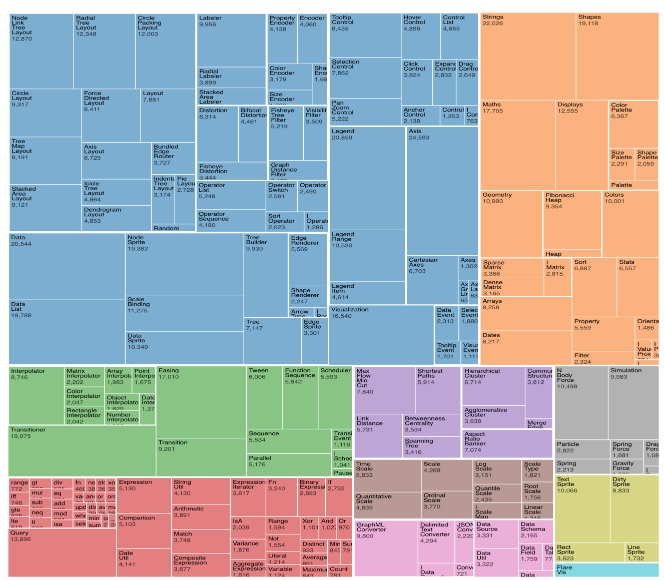
Graphs vs. Sets

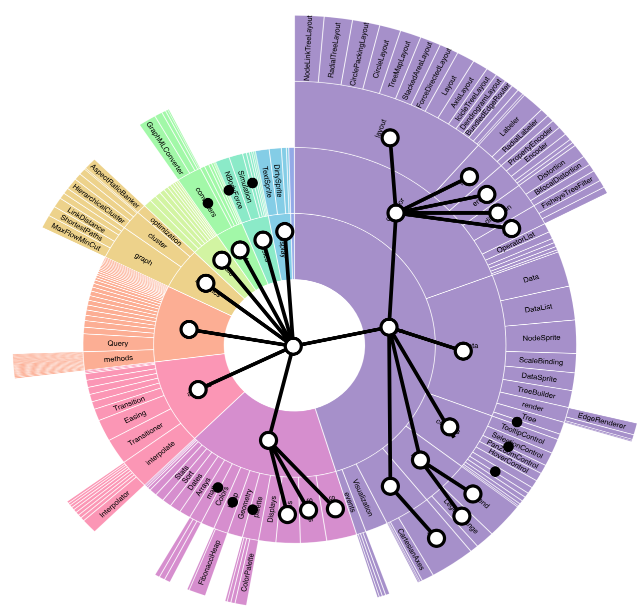
- Graphs are defined by a set of edges, which are sets of two elements.

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



- 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 $G = (V, E)$

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

Hypergraph $H = (V, E)$

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

Example:

$V = \{$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:

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

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

Hypergraph drawing

Hypergraph drawing

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

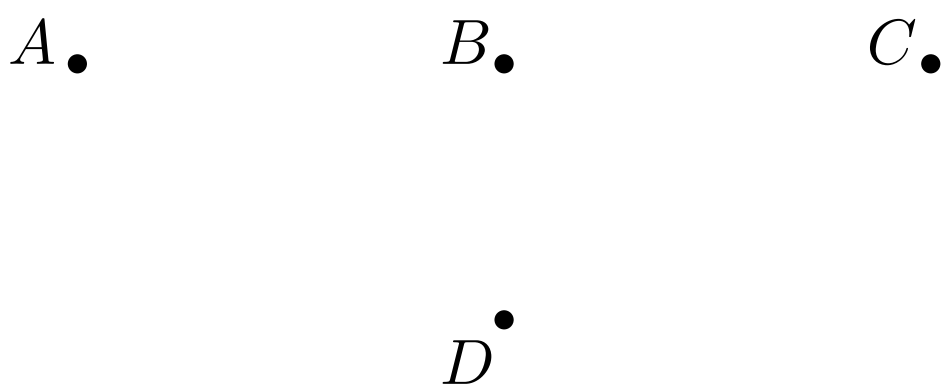
Hypergraph drawing

- 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

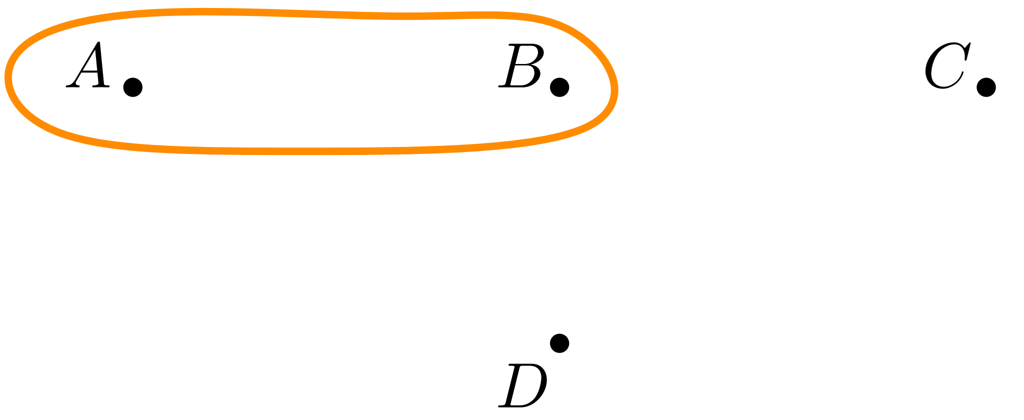
$$H = \Big(\{A,B,C,D\},\{\{A,B\},\{B,C,D\},\{A,D,C\}\}\Big)$$



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

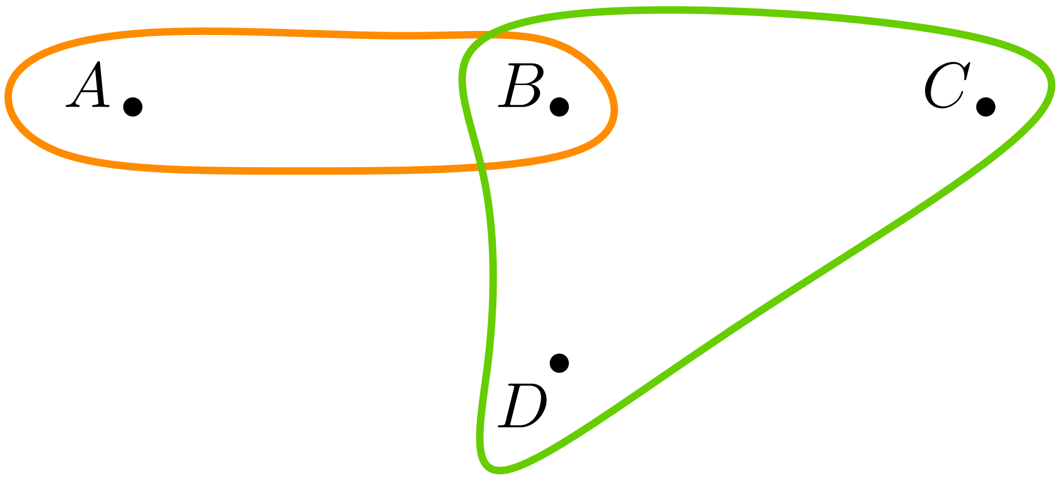
$$H = \Big(\{A, B, C, D\}, \{\underline{\{A, B\}}, \{B, C, D\}, \{A, D, C\}\}\Big)$$



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

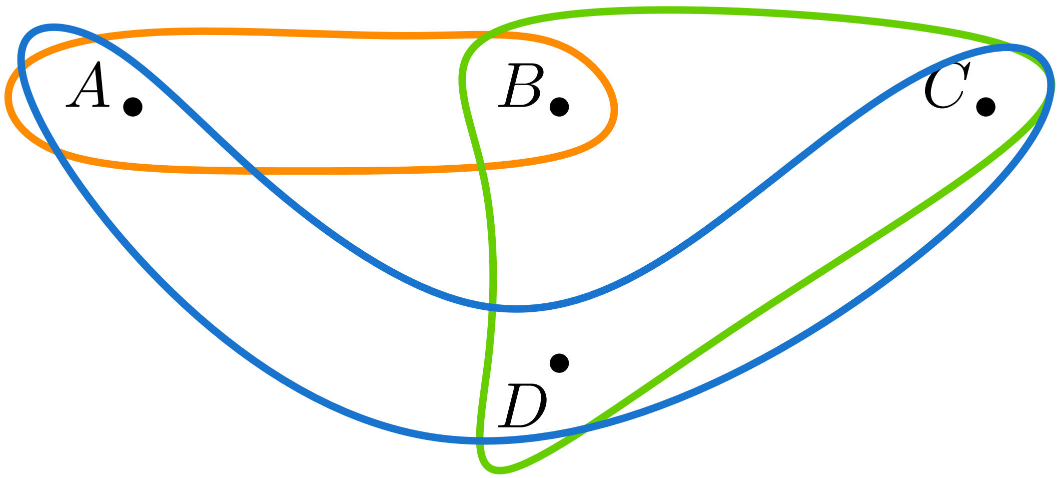
$$H = \Big(\{A, B, C, D\}, \{\{A, B\}, \underline{\{B, C, D\}}, \{A, D, C\}\}\Big)$$



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

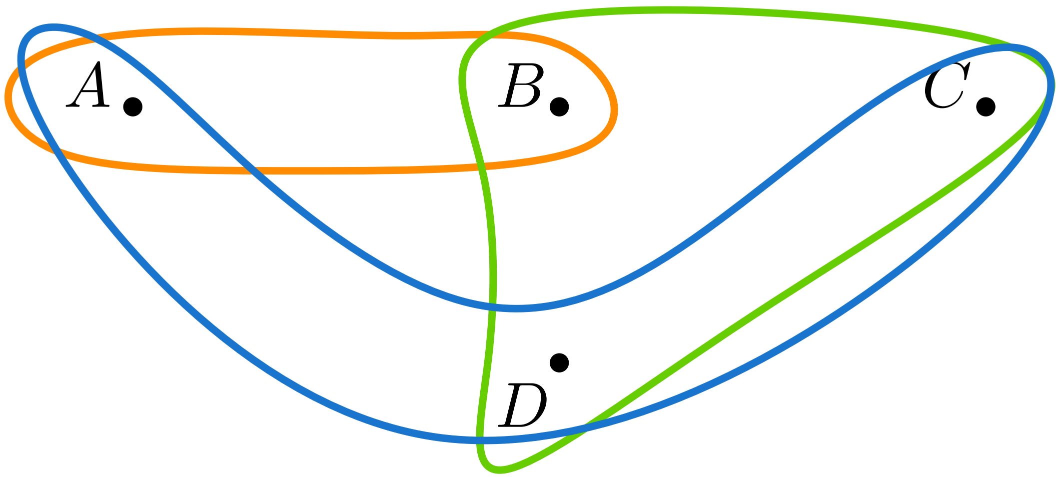
$$H = \Big(\{A, B, C, D\}, \{\{A, B\}, \{B, C, D\}, \underline{\{A, D, C\}}\}\Big)$$



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

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



- 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.

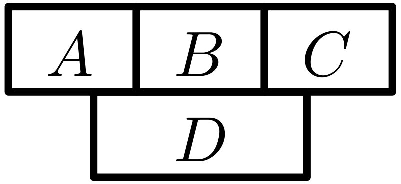
- Subset-based method gets easily confusing.
- **Alternatives:** subdivision-based & edge-based

  - vertices are regions
  - hyperedges yield connected unions
  - ... and more criterions

Hypergraph drawing cont.

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

  - vertices are regions
  - hyperedges yield connected unions
  - ... and more criterions

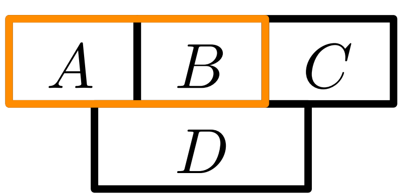


$$H = \Big(\{A, B, C, D\}, \{\{A, B\}, \{B, C, D\}, \{A, D, C\}\}\Big)$$

Hypergraph drawing cont.

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

  – vertices are regions
  – hyperedges yield connected unions
  – ... and more criterions

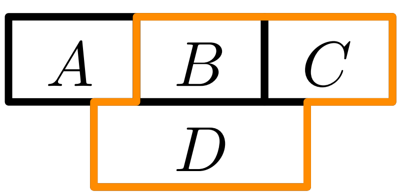


$$H = \Big(\{A, B, C, D\}, \{\underline{\{A, B\}}, \{B, C, D\}, \{A, D, C\}\}\Big)$$

Hypergraph drawing cont.

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

  - vertices are regions
  - hyperedges yield connected unions
  - ... and more criterions

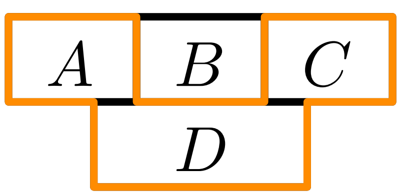


$$H = \left(\{A,B,C,D\}, \{\{A,B\}, \underline{\{B,C,D\}}, \{A,D,C\}\}\right)$$

Hypergraph drawing cont.

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

  – vertices are regions
  – hyperedges yield connected unions
  – ... and more criterions



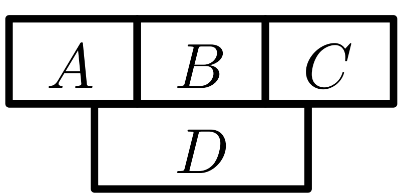
$$H = \Big(\{A, B, C, D\}, \{\{A, B\}, \{B, C, D\}, \{\underline{A, D, C}\}\}\Big)$$

Hypergraph drawing cont.

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

- vertices are regions
- hyperedges yield connected unions
- ... and more criterions

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



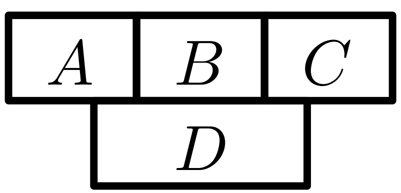
$$H = \Big(\{A, B, C, D\}, \{\{A, B\}, \{B, C, D\}, \{A, D, C\}\}\Big)$$

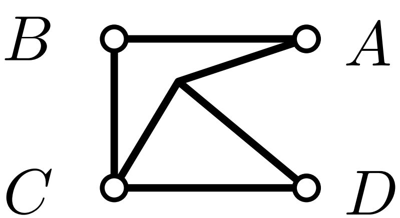
Hypergraph drawing cont.

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

- vertices are regions
- hyperedges yield connected unions
- ... and more criterions

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



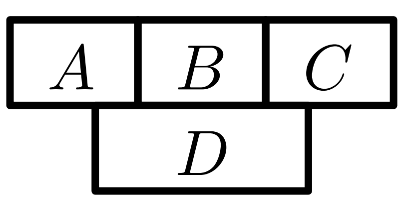


$$H = \Big(\{A, B, C, D\}, \{\{A, B\}, \{B, C, D\}, \{A, D, C\}\}\Big)$$

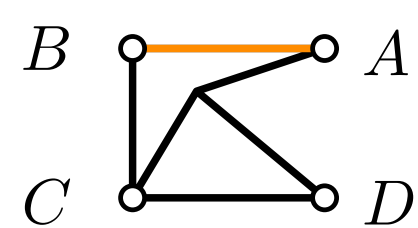
Hypergraph drawing cont.

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

Subdivision-based:
- vertices are regions
- hyperedges yield connected unions
- … and more criterions



Edge-based:
- drawn as node-link diagram, with vertices as some nodes
- hyperedges yield connected subgraphs
- … and more criteria



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

Hypergraph drawing cont.

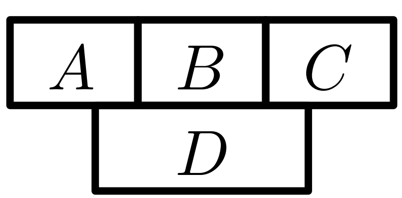
- Subset-based method gets easily confusing.
- **Alternatives:** subdivision-based & edge-based

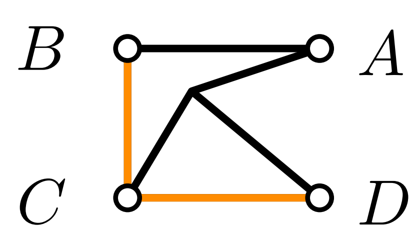
Subdivision-based:

- vertices are regions
- hyperedges yield connected unions
- ... and more criterions

Edge-based:

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





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

Hypergraph drawing cont.

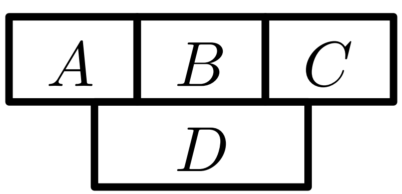
- Subset-based method gets easily confusing.
- **Alternatives:** subdivision-based & edge-based

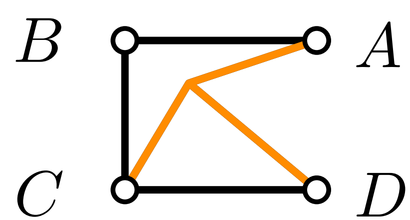
Subdivision-based:

- vertices are regions
- hyperedges yield connected unions
- … and more criterions

Edge-based:

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





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

Subdivision-based methods

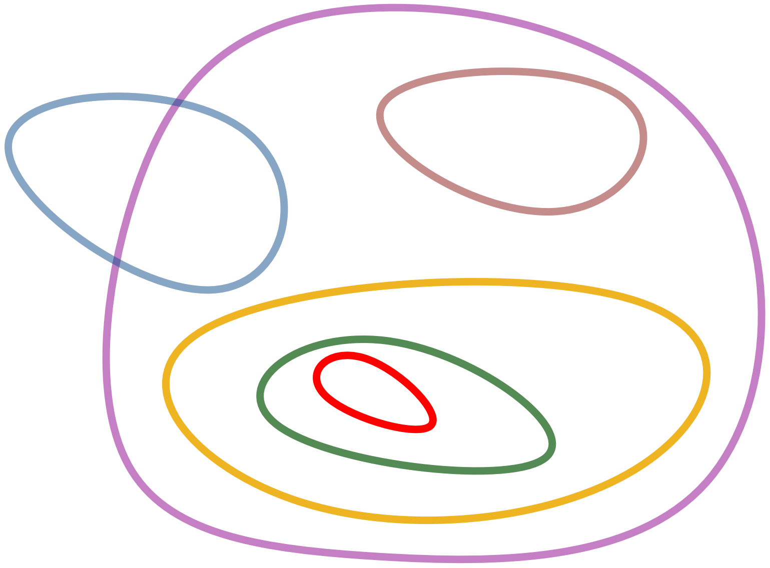
Subdivision-based methods

Concrete Euler Diagrams

Subdivision-based methods

Concrete Euler Diagrams

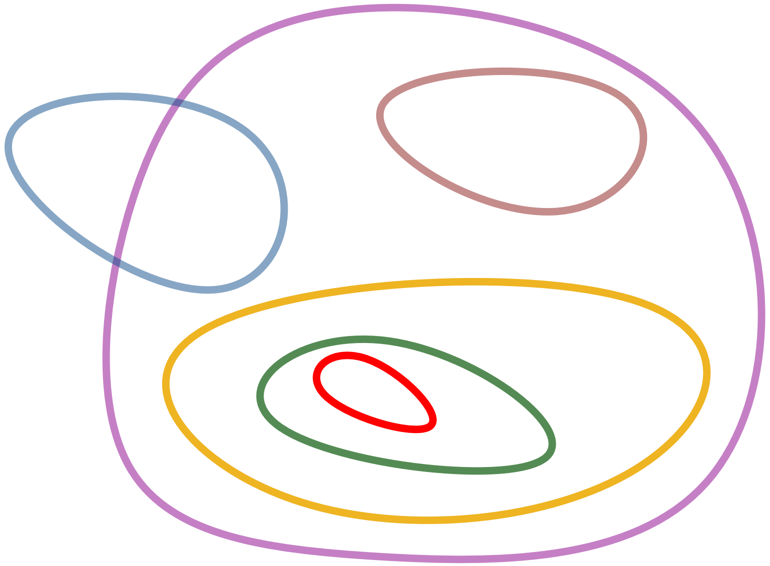
- hyperedges are drawn as simple closed curves (interior/exterior)



Subdivision-based methods

Concrete Euler Diagrams

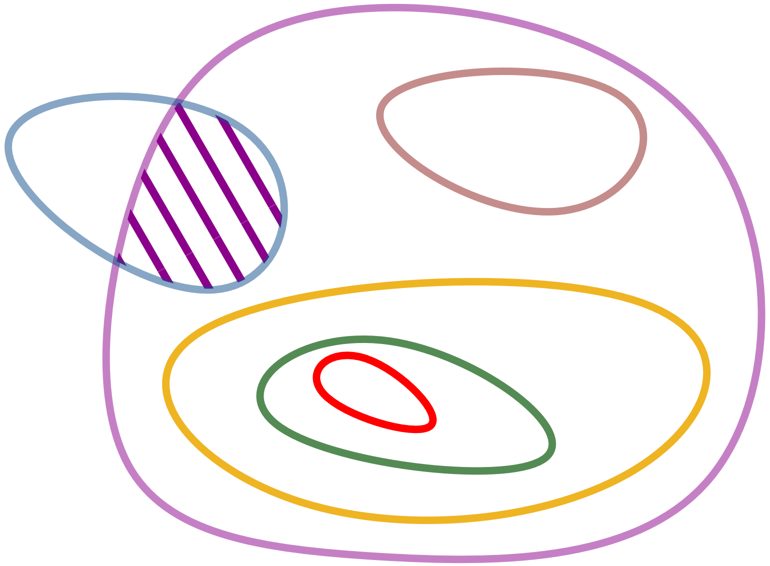
- hyperedges are drawn as simple closed curves (interior/exterior)
- intersections of hyperedges = zone



Subdivision-based methods

Concrete Euler Diagrams

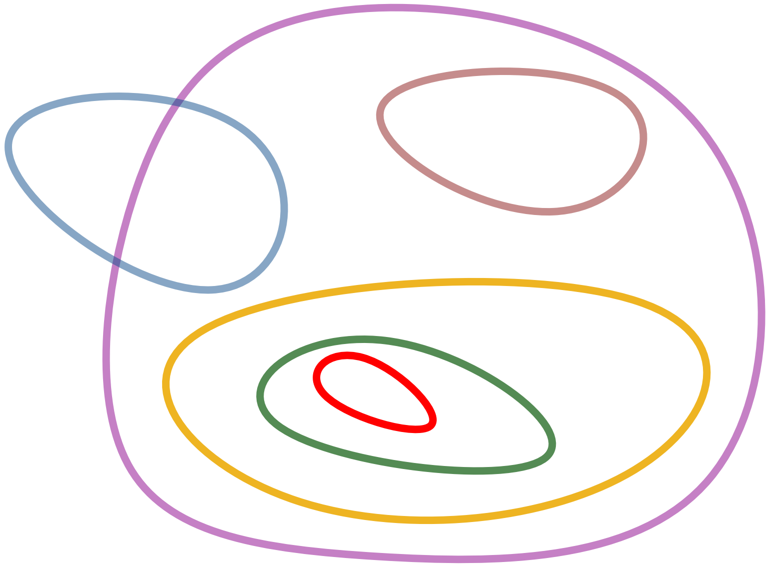
- hyperedges are drawn as simple closed curves (interior/exterior)
- intersections of hyperedges = zone



Subdivision-based methods

Concrete Euler Diagrams

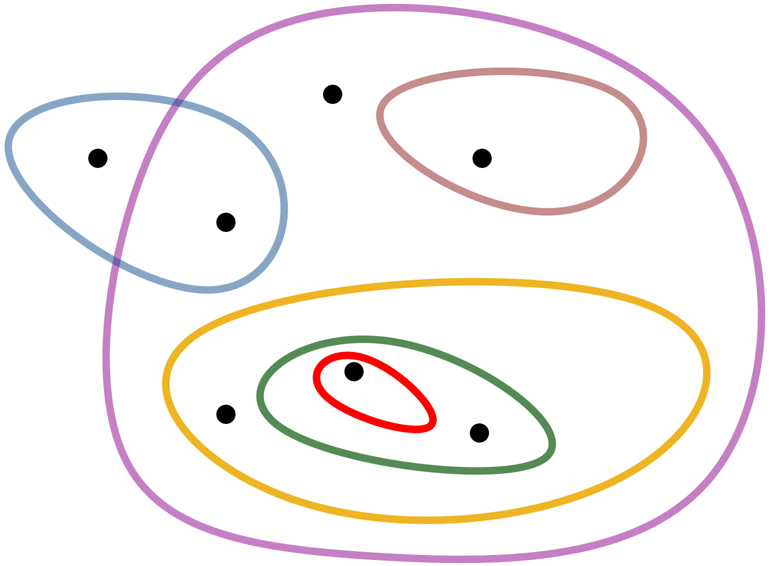
- hyperedges are drawn as simple closed curves (interior/exterior)
- intersections of hyperedges = zone
- for every zone there is a vertex in the corresponding intersection



Subdivision-based methods

Concrete Euler Diagrams

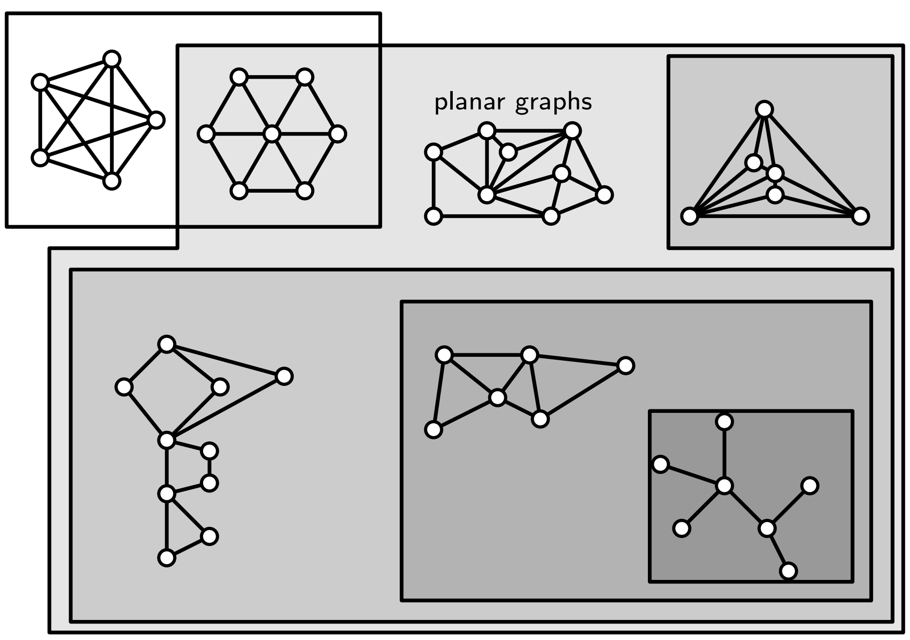
- hyperedges are drawn as simple closed curves (interior/exterior)
- intersections of hyperedges = zone
- for every zone there is a vertex in the corresponding intersection

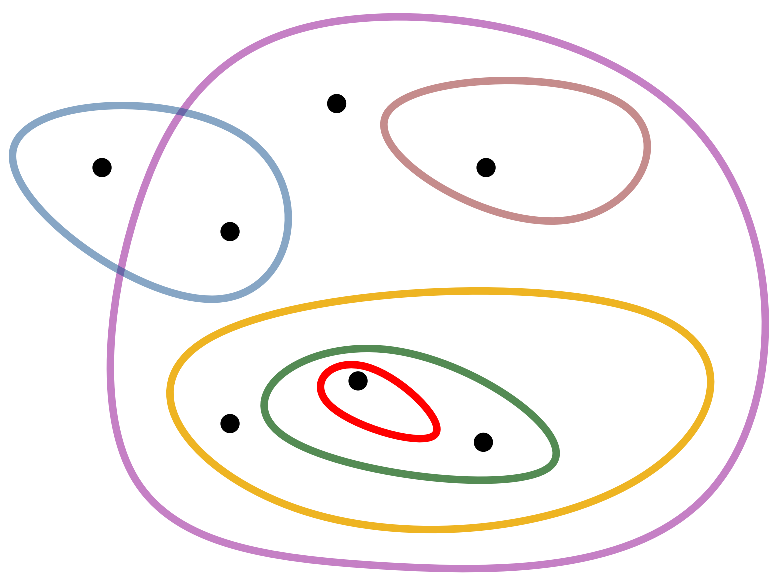


Subdivision-based methods

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

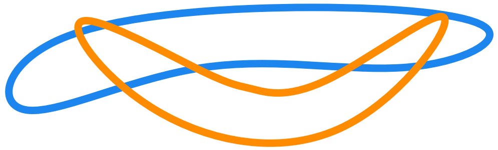




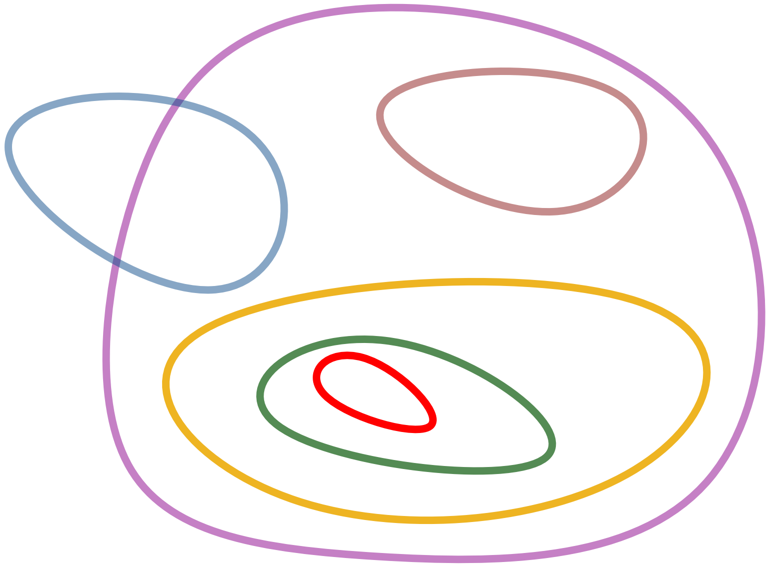
Subdivision-based methods

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



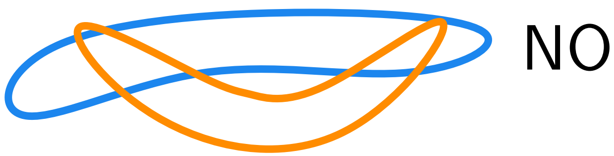
NO



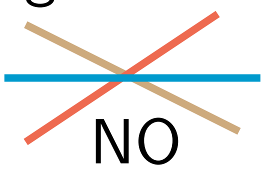
Subdivision-based methods

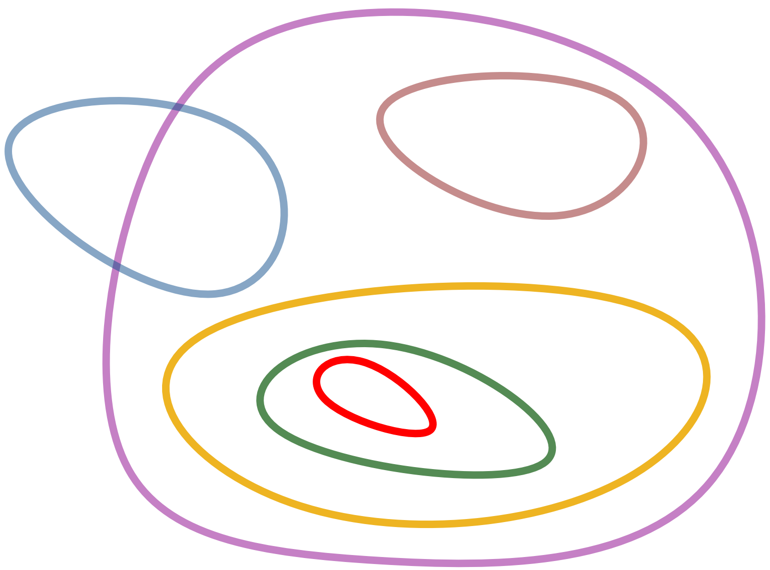
Concrete Euler Diagrams

- 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

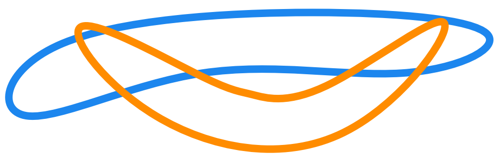




Subdivision-based methods

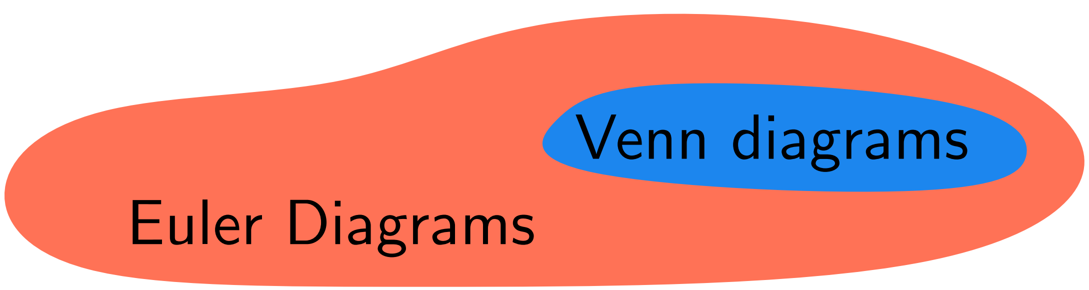
Concrete Euler Diagrams

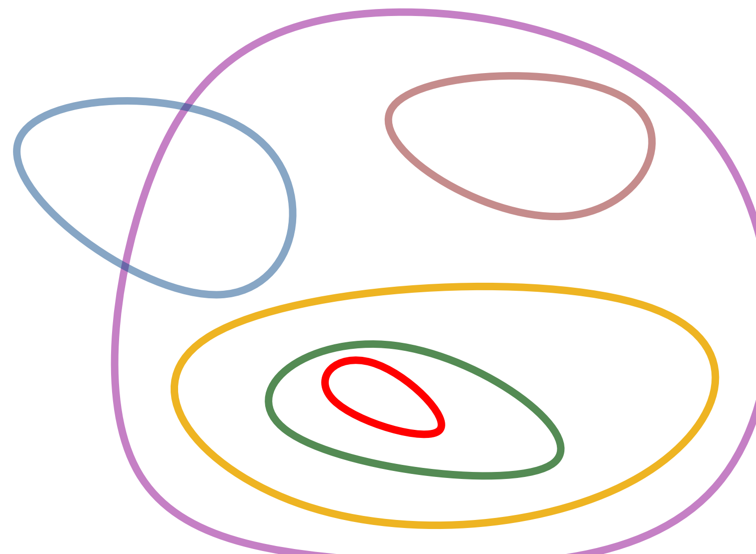
- hyperedges are drawn as simple closed
- intersections of hyperedges = zone
- for every zone there is a vertex in the c
- no two zones for the same intersection



NO

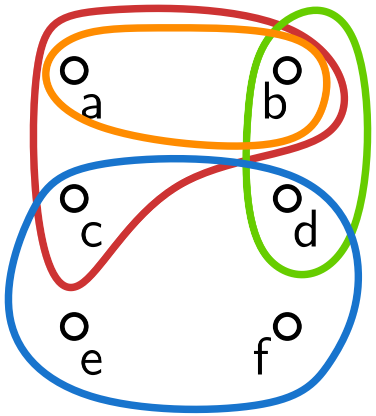
- only proper crossings





Edge-based methods

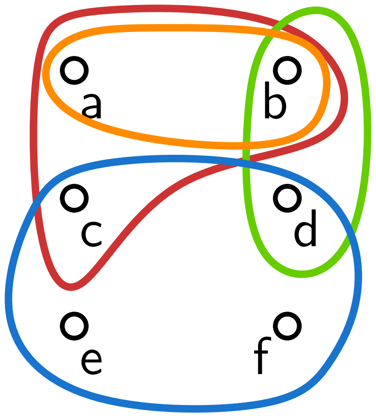
Edge-based methods



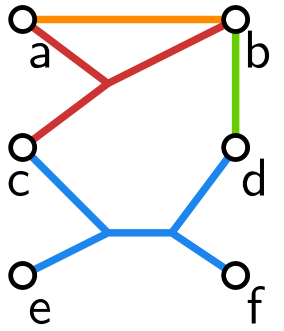
subset-based
drawing

Edge-based methods

Examples:



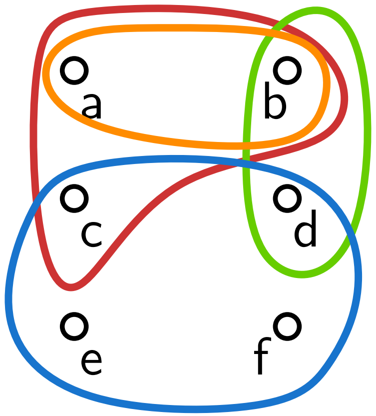
subset-based
drawing



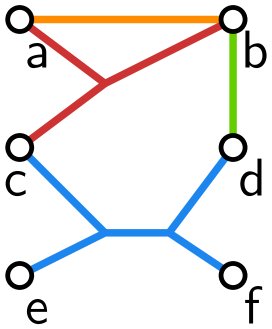
edge-based
drawing

Edge-based methods

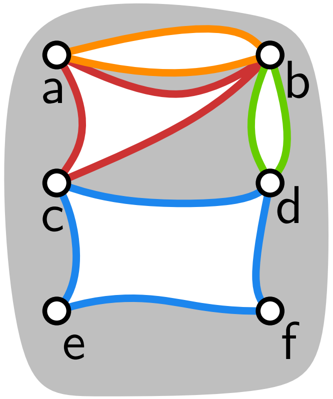
Examples:



subset-based drawing



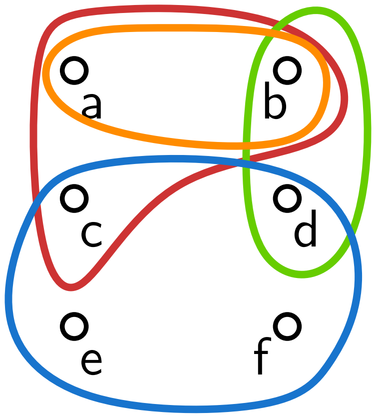
edge-based drawing



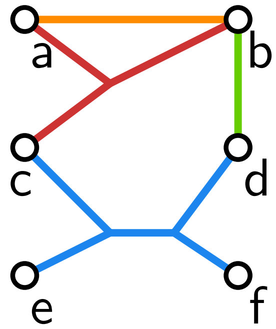
Zykov representation

Edge-based methods

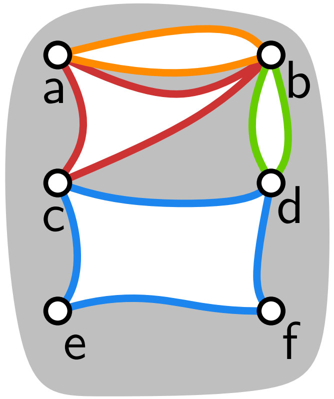
Examples:



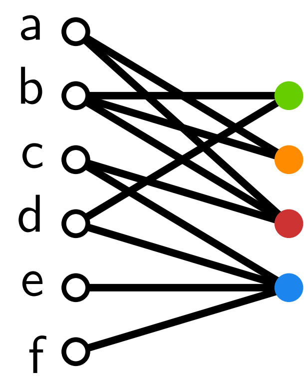
subset-based drawing



edge-based drawing



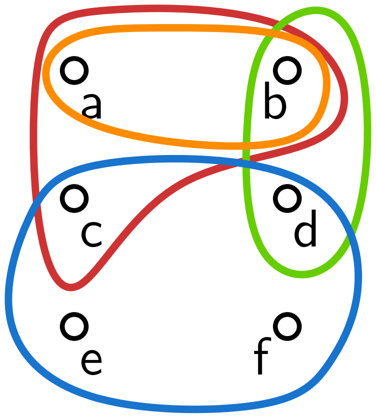
Zykov representation



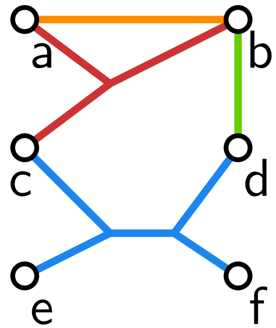
incidence representation

Edge-based methods

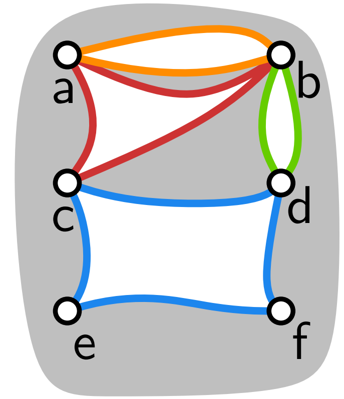
Examples:



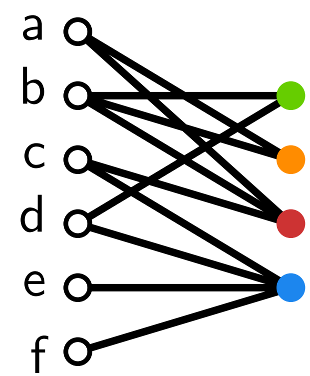
subset-based drawing



edge-based drawing



Zykov representation

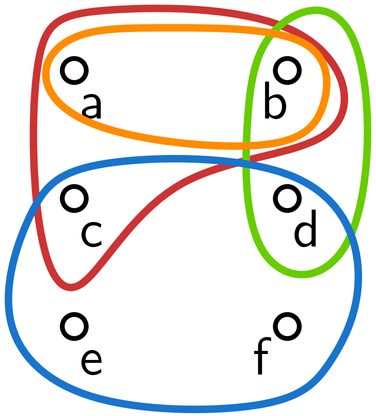


incidence representation

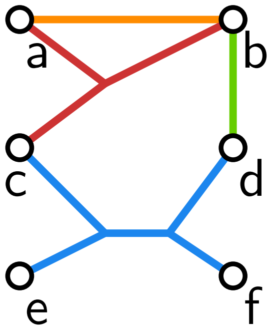
planarity is equivalent in all three models

Edge-based methods

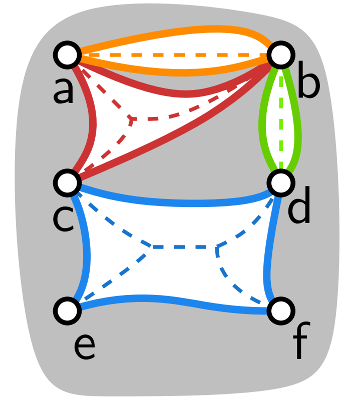
Examples:



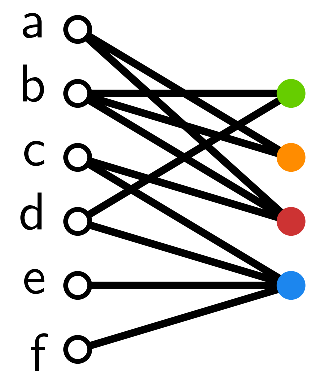
subset-based drawing



edge-based drawing



Zykov representation

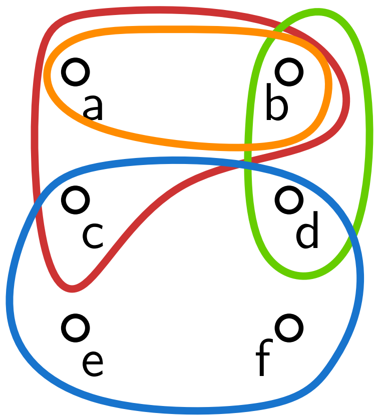


incidence representation

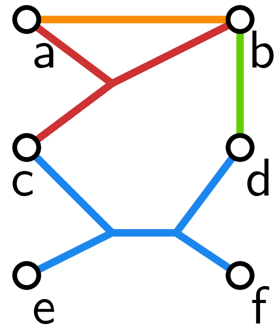
planarity is equivalent in all three models

Edge-based methods

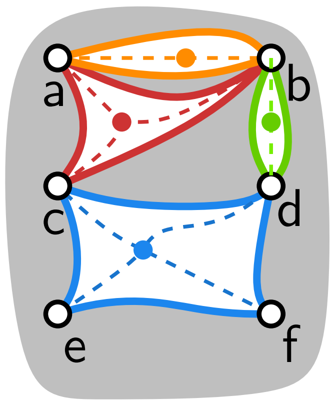
Examples:



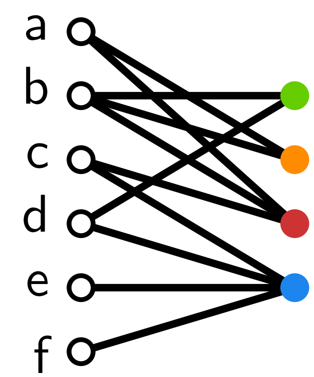
subset-based drawing



edge-based drawing



Zykov representation

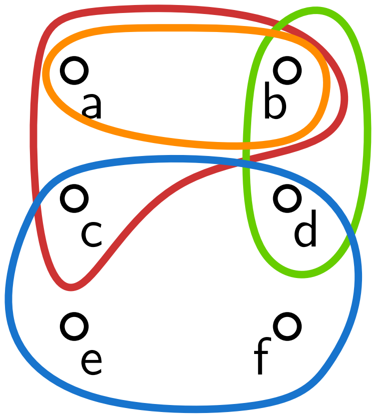


incidence representation

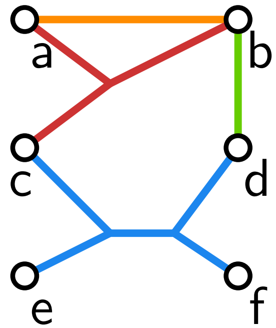
planarity is equivalent in all three models

Edge-based methods

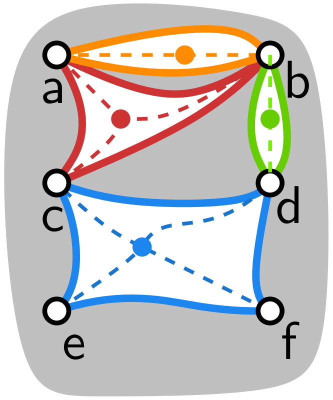
Examples:



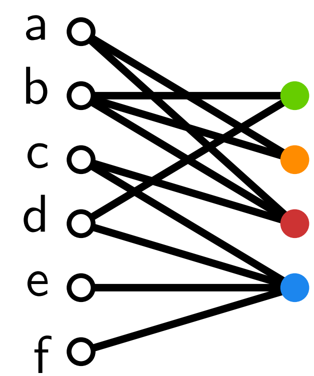
subset-based drawing



edge-based drawing



Zykov representation



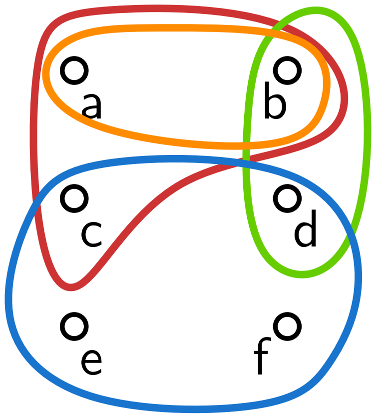
incidence representation

“planarity” NP-complete

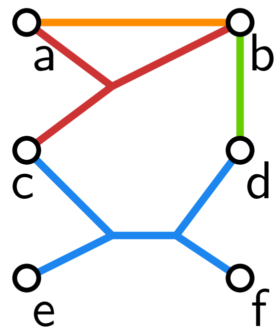
planarity is equivalent in all three models $\in$ P

Edge-based methods

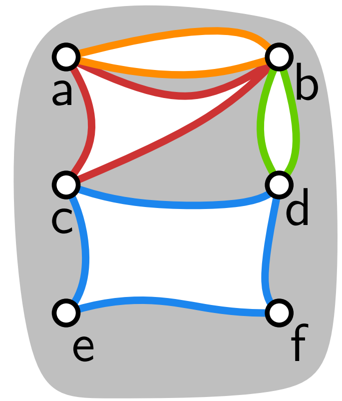
Examples:



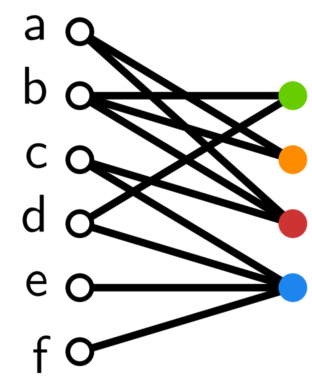
subset-based drawing



edge-based drawing



Zykov representation

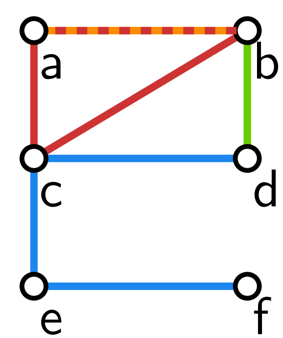


incidence representation

“planarity” NP-complete

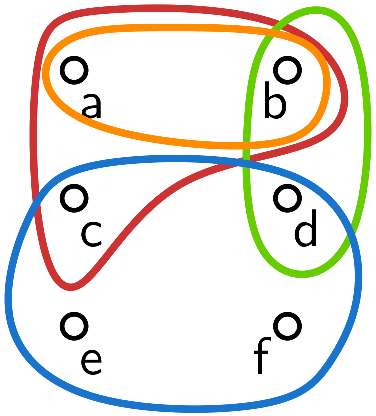
planarity is equivalent in all three models $\in$ P

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

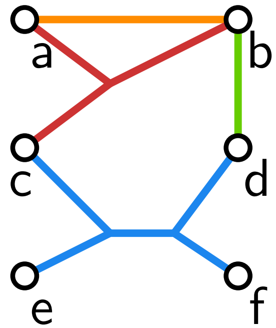


Edge-based methods

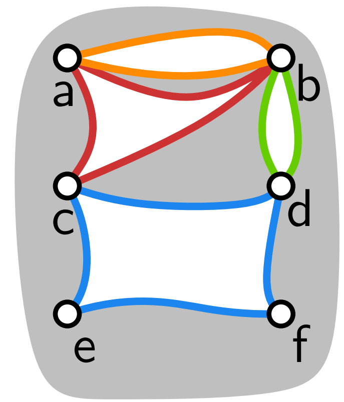
Examples:



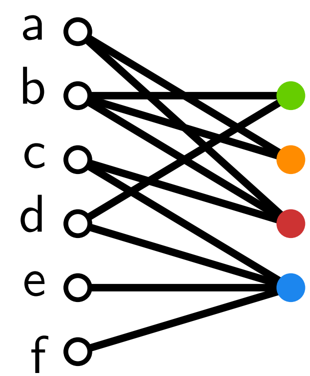
subset-based drawing



edge-based drawing



Zykov representation



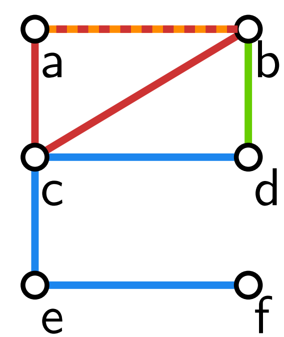
incidence representation

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

"planarity"
NP-complete

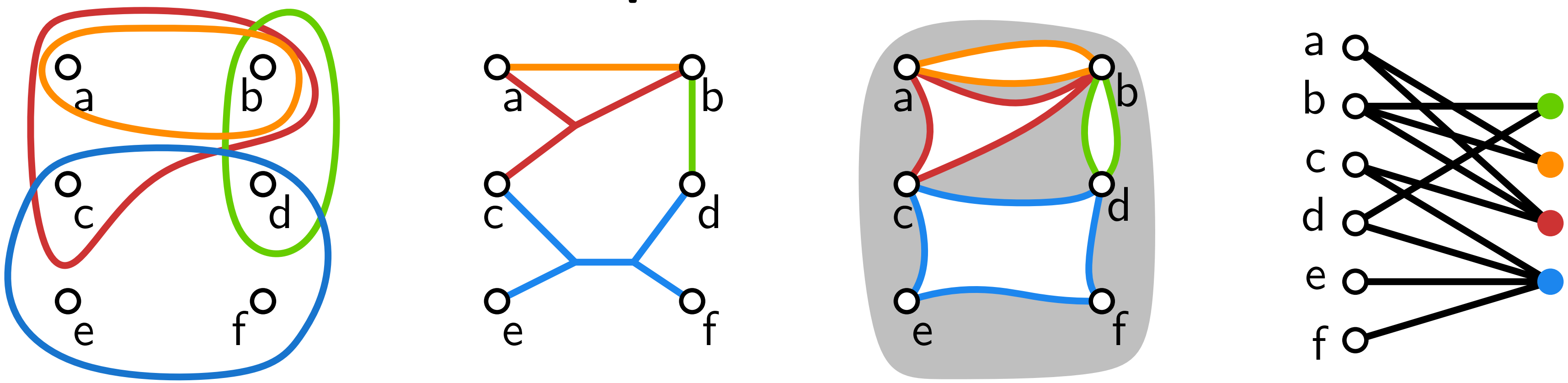
Def.: Support of a hypergraph is a graph such that every hyperedge induces a connected subgraph.

= planar support



Edge-based methods

Examples:



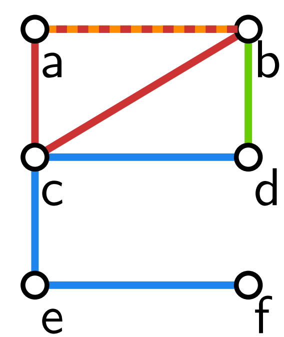
subset-based drawing | edge-based drawing | Zykov representation | incidence representation

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.

Simultaneous embeddings of hypergraphs

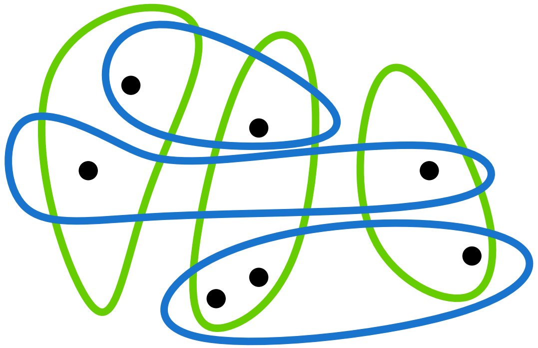
Simultaneous embeddings of hypergraphs

- Athenstädt et al. [GD'14] studied embeddings of two *partitions*:

Simultaneous embeddings of hypergraphs

- Athenstädt et al. [GD'14] studied embeddings of two *partitions*:

  3 Models

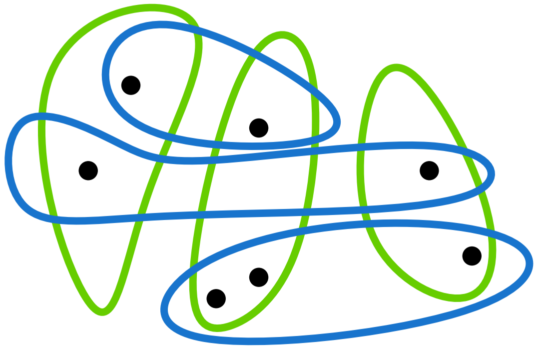


weak embedding
(no crossing sets)

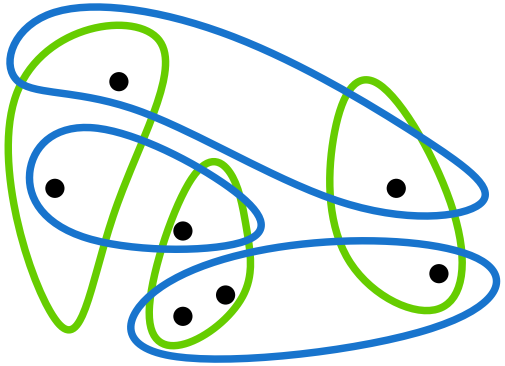
Simultaneous embeddings of hypergraphs

- Athenstädt et al. [GD'14] studied embeddings of two *partitions*:

  3 Models



weak embedding
(no crossing sets)

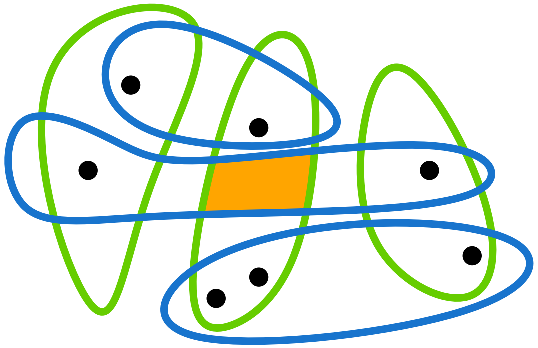


strong embedding
(+ no empty intersections)

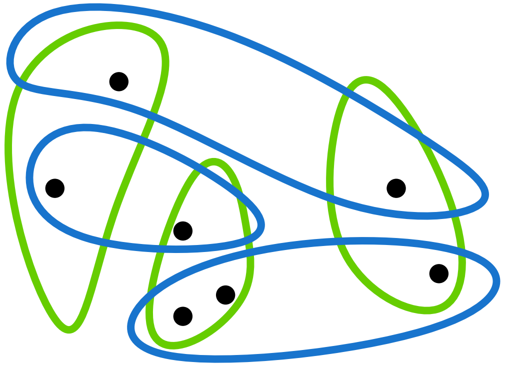
Simultaneous embeddings of hypergraphs

- Athenstädt et al. [GD'14] studied embeddings of two *partitions*:

  3 Models



weak embedding
(no crossing sets)

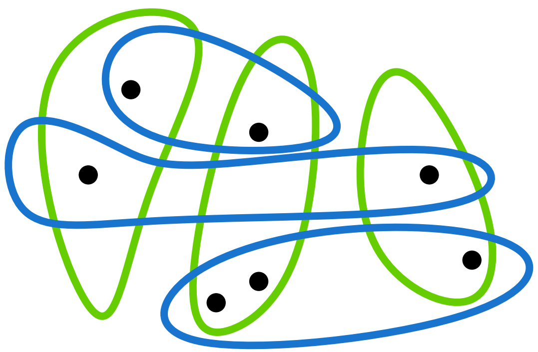


strong embedding
(+ no empty intersections)

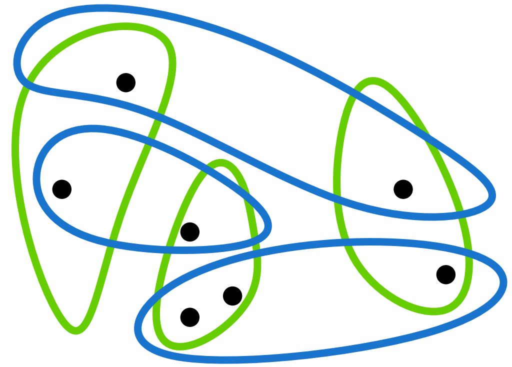
Simultaneous embeddings of hypergraphs

- Athenstädt et al. [GD'14] studied embeddings of two *partitions*:

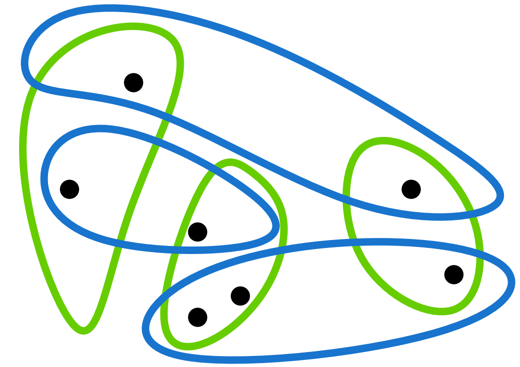
  3 Models



weak embedding
(no crossing sets)



strong embedding
(+ no empty intersections)

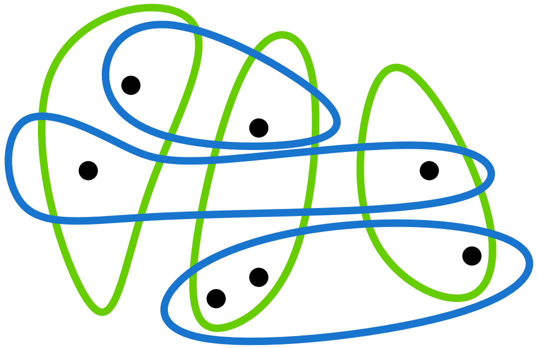


full embedding
(+ only pseudodisks)

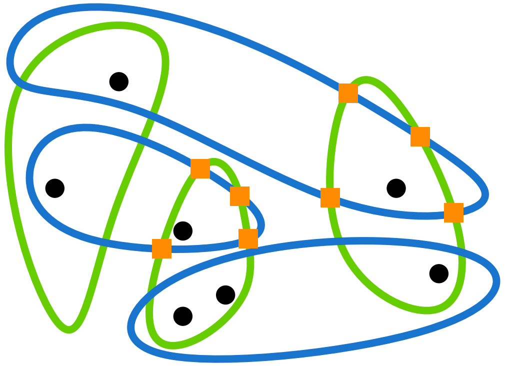
Simultaneous embeddings of hypergraphs

- Athenstädt et al. [GD'14] studied embeddings of two *partitions*:

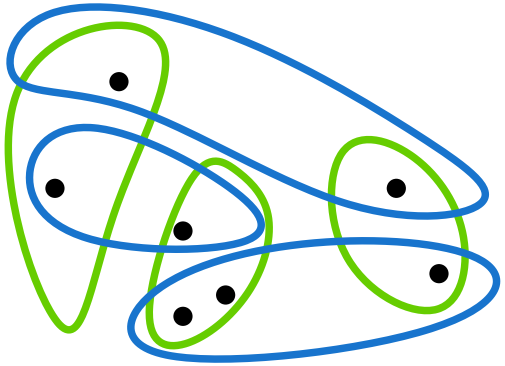
  3 Models



weak embedding (no crossing sets)



strong embedding (+ no empty intersections)

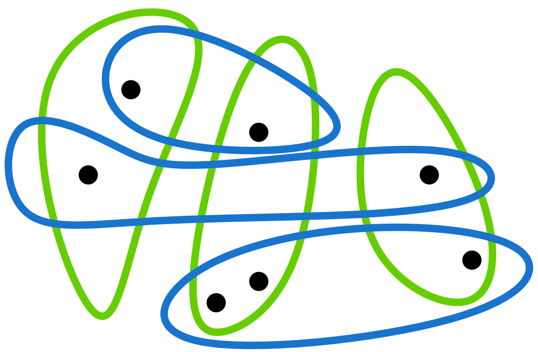


full embedding (+ only pseudodisks)

Simultaneous embeddings of hypergraphs

- Athenstädt et al. [GD'14] studied embeddings of two *partitions*:

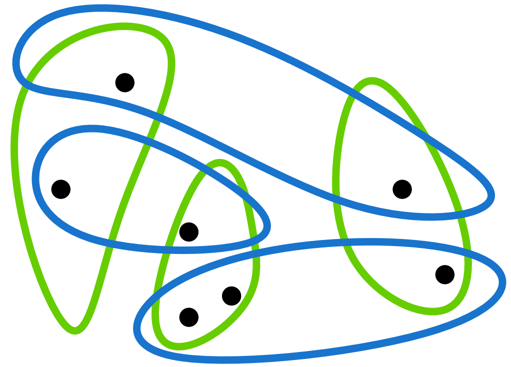
  3 Models



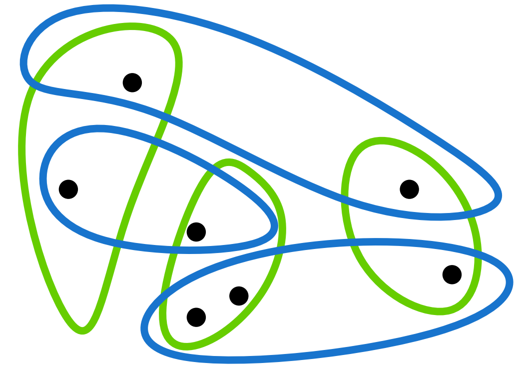
weak embedding
(no crossing sets)

↑

always exists



strong embedding
(+ no empty intersections)

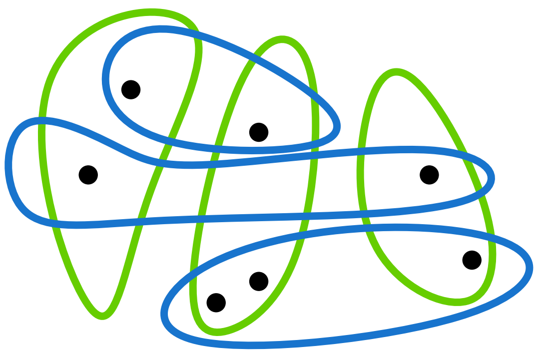


full embedding
(+ only pseudodisks)

Simultaneous embeddings of hypergraphs

- Athenstädt et al. [GD'14] studied embeddings of two *partitions*:

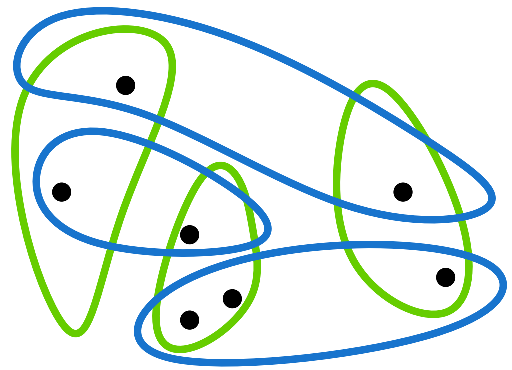
3 Models



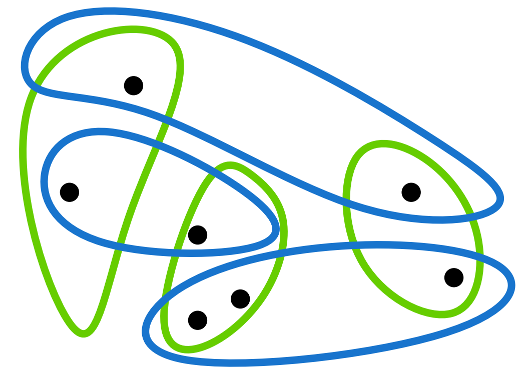
weak embedding
(no crossing sets)

↑

always exists



strong embedding
(+ no empty intersections)



full embedding
(+ only pseudodisks)

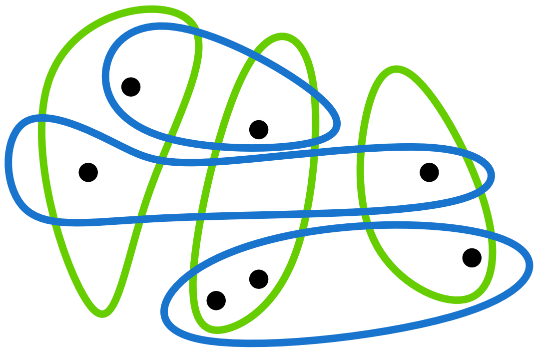
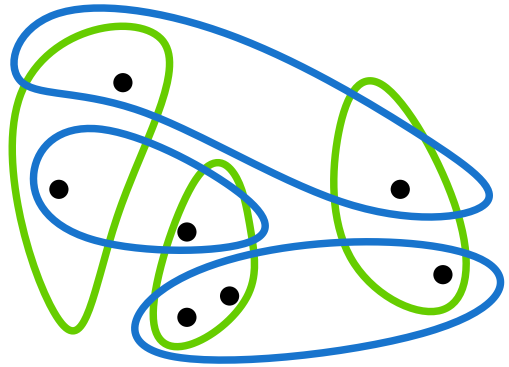
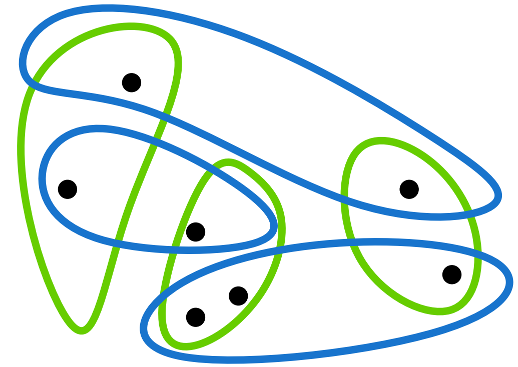
↑

can be tested efficiently

Simultaneous embeddings of hypergraphs

- Athenstädt et al. [GD'14] studied embeddings of two *partitions*:

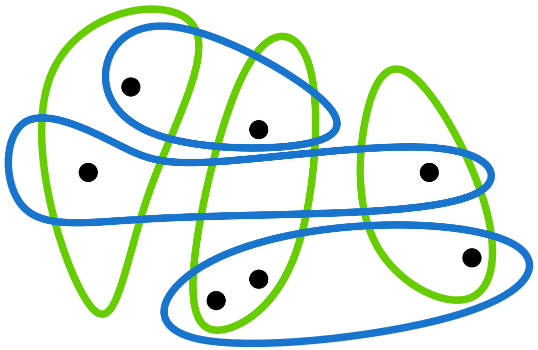
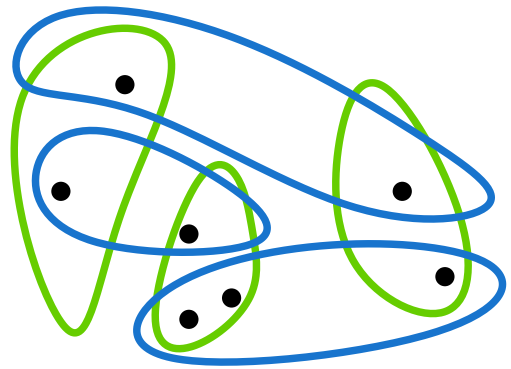
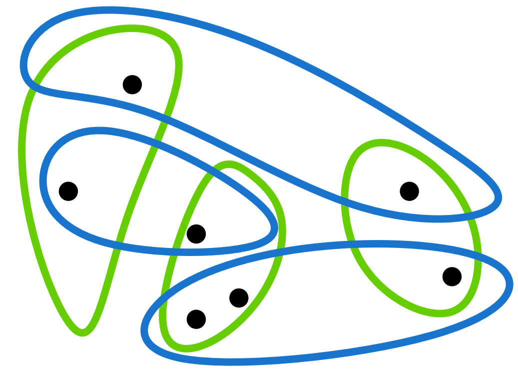
  3 Models

		
weak embedding (no crossing sets)	strong embedding (+ no empty intersections)	full embedding (+ only pseudodisks)
↑	↑	↑
always exists	**test is NP-hard**	**can be tested efficiently**

Simultaneous embeddings of hypergraphs

- Athenstädt et al. [GD'14] studied embeddings of two *partitions*:

  3 Models

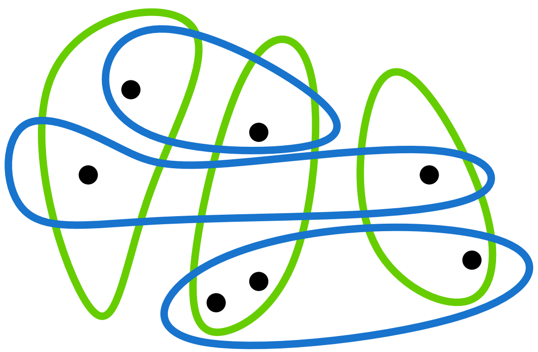
		
weak embedding (no crossing sets)	strong embedding (+ no empty intersections)	full embedding (+ only pseudodisks)
↑	↑	↑
always exists	**test is NP-hard**	**can be tested efficiently**

- No results for more than 2 partitions or more general hypergraphs.

Simultaneous embeddings of hypergraphs

- Athenstädt et al. [GD'14] studied embeddings of two *partitions*:

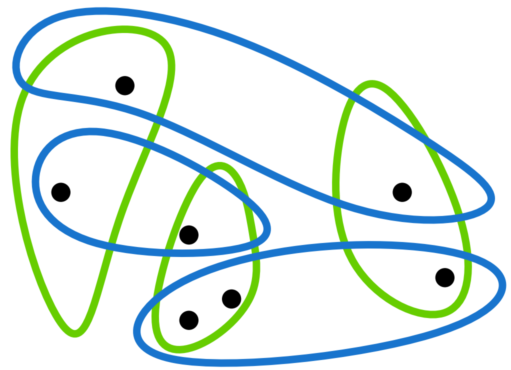
  3 Models



weak embedding
(no crossing sets)

↑

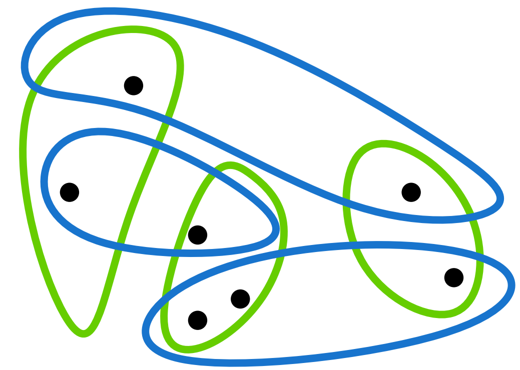
always exists



strong embedding
(+ no empty intersections)

↑

test is NP-hard



full embedding
(+ only pseudodisks)

↑

can be tested efficiently

- No results for more than 2 partitions or more general hypergraphs.
  E.g., *linear* hypergraphs, where 2 hyperedges share at most 1 vertex.

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
  - 3d graph drawing
  - 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
  - 3d graph drawing
  - layered drawings
  - bus drawings
  - more subdivision drawings for hypergraphs
  - ...

Thank you.