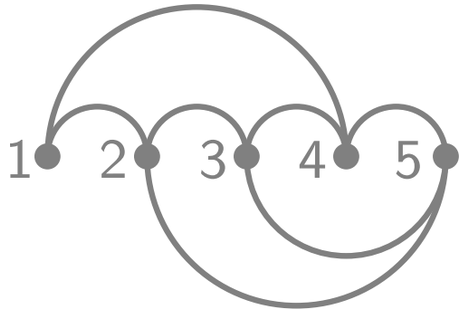
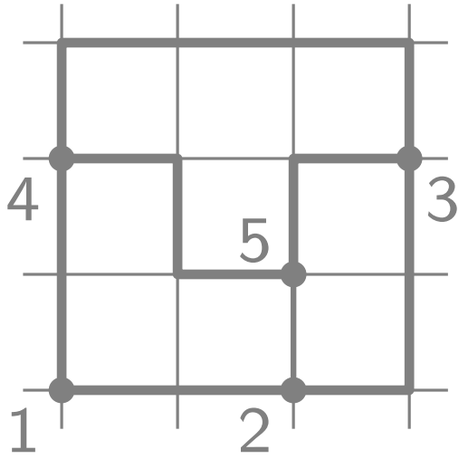
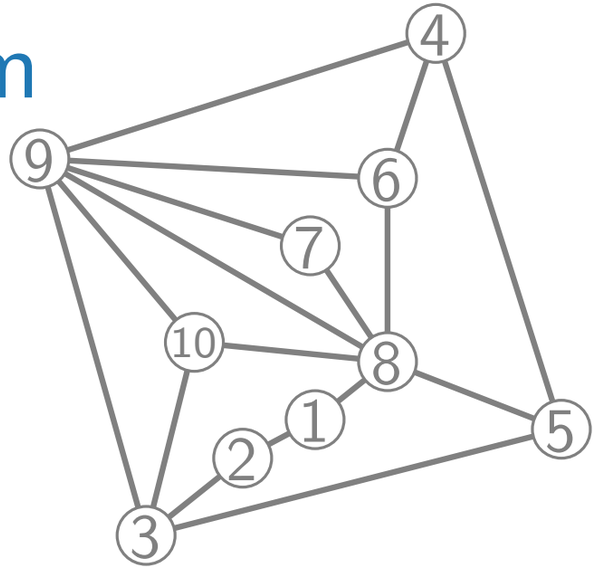
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Visualization of Graphs



Lecture 0:
Introduction
The Graph Visualization Problem





Alexander Wolff

Summer term 2026

Organizational

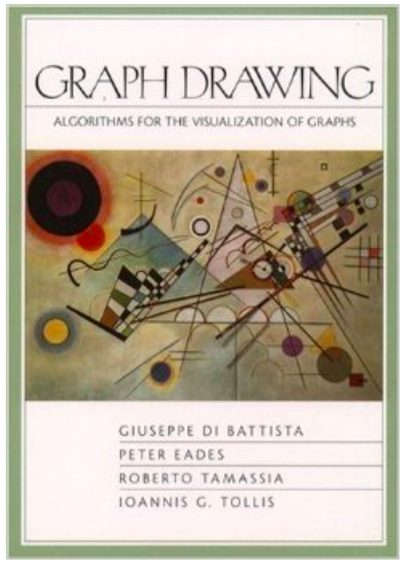
Lectures:

- Alexander Wolff (M4, room 01.001, firstname.familyname@uni-wuerzburg.de)
- Friday, 10:15–11:45, SE II
- videos (in German) from 2021 by Jonathan Klawitter available on WueCampus
- videos (in English) by Philipp Kindermann available on YouTube

Tutorials:

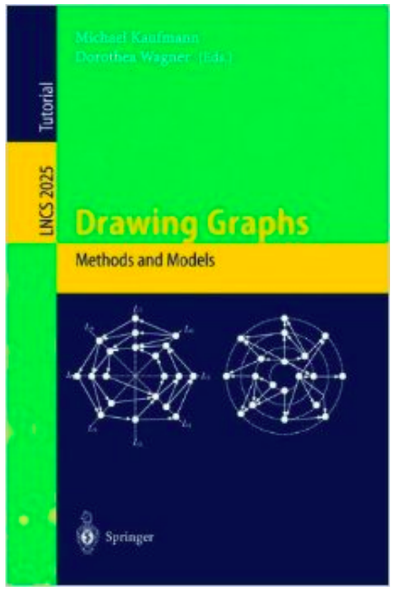
- Adrian Samoticha (firstname.familyname@stud-mail.uni-wuerzburg.de)
- Wednesday, **16:00**–17:30, SE II (first tutorial: April 22)
- One exercise sheet each week (Friday to Friday; first sheet appears today).
- 20 points per sheet
- Average score 50% or more $\Rightarrow$ bonus of 0.3 grade points.
- Submit solutions online (WueCampus).
- Please use LaTeX (template on WueCampus); don't use ChatGPT!
- Discussions and solutions...

Books



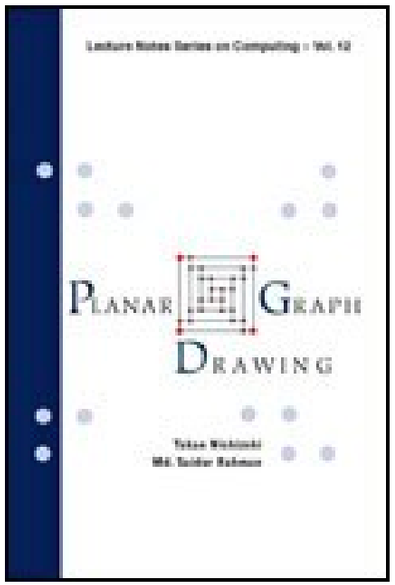
G. Di Battista, P. Eades, R. Tamassia, I. Tollis:
Graph Drawing: Algorithms for the Visualization of Graphs
Prentice Hall, 1998

[GD]



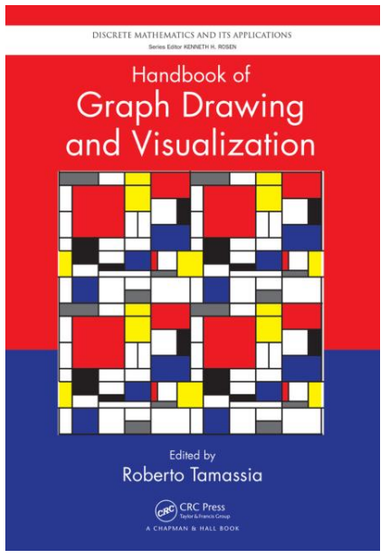
[DG]

M. Kaufmann, D. Wagner:
Drawing Graphs: Methods and Models
Springer, 2001



T. Nishizeki, Md. S. Rahman:
Planar Graph Drawing
World Scientific, 2004

[PGD]



[HGDV]

R. Tamassia:
Handbook of Graph Drawing and Visualization
CRC Press, 2013
`http://cs.brown.edu/people/rtamassi/gdhandbook/`

What Is this Course About?

Learning objectives

- Overview of graph visualization
- Improved knowledge of modeling and solving problems via graph algorithms

Visualization problem:

- Given a graph G, visualize it with a drawing Γ

Here:

- Reducing the visualization problem to its **algorithmic core**

graph class $\Rightarrow$ layout style $\Rightarrow$ algorithm $\Rightarrow$ analysis

- modeling
- data structures
- divide & conquer, incremental
- combinatorial optimization (flows, ILPs)
- force-based algorithm
- proofs

What Is this Course About?

Topics

- Drawing Trees and Series-Parallel Graphs
- Force-Based Drawing Algorithms and Tutte Embedding
- Straight-Line Drawings of Planar Graphs
- Upwards Planar Drawings
- Orthogonal Grid Drawings
- Contact Representations
- Hierarchical Layouts of Directed Graphs
- Visibility Representations
- The Crossing Lemma
- Linear Layouts
- Beyond Planarity
- Octilinear Drawings for Metro Maps

Graphs and Their Representations

What is a graph?

- graph G
- vertex set $V(G) = \{v_1, v_2, \ldots, v_n\}$
- edge set $E(G) = \{e_1, e_2, \ldots, e_m\}$, where each edge is a pair of vertices.

Representation?

- Set notation

$$V(G) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$$
$$E(G) = \{\{v_1, v_2\}, \{v_1, v_8\}, \{v_2, v_3\}, \{v_3, v_5\}, \{v_3, v_9\}, \{v_3, v_{10}\}, \{v_4, v_5\}, \{v_4, v_6\}, \{v_4, v_9\}, \{v_5, v_8\}, \{v_6, v_8\}, \{v_6, v_9\}, \{v_7, v_8\}, \{v_7, v_9\}, \{v_8, v_{10}\}, \{v_9, v_{10}\}\}$$

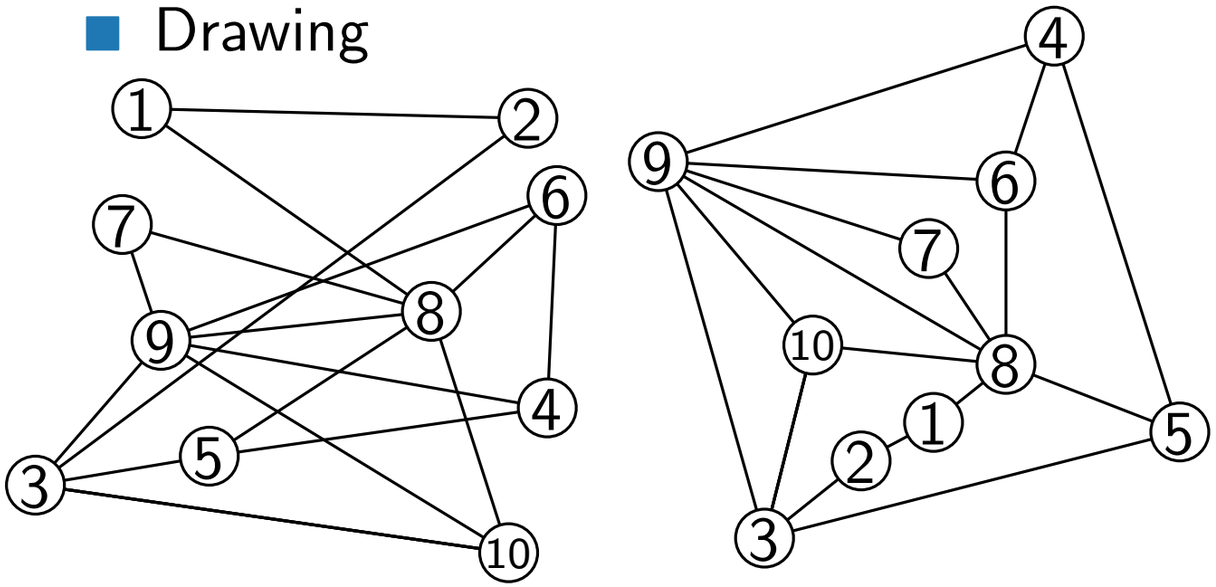
- Adjacency list

v_1: v_2, v_8
v_2: v_1, v_3
v_3: v_2, v_5, v_9, v_{10}
v_4: v_5, v_6, v_9
v_5: v_3, v_4, v_8
v_6: v_4, v_8, v_9
v_7: v_8, v_9
v_8: $v_1, v_5, v_6, v_7, v_9, v_{10}$
v_9: $v_3, v_4, v_6, v_7, v_8, v_{10}$
v_{10}: v_3, v_8, v_9

- Adjacency matrix

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

- Drawing



Why to Draw Graphs?

Graphs are a mathematical representation of
real physical and abstract networks.

Physical networks

- Metro systems
- Road networks
- Power grids
- Telecommunication networks
- Integrated circuits
- ...

Abstract networks

- Social networks
- Communication networks
- Phylogenetic networks
- Metabolic networks
- Class/Object Relation Digraphs (UML)
- ...

Why to Draw Graphs?

Graphs are a mathematical representation of
real physical and abstract networks.

- **People think visually** – complex graphs are hard to grasp without good visualizations!

Why to Draw Graphs?

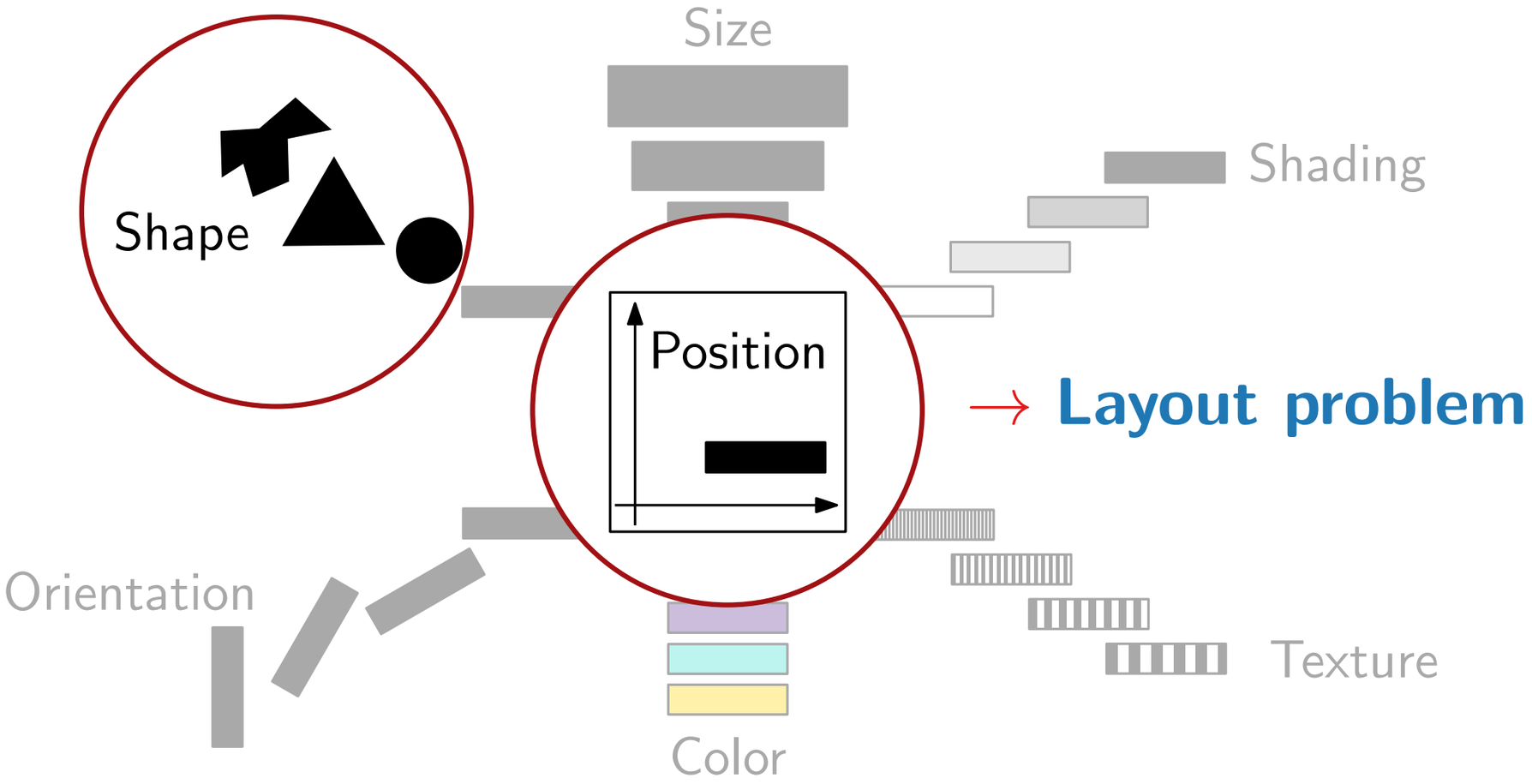
Graphs are a mathematical representation of
real physical and abstract networks.

- **People think visually** – complex graphs are hard to grasp without good visualizations!
- Visualizations help with the **communication** and **exploration** of networks.
- Some graphs are too big to draw them by hand.

We need algorithms that draw graphs automatically to make networks more accessible to humans.

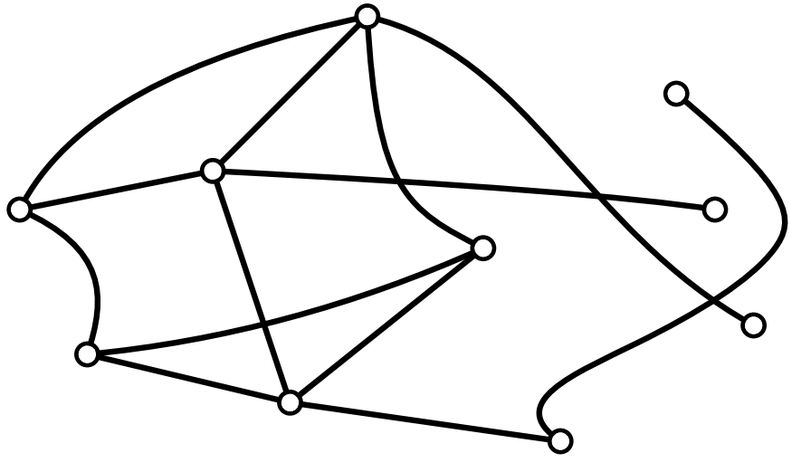
What Are We Interested In?

- Jacques Bertin defined *visualization variables* (1967)



The Layout Problem?

- *Here* restricted to the **standard representation**, so-called node–link diagrams.



Graph Visualization Problem

in: graph G

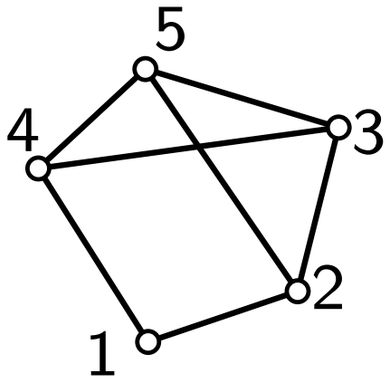
out: **nice** drawing Γ of G

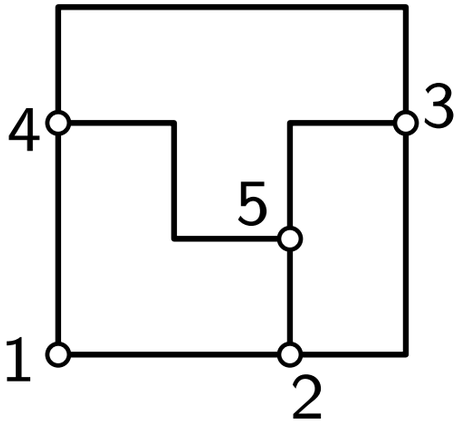
- $\Gamma\colon V(G) \to \mathbb{R}^2$, vertex $v \mapsto$ point $\Gamma(v)$
- $\Gamma\colon E(G) \to$ simple, open curves in $\mathbb{R}^2$
  $\{u, v\} \mapsto \Gamma(\{u, v\})$ with endpoints $\Gamma(u)$ and $\Gamma(v)$

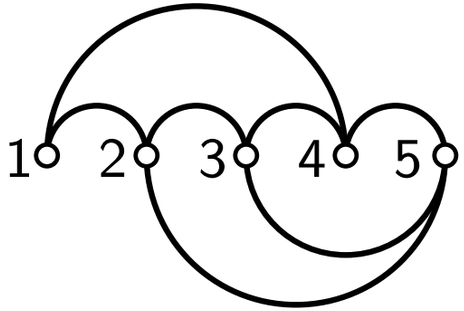
But what is a **nice** drawing?

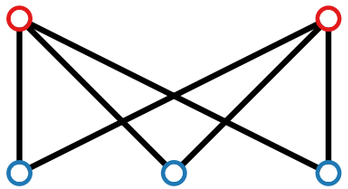
Requirements of a Graph Layout

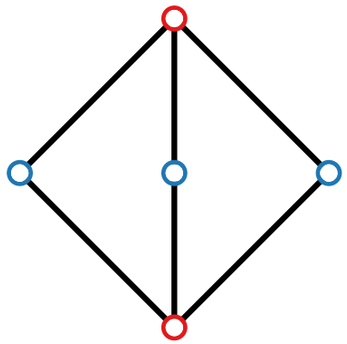
1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization





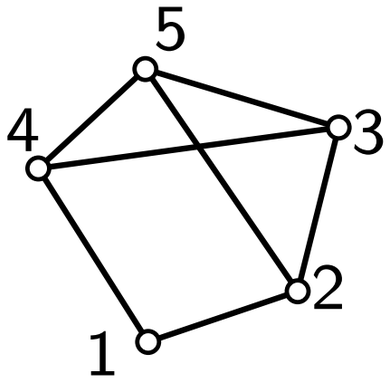


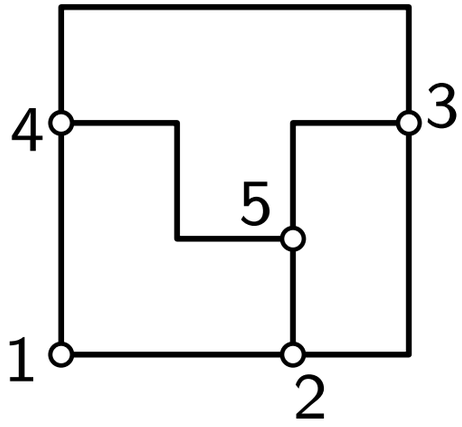


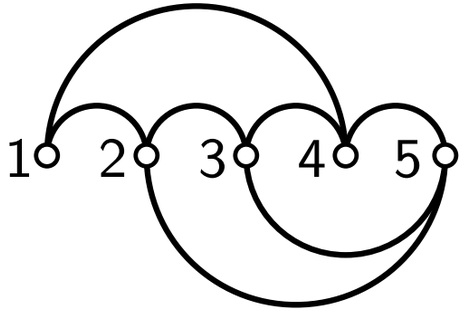


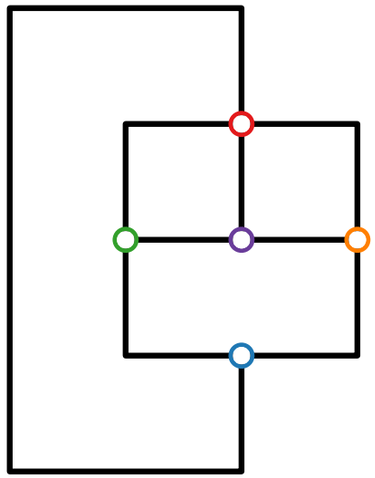
Requirements of a Graph Layout

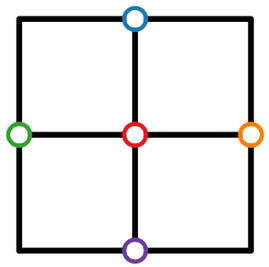
1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization





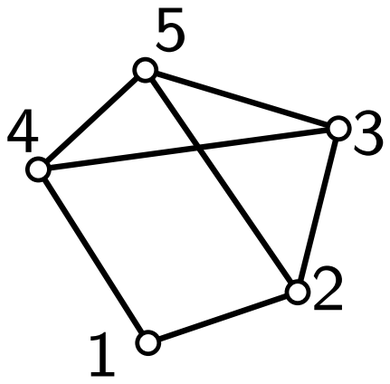


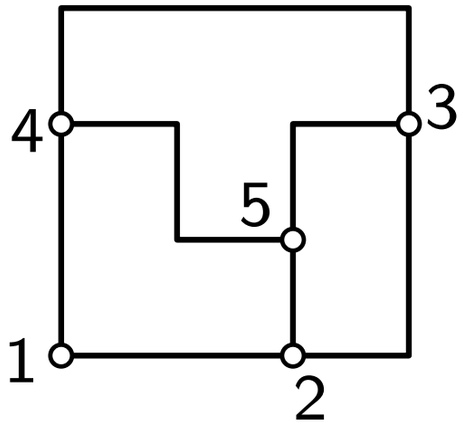


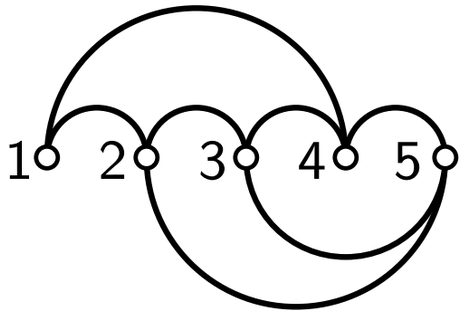


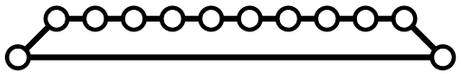
Requirements of a Graph Layout

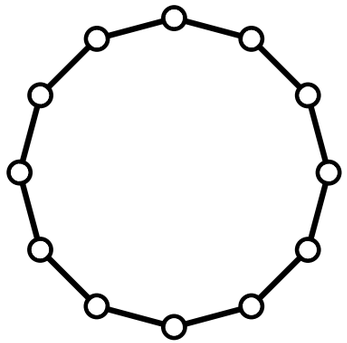
1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization
   - edge length uniformity





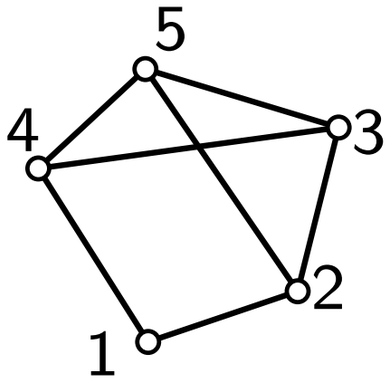


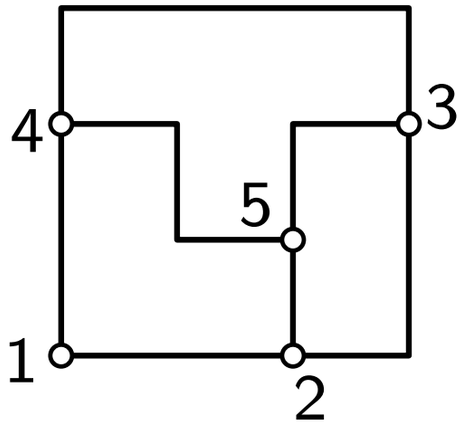


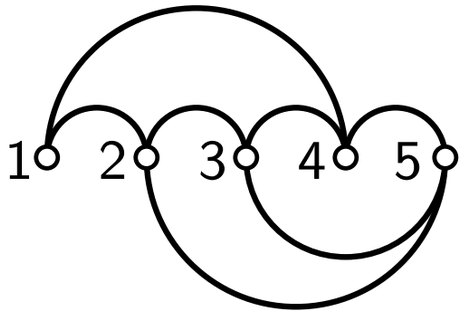


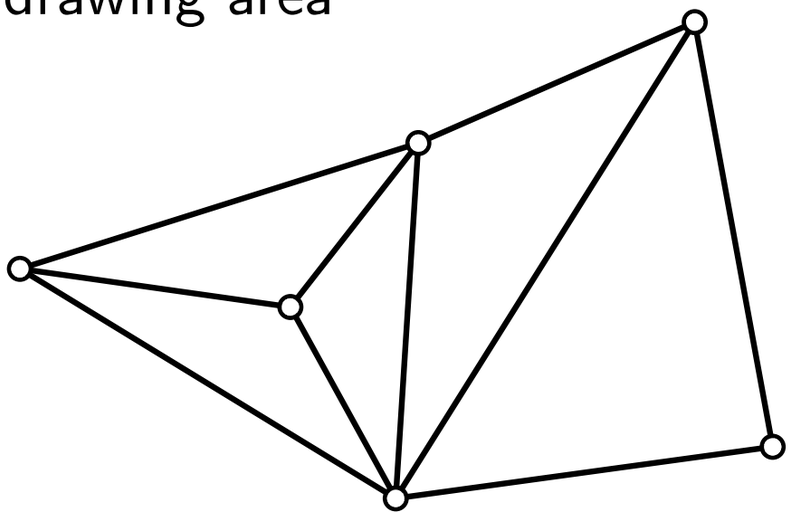
Requirements of a Graph Layout

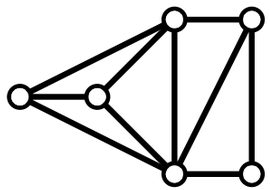
1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization
   - edge length uniformity
   - minimizing total edge length/drawing area





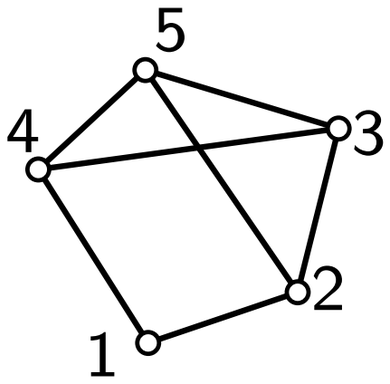


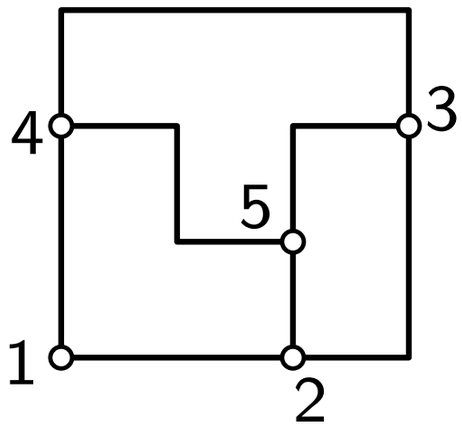


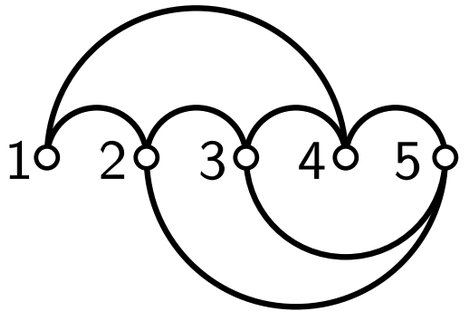


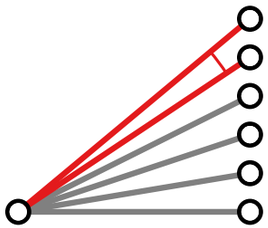
Requirements of a Graph Layout

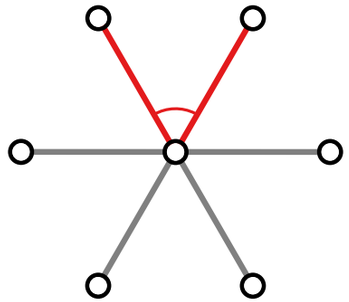
1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization
   - edge length uniformity
   - minimizing total edge length/drawing area
   - angular resolution





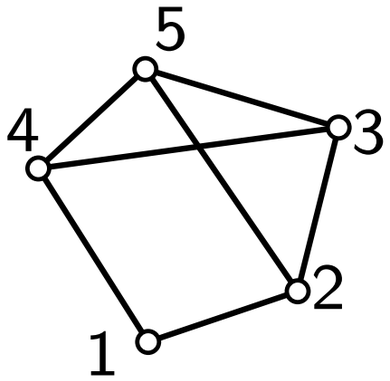


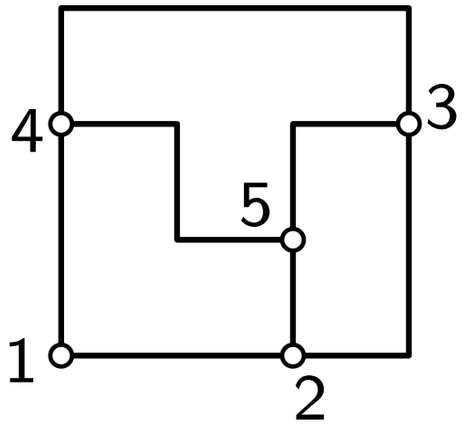


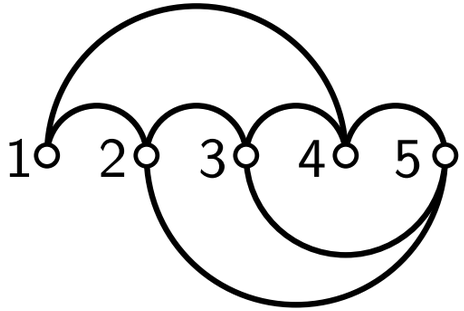


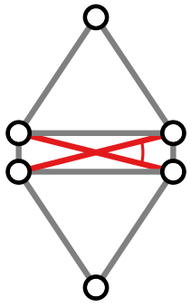
Requirements of a Graph Layout

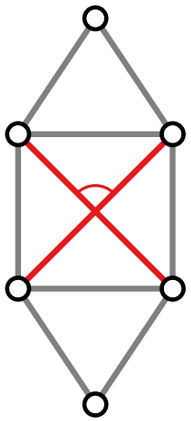
1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization
   - edge length uniformity
   - minimizing total edge length/drawing area
   - angular resolution





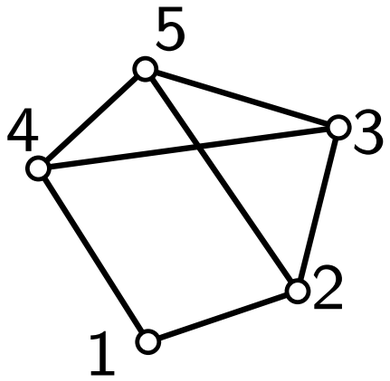


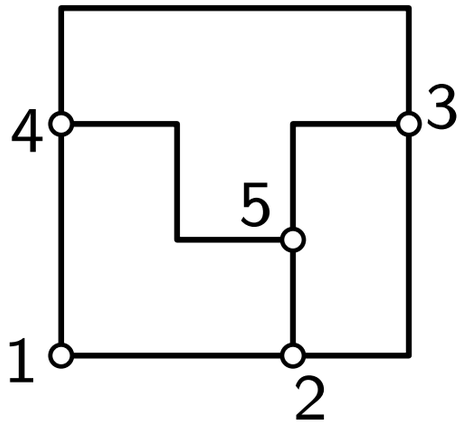


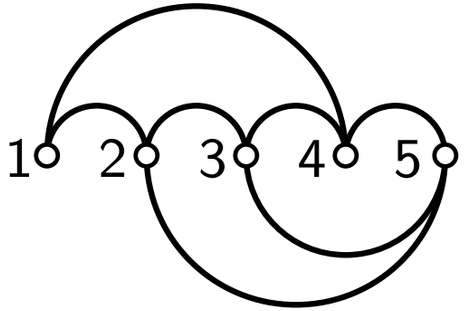


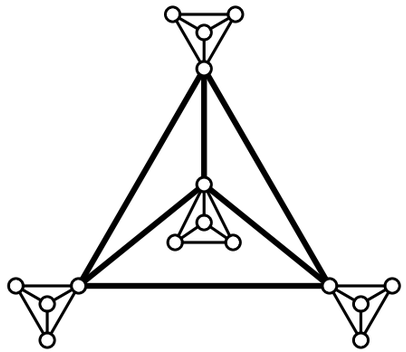
Requirements of a Graph Layout

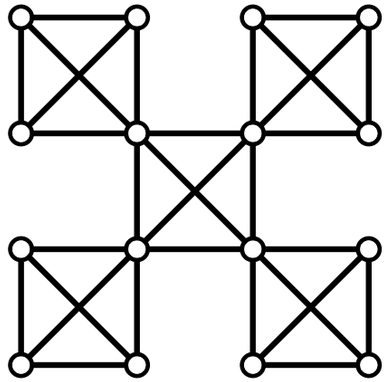
1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization
   - edge length uniformity
   - minimizing total edge length/drawing area
   - angular resolution
   - symmetry/structure



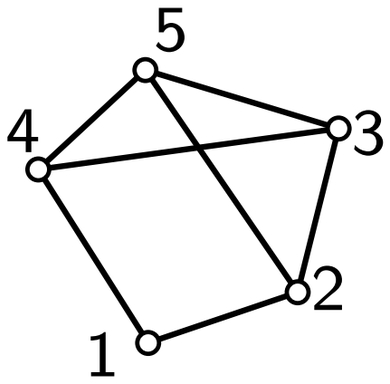


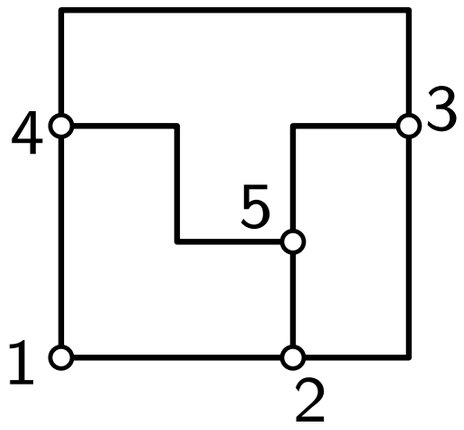


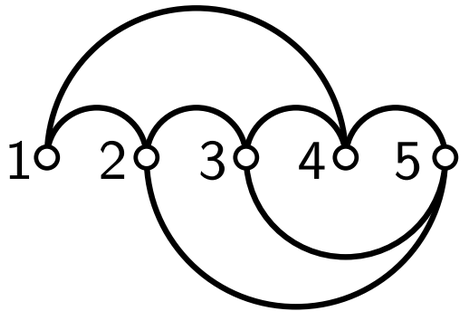




Requirements of a Graph Layout







1. Drawing conventions and requirements, e.g.,
   - straight edges with $\Gamma(uv) = \overline{\Gamma(u)\Gamma(v)}$
   - orthogonal edges (with bends)
   - grid drawings
   - without crossing
2. Aesthetics to be optimized, e.g.
   - crossing/bend minimization
   - edge length uniformity
   - minimizing total edge length/drawing area
   - angular resolution
   - symmetry/structure
3. Local Constraints, e.g.
   - restrictions on neighboring vertices (e.g., “upward”).
   - restrictions on groups of vertices/edges (e.g., “clustered”).

$\rightarrow$ such criteria are often inversely related

$\rightarrow$ lead to NP-hard optimization problems

The Layout Problem

Graph Visualization Problem (more general)

in: Graph G

out: Drawing Γ of G such that

- **drawing conventions** are met,
- **aesthetic criteria** are optimized, while
- some **additional constraints** are satisfied.

Graph Drawing Contest 2026

- We have seen that it is not always clear how a *nice* graph visualizations looks like.
- Therefore, there is a graph drawing contest at the Annual International Symposium on Graph Drawing and Network Visualization (GD).
- GD 2026: August 17–21, 2026, St. Catherines, Ontario, Canada
  https://mozart.diei.unipg.it/gdcontest/2026/
- Creative topic: Visualize data from the Eurovision song contest.
- Live Challenge (2025): *Minimizing the local crossing number*:
  - Given: a graph G,
  - task: assign the vertices of G to grid points of a square grid of restricted size,
  - objective: minimize (over all straight-line grid drawings) the maximum number of crossings over all edges of G.
- Interested in implementing a program for the live challenge? Can be done as a Praktikum!