Radial Coordinate Assignment for Level Graphs

Christian Bachmaier

University of Konstanz, Germany

Florian Fischer

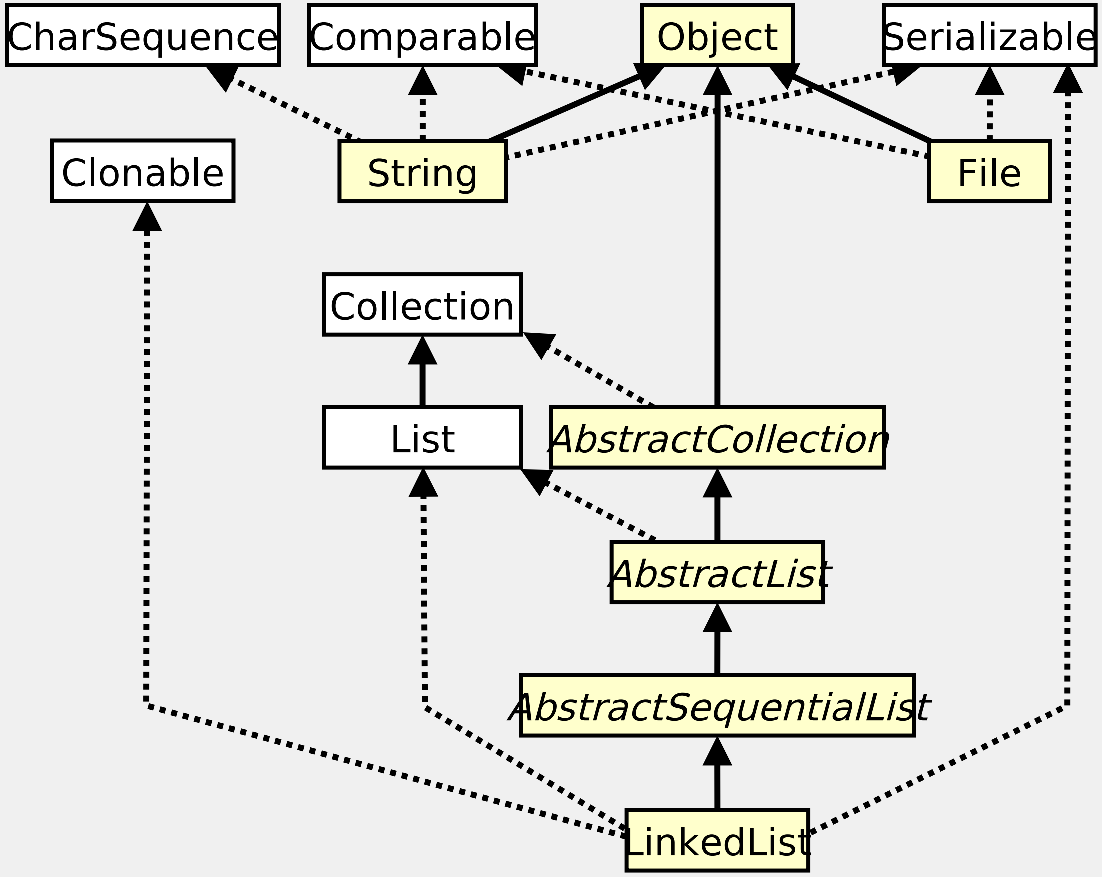
Projective Software GmbH, Germany

Michael Forster

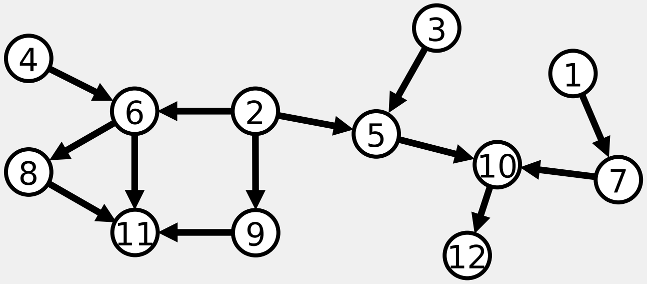
IMAGEN Program, National ICT Australia

UML-Diagram

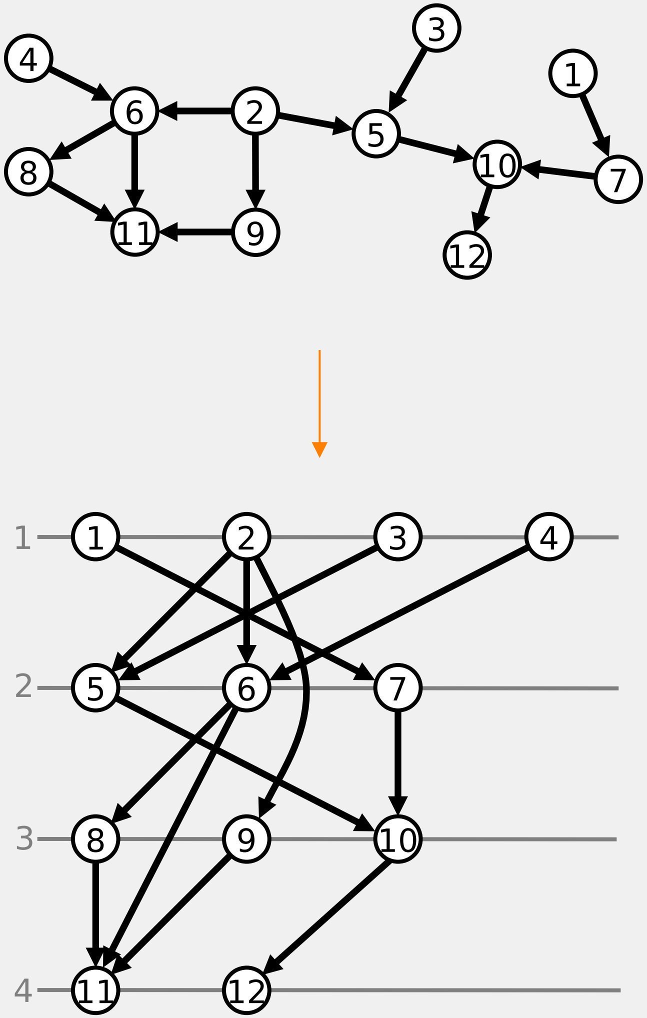
- Vertices
  - Classes
  - Interfaces
- Edges
  - Inheritance
  - Implementations



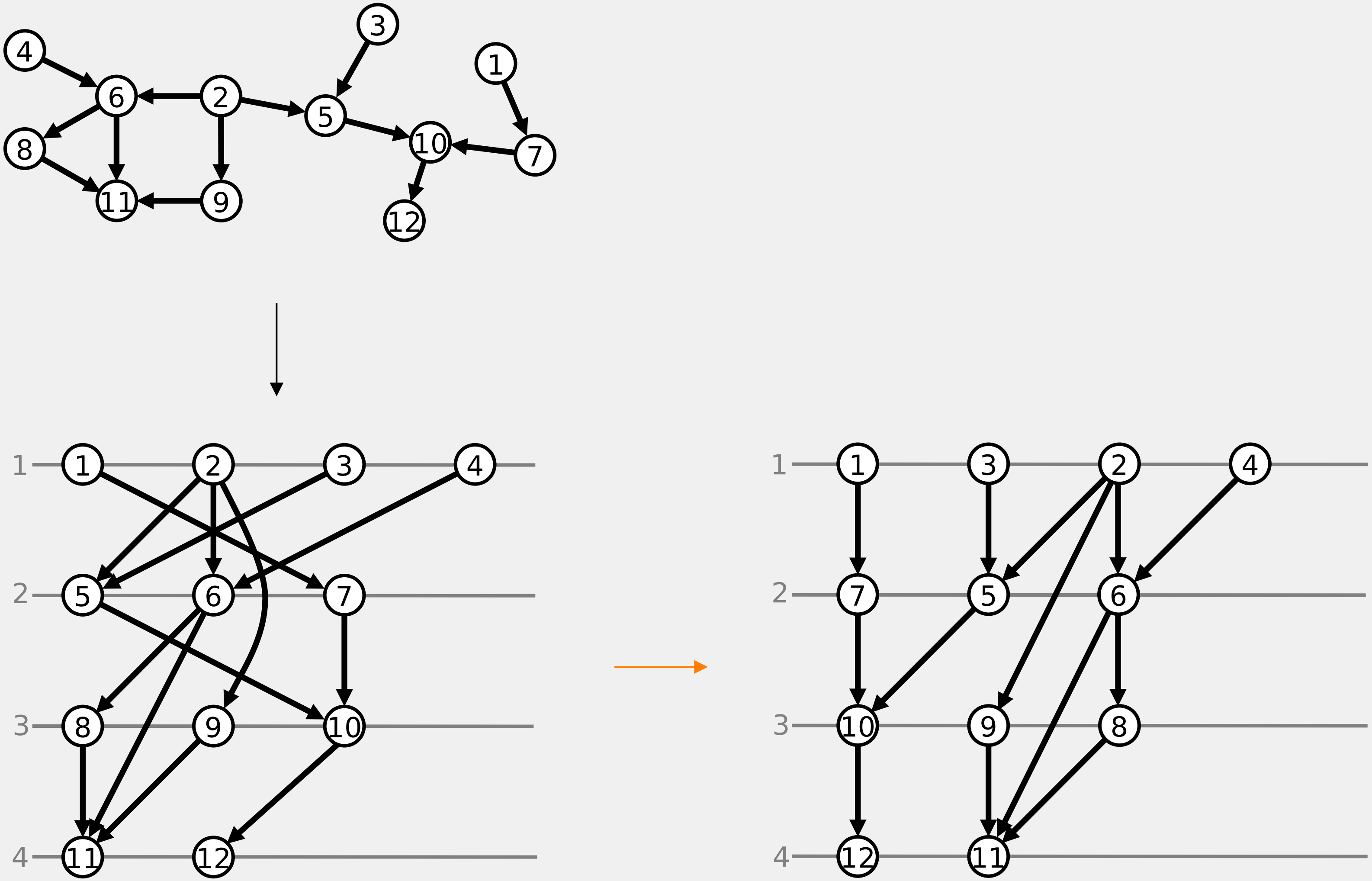
Sugiyama Framework



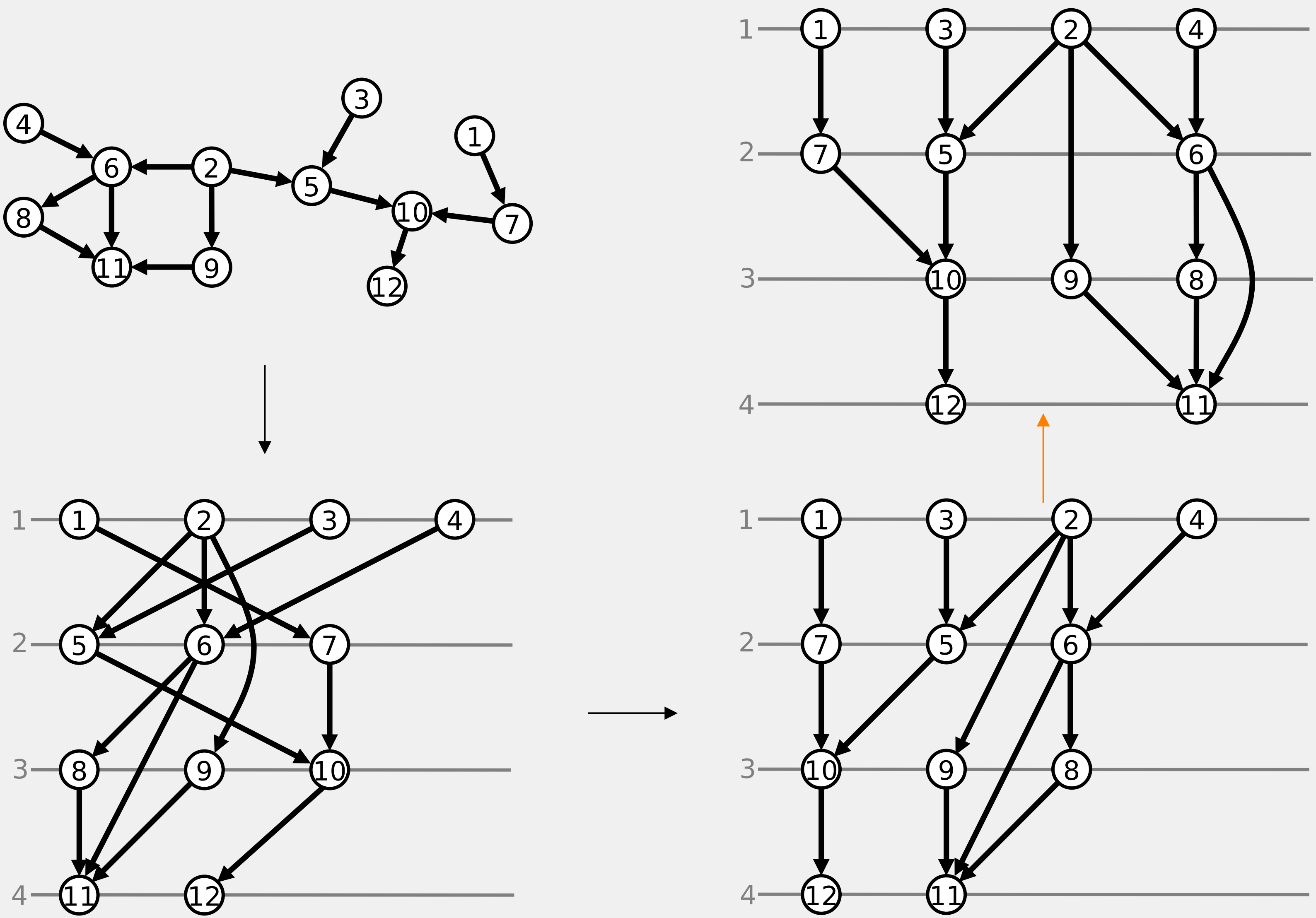
Sugiyama: Leveling



Sugiyama: Crossing Reduction

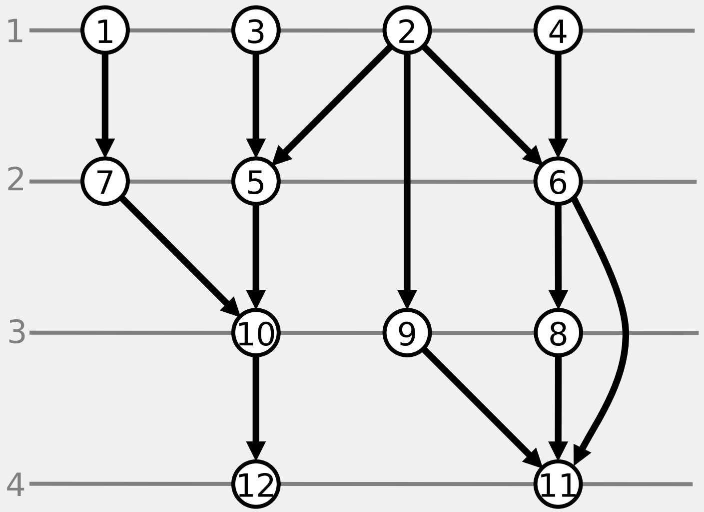


Sugiyama: Coordinate Assignment

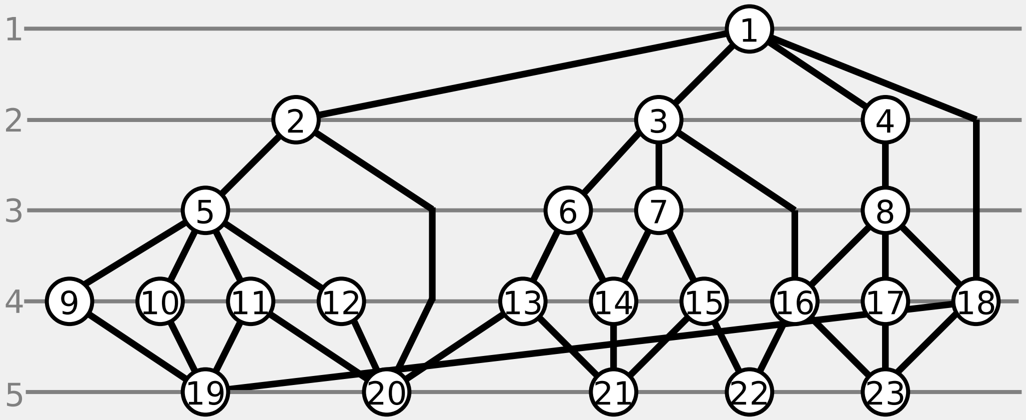


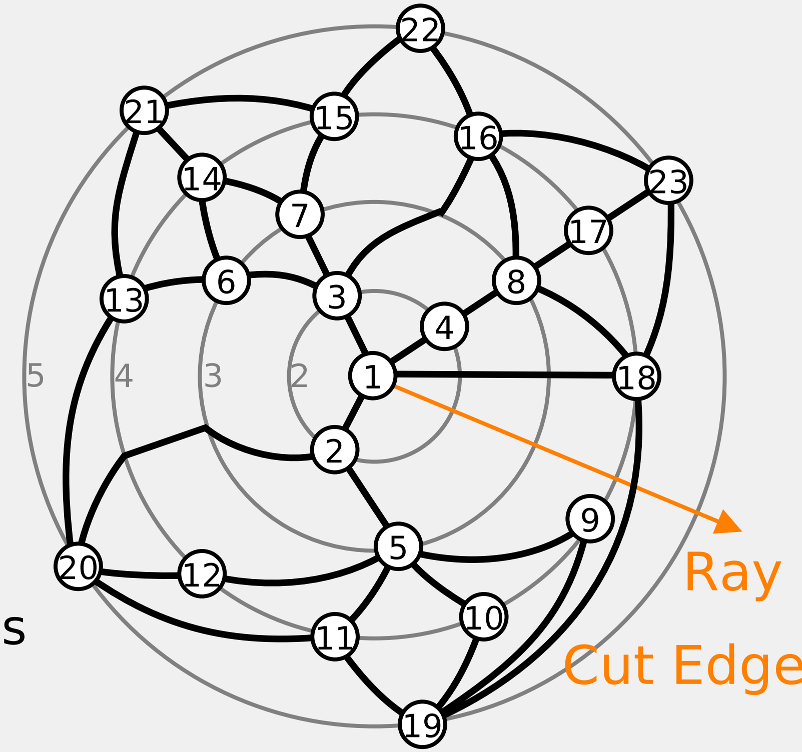
Coordinate Assignment

- Assigning x-coordinates
- Common esthetic criteria
  - Small area
  - Good separation of vertices on the same level
  - Slope of edges
  - Balancing of edges incident to the same vertex
  - ...
  - Straightness of long edges



Radial Level Drawings





- Idea
  - Bend level lines to concentric circles
  - Draw edges as spirals
- Advantages of radial level lines
  - Fewer edge crossings (experimentally 30%)
  - More planar level graphs
  - Good resolution for graphs with more vertices on higher levels

Example Applications

- Social networks
  - Vertices are actors
  - Edges are relations
  - Importance expressed by closeness to concentric center
    - Structural centrality measure
    - Mapping to geometric centrality
- Web structures
- Centralized network views
- Citation/recommendation networks

Overview

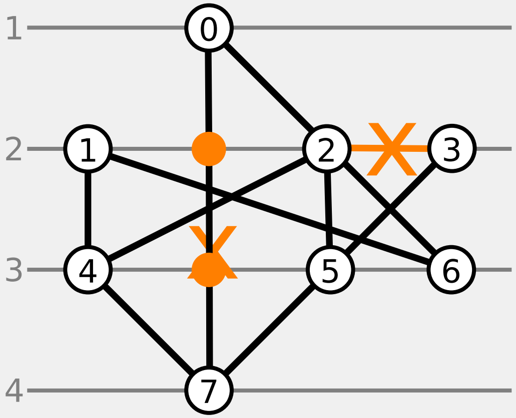
- Motivation
- Preliminaries
- Horizontal coordinate assignment
- Radial coordinate assignment
  - Differences to horizontal case
  - Algorithm

Preliminaries

Definition, Previous Work

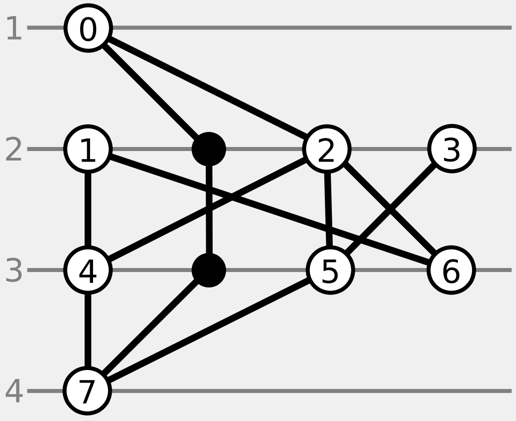
Level Graph

- k-level graph
  - $G = (V_1 \cup V_2 \cup \ldots \cup V_k, E)$
  - Vertex partitioning into k disjoint levels
  - No horizontal edges
- G is proper
  - Edges between adjacent levels
- Edge routing must be known
  - Consider only proper graphs
  - Dummy vertices splitting long edges
  - Segments
    - Inner segment ⇔ between two dummies
    - Outer segment ⇔ incident to one dummy



Level Embedding

- Vertex ordering on the levels given
  - From crossing reduction
  - From level planarity test
  - Fix



Previous Work

- Several algorithms for horizontal coordinate assignment
  - [Buchheim, Jünger, Leipert 00]
  - [Eades, Lin, Tamassia 96]
  - [Eades, Feng, Lin, Nagamochi 05]
  - [Eades, Sugiyama 90]
  - [Sugiyama, Tagawa, Toda 81]
  - ...
- Most interesting [Brandes, Köpf 01]
  - Max. two bends per edge
    - Dummy vertices of an edge aligned vertically
  - Good results for other esthetics
  - O(N) time

Horizontal

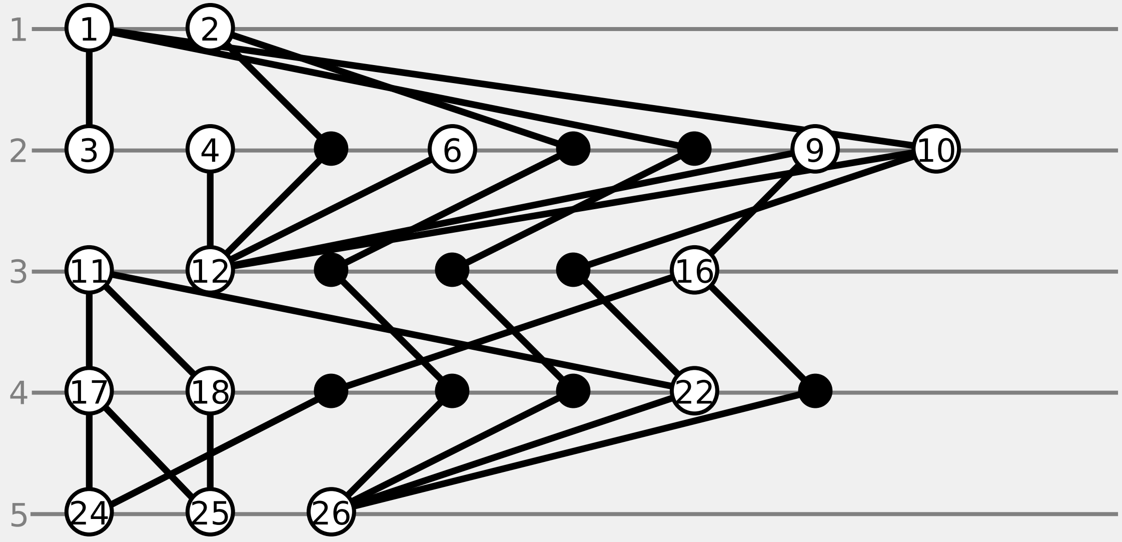
[Brandes, Köpf 01]

Concepts

- Vertical alignment of dummy vertices in any case
  - Each inner segment vertical
- Conflicts
  - Two segments cannot be vertically aligned at the same time
    - Crossing
    - Common end vertex
- Type 1 conflict
  - Inner segment and non-inner segment
    - Solved in favor for the inner segment
- Type 2 conflict
  - Two inner segments
    - Input embedding must not contain type 2 conflicts
    - Many crossing reduction algorithms guarantee their absence

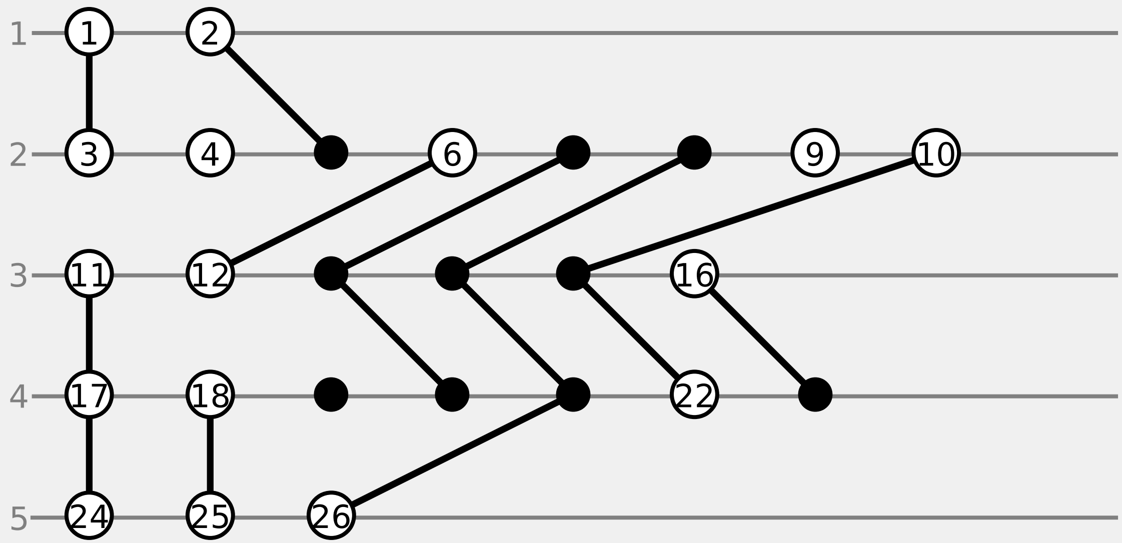
[Brandes, Köpf 01]

- Level-embedding
- Candidates



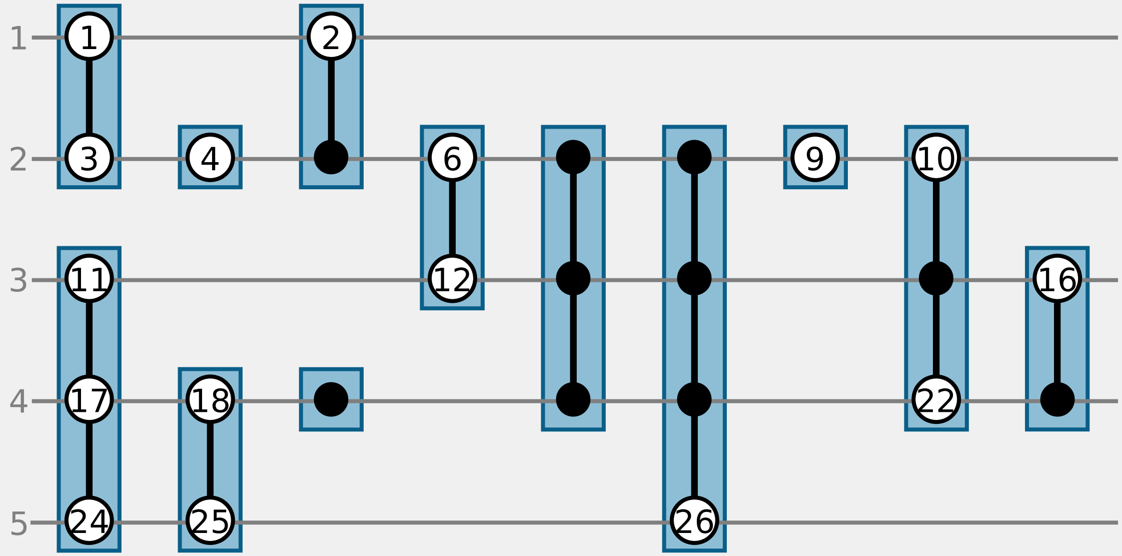
[Brandes, Köpf 01]

- Level-embedding
- Candidates



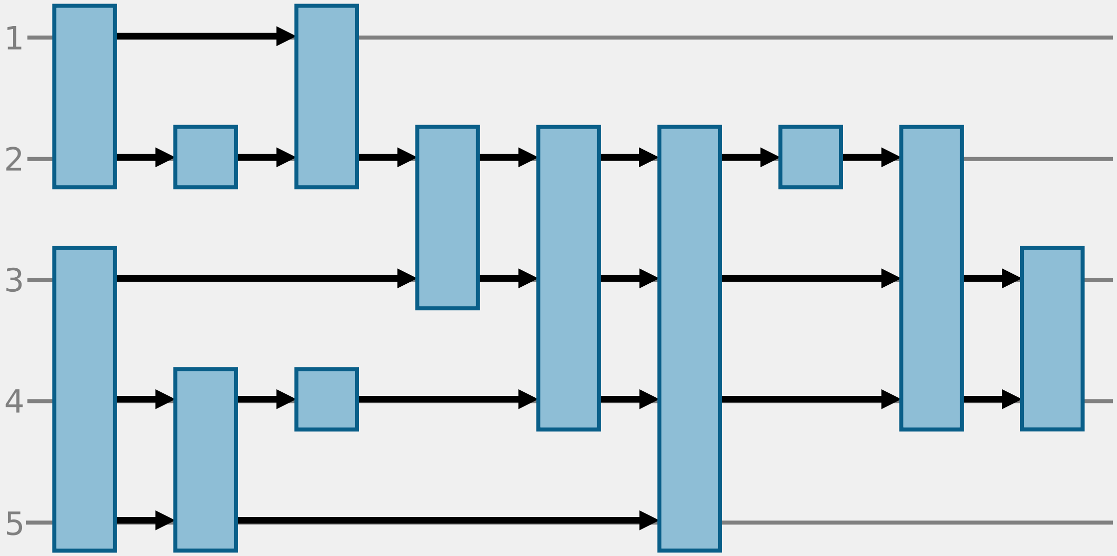
[Brandes, Köpf 01]

- Level-embedding
- Candidates
- Blocks



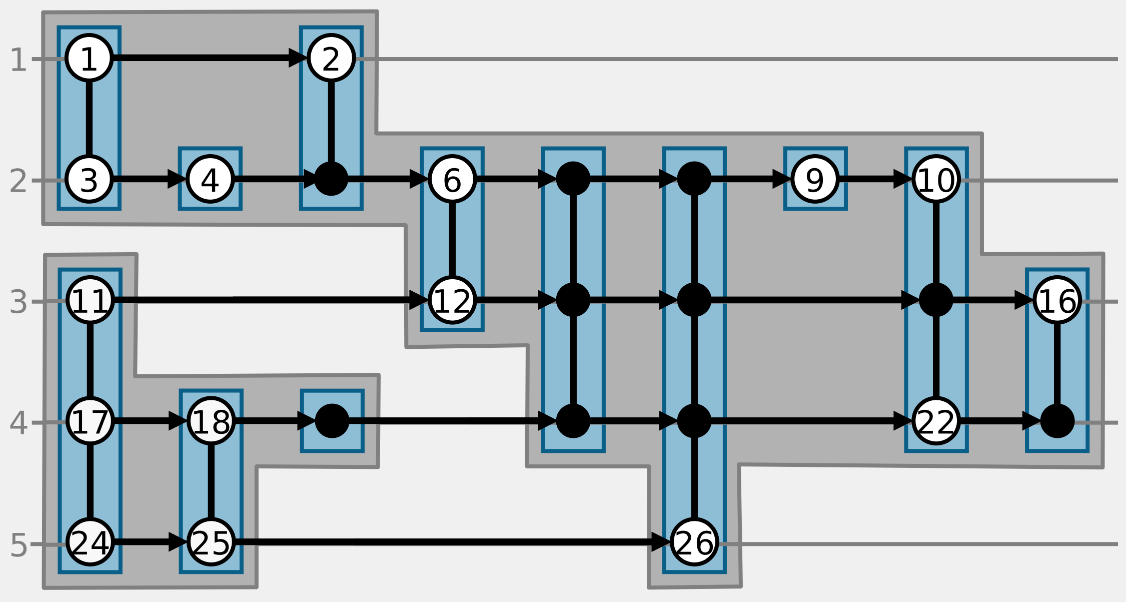
[Brandes, Köpf 01]

- Level-embedding
- Candidates
- Blocks
- Block graph



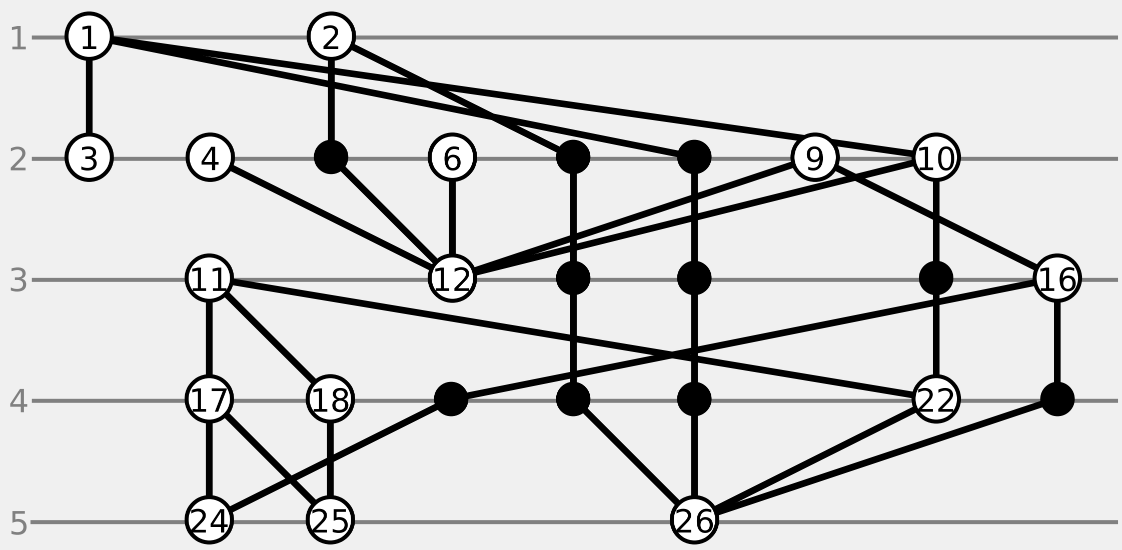
[Brandes, Köpf 01]

- Level-embedding
- Candidates
- Blocks
- Block graph
- Classes



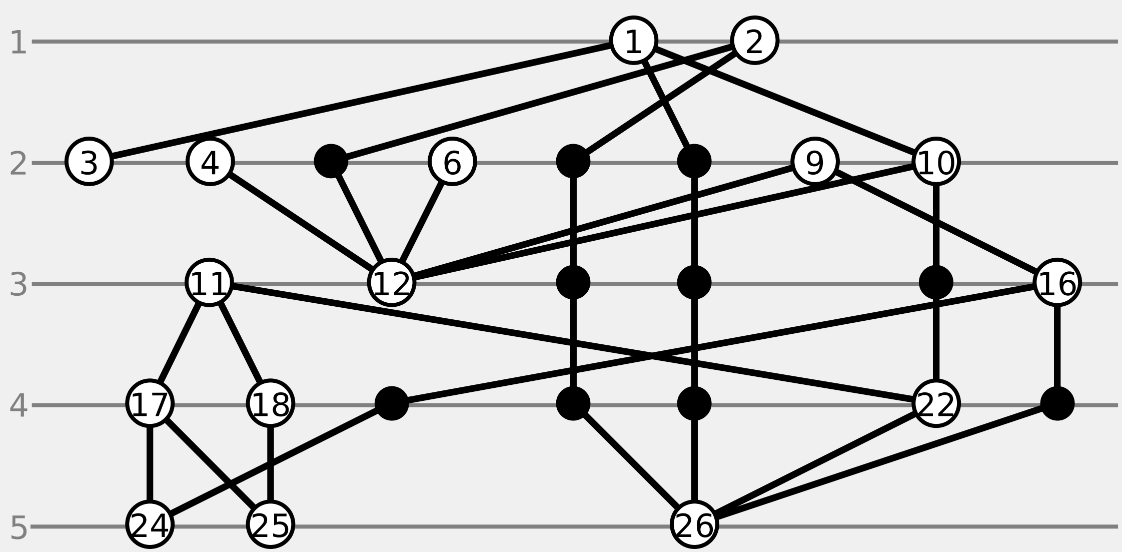
[Brandes, Köpf 01]

- Level-embedding
- Candidates
- Blocks
- Block graph
- Classes
- Layout
  - Four exemplars



[Brandes, Köpf 01]

- Level-embedding
- Candidates
- Blocks
- Block graph
- Classes
- Layout
  - Four exemplars
- Balanced layout

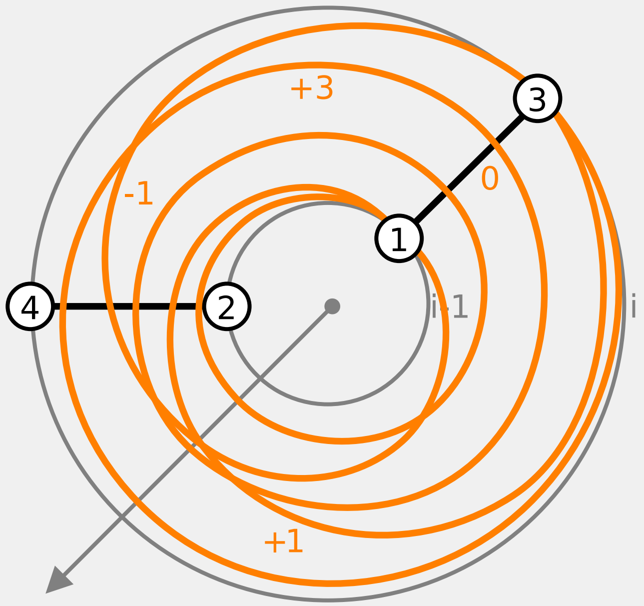


Radial

Concentrical Alignment, Max. two Bends per Edge

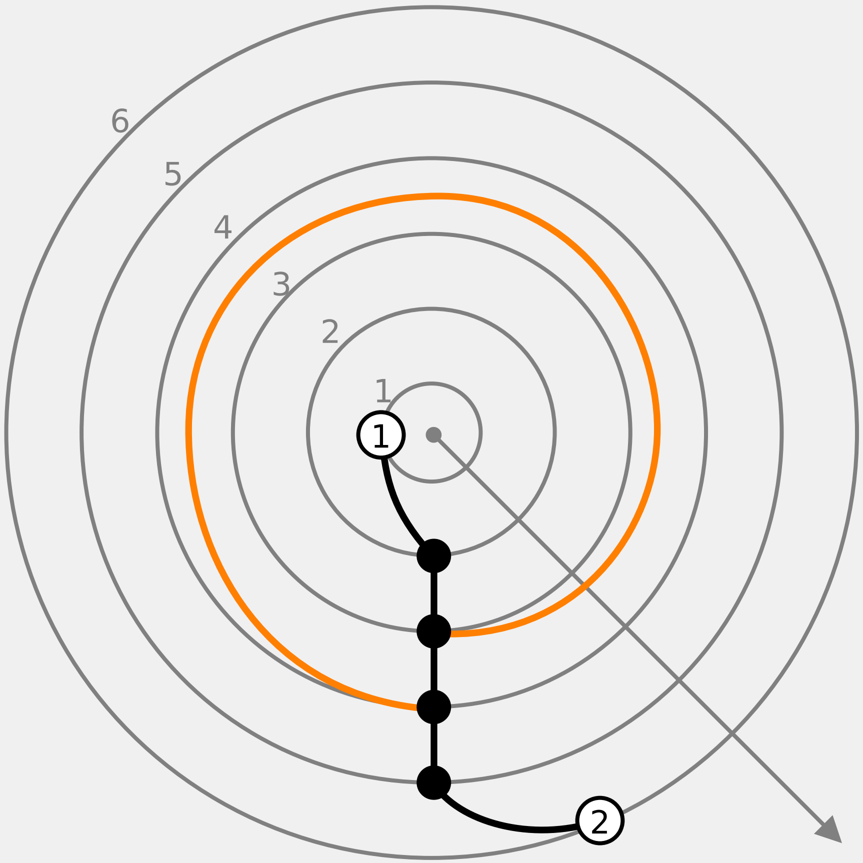
Edge Offset

- Cut edge/segment
  - Crossing the ray
- Offset of an edge/segment
  - Number of crossings between edge and ray
  - Number of windings around center
  - Sign determines direction
    - From inner to outer level
    - Positive ⇔ counter-clockwise
    - Negative ⇔ clockwise
- Offsets are given with the embedding as input



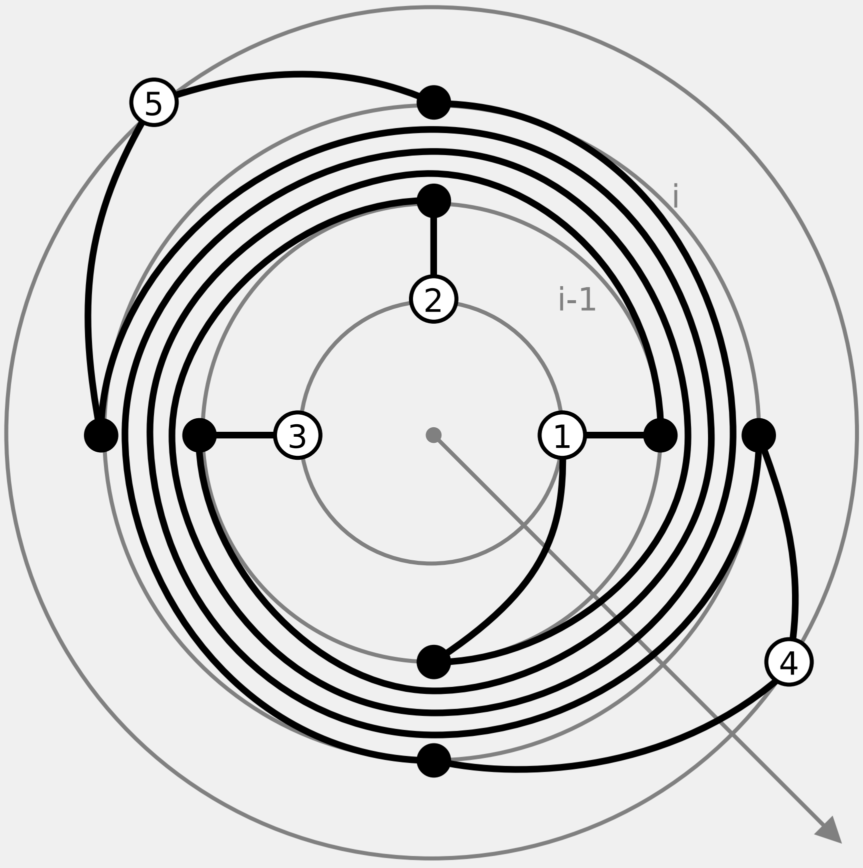
Long Edges With Inner Cut Segments

- Type 3 conflicts
  - Inner cut segments
- Elimination necessary
  - Otherwise more than 2 bends
  - Without introducing new crossings



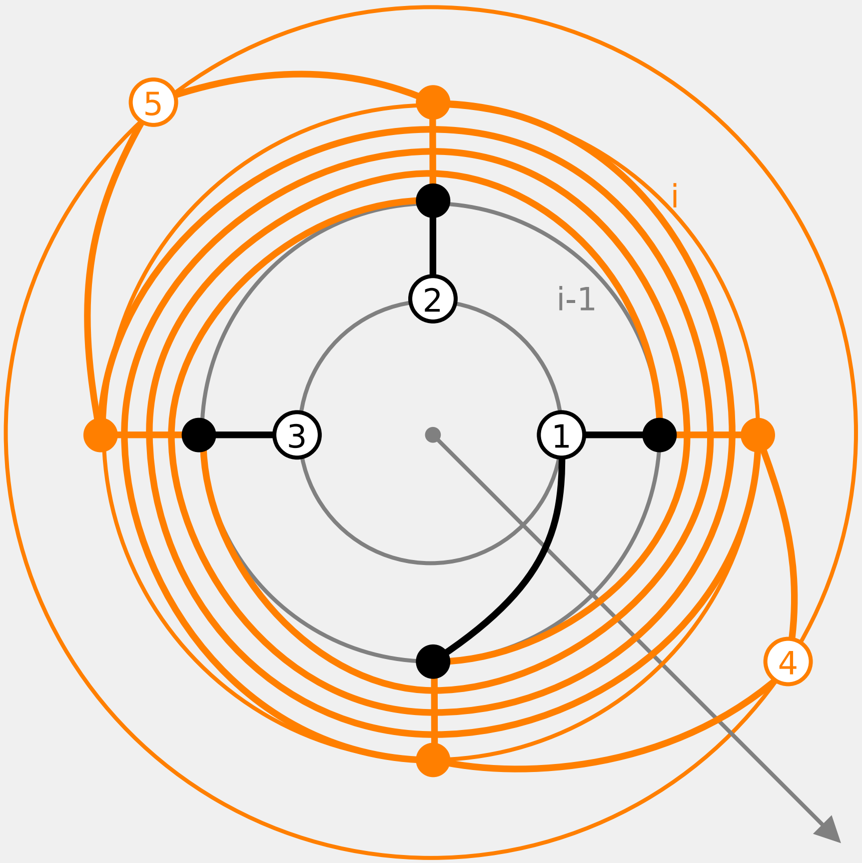
Unwinding

- Segments between medium levels have offset +1
- Offsets of inner segments entering a common level always differ at most by 1
  - Otherwise type 2 conflicts



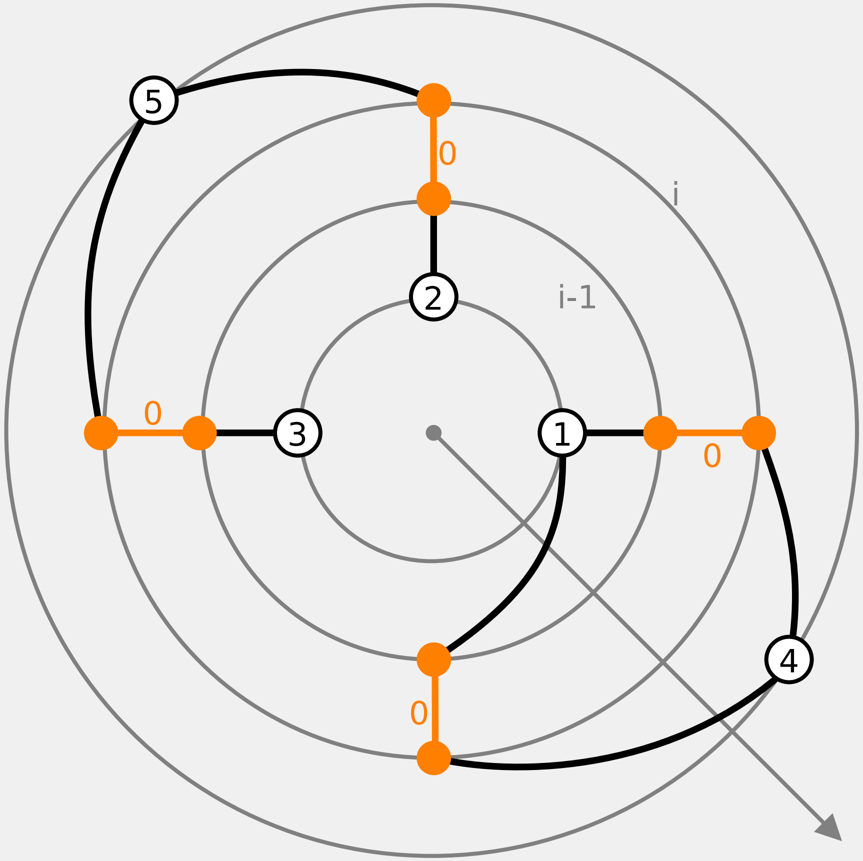
Unwinding

- Segments between medium levels have offset +1
- Offsets of inner segments entering a common level always differ at most by 1
  - Otherwise type 2 conflicts
- Rotate outer half of the graph
  - Minimum of inner offsets times
  - Multiples of 360°
  - Clockwise since positive



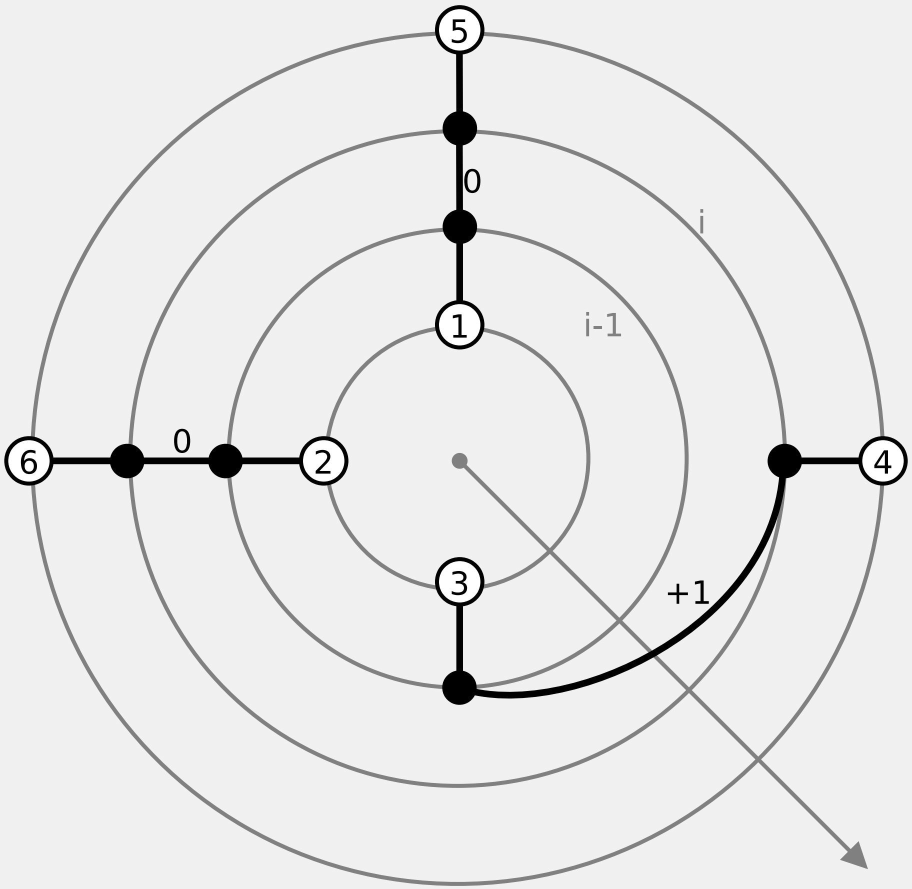
Unwinding

- Segments between medium levels have offset +1
- Offsets of inner segments entering a common level always differ at most by 1
  - Otherwise type 2 conflicts
- Rotate outer half of the graph
  - Minimum of inner offsets times
  - Multiples of 360°
  - Clockwise since positive
- Afterwards inner segments have offset 0 or +1
- For each level



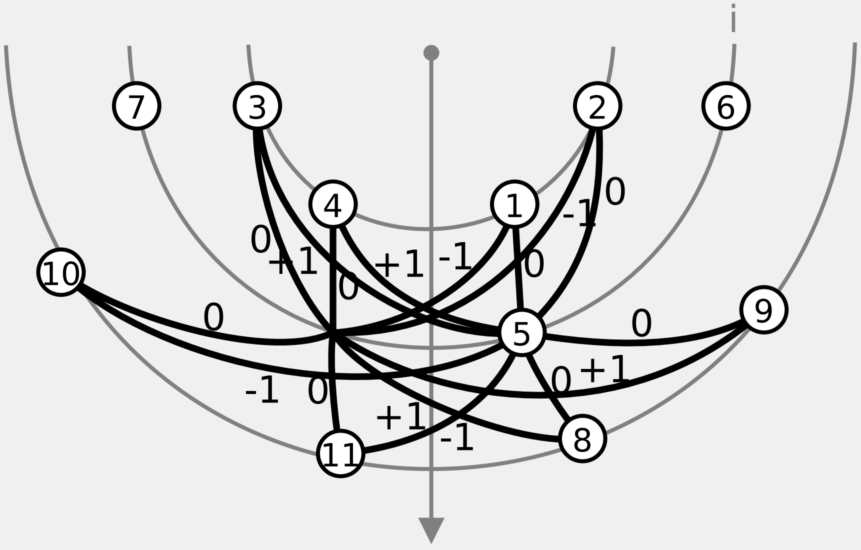
Problem

- Unwinding may leave inner offsets of +1
- Rotating by multiples of 360° does not solve the problem
- Only between dummies
  - At the end of inner level
  - At the beginning of outer level
- Elimination by rotation
  - Of single levels
  - Individual angles
  - Moving single vertices



Rotation

- Rotation of a level
- Clockwise
  - Offset updates of adjacent segments
- Counter-clockwise symmetrically
  - Here not needed
- No new crossings
- For each level outwards
- No more type 3 conflicts

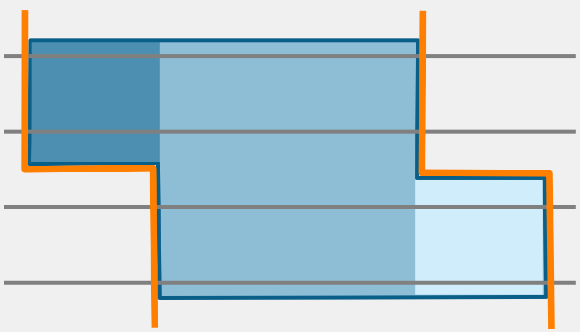


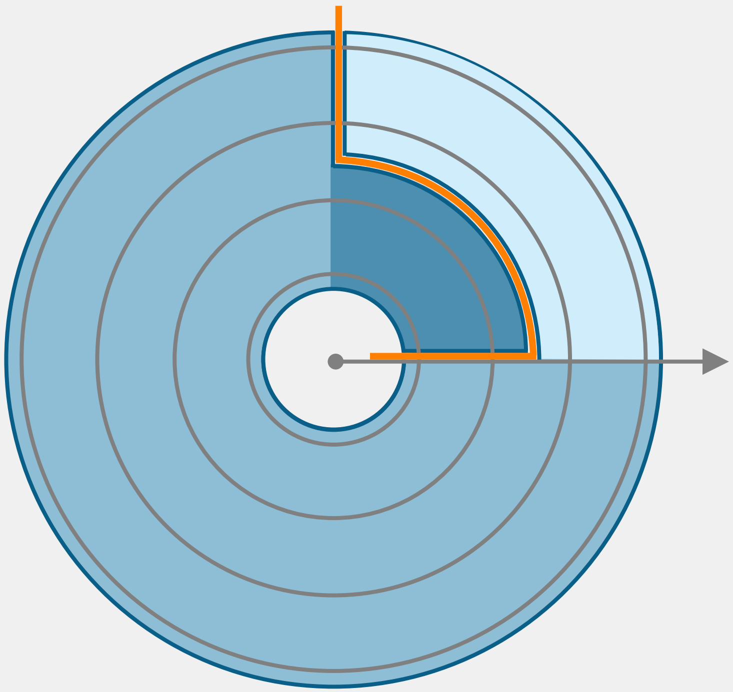
Algorithm

- Eliminate type 3 conflicts
  - Unwind
  - Rotate
- Split at the ray's position
- Apply [Brandes, Köpf 01]
- Transform coordinates of vertices to get a radial drawing
  - Reinterpret the "horizontal" coordinates as polar coordinates
  - Transform them back to Cartesian coordinates
  - $(x_r, y_r) = (y \cos(\frac{2\pi}{z} x), y \sin(\frac{2\pi}{z} x))$
  - $\frac{2\pi}{z}$ normalizes the length to the circumference of the level
  - z is the largest width of a single horizontal level

Overlap

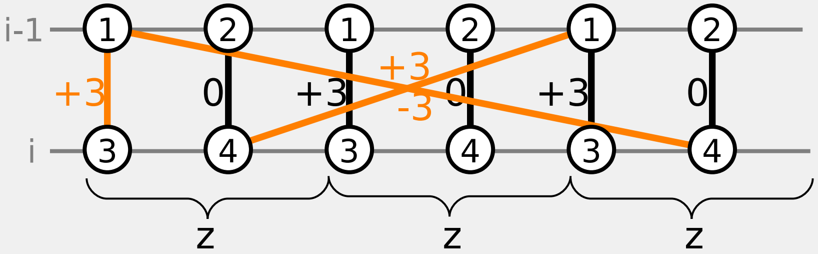
- Normalization in dependency to z realizes the necessary overlap of the left and right contour





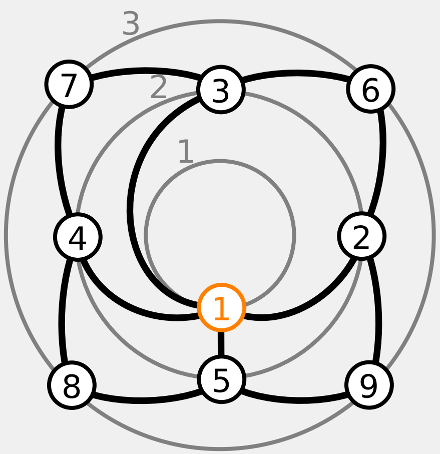
Drawing Edges

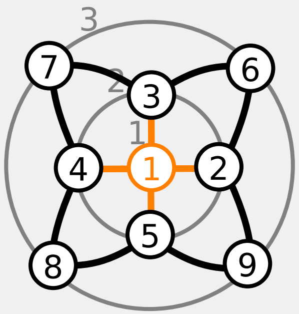
- Also transform each (interpolation) point of the edges
  - $(x, y) = (1 - t)(x_1, y_1) + t(x_2, y_2)$
  - Segment of a spiral
- For cut edges e
  - |offset(e)| many drawings side by side in a row
  - Clockwise direction
    - Draw e from the left to the right drawing
  - Counter-clockwise symmetric
  - $(x, y) = (1 - t)(x_1, y_1) + t(x_2 + z \cdot \text{offset}(e), y_2)$



Refinement

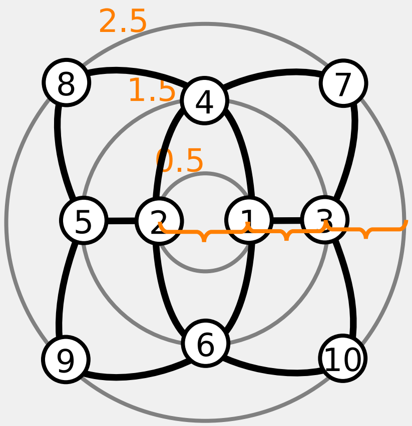
- Special case $|V_1| = 1$
  - Place vertex into concentric center
  - Draw adjacent edges straight line





Refinement

- Special case $|V_1| = 1$
  - Place vertex into concentric center
  - Draw adjacent edges straight line
- Harmonic picture [Eades 92]
  - Diameter of first level equals radial distance of level lines
  - Using 0.5, 1.5, 2.5, …, k – 0.5 as level numbers



Conclusion

Past and Future Work

Implementation

- Running time
  - O(N)
  - Competitive in practice
- Prototype
  - Java
  - Gravisto [B., Brandenburg, Forster, Raitner, Holleis 03]
  - N = 50.000 needs 50 sec. on 1.8 GHz PC

Moving Forward

- Vertices with arbitrary size
- Radial crossing reduction
  - Use additional freedom to reduce number of crossings
  - Avoid type 3 conflicts
    - Unwinding and rotation may raise the offsets of non-inner segments
  - Minimize absolute values of edge offsets

The End

Thanks for your attention!