Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

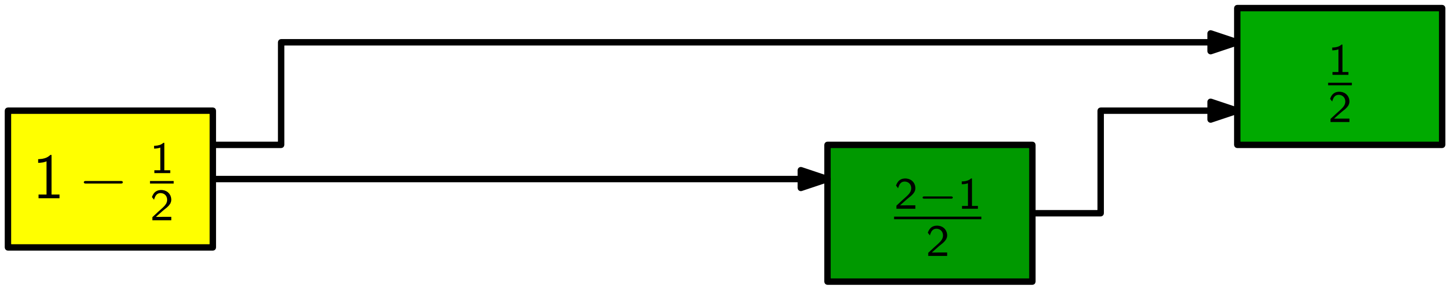
Drawing Graphs within Restricted Area

Martin Fink
Department of Computer Science
University of California, Santa Barbara

Joint work with Maximilian Aulbach,
Julian Schuhmann, and Alexander Wolff

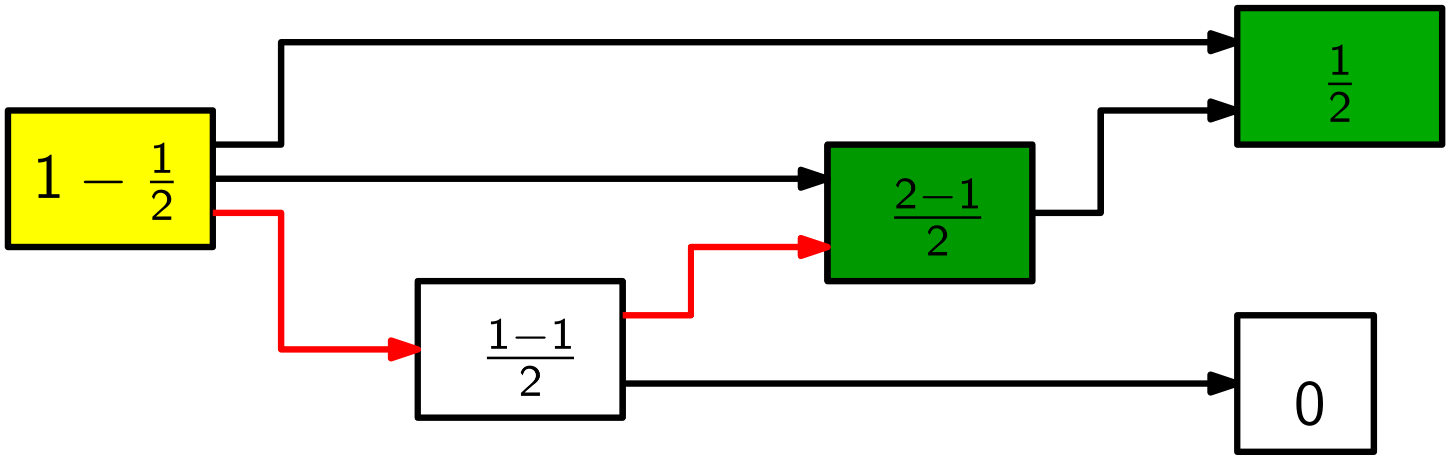
Motivation – Calculation Graphs

- directed graph of calculation steps



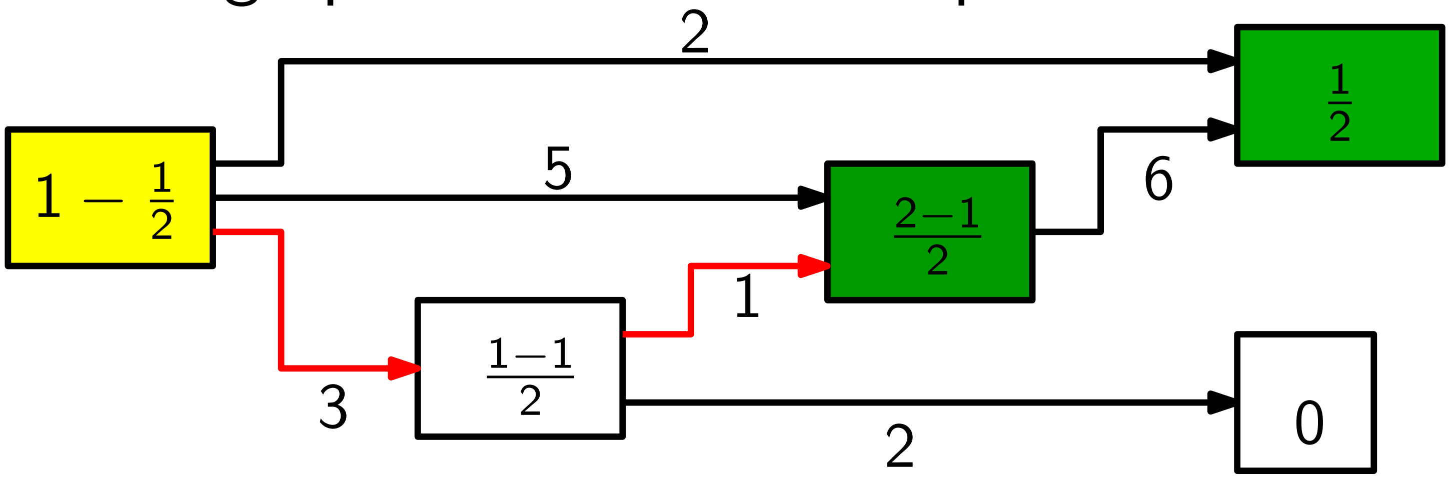
Motivation – Calculation Graphs

- directed graph of calculation steps



Motivation – Calculation Graphs

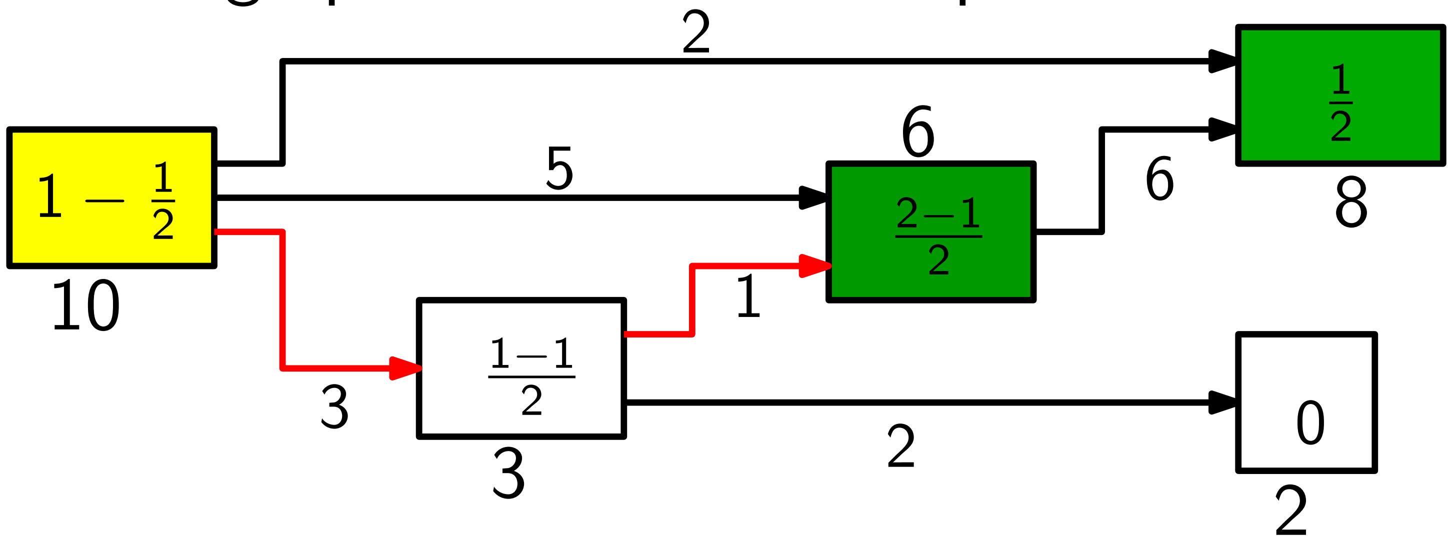
- directed graph of calculation steps



- source: studies with many students

Motivation – Calculation Graphs

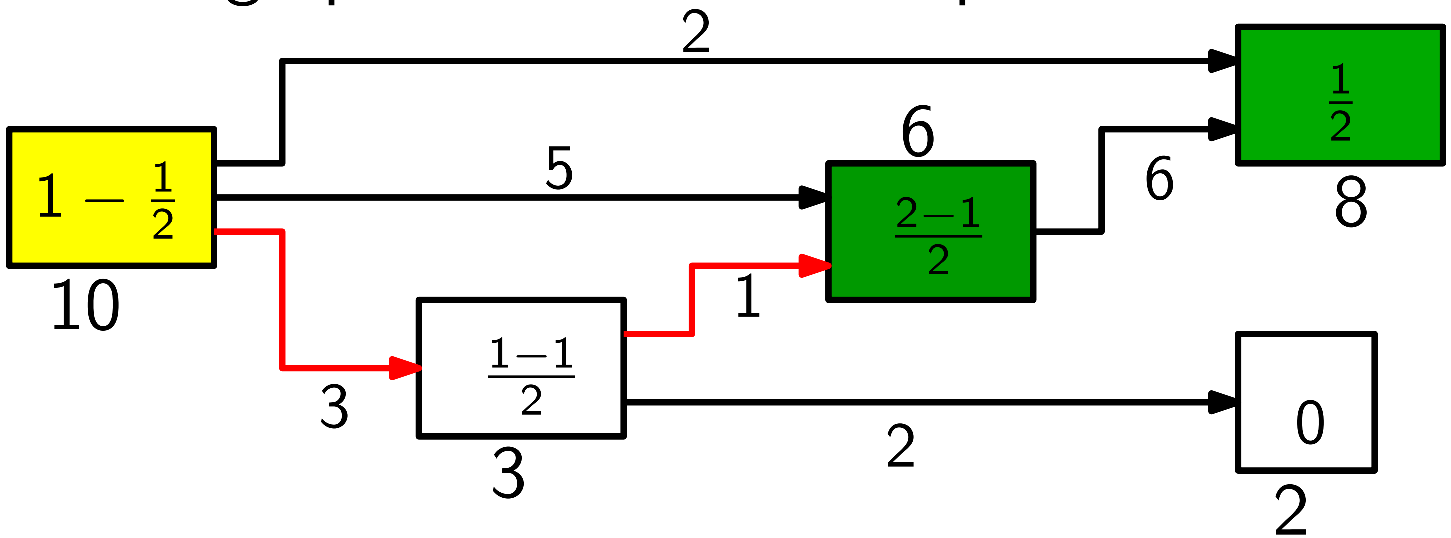
- directed graph of calculation steps



- source: studies with many students

Motivation – Calculation Graphs

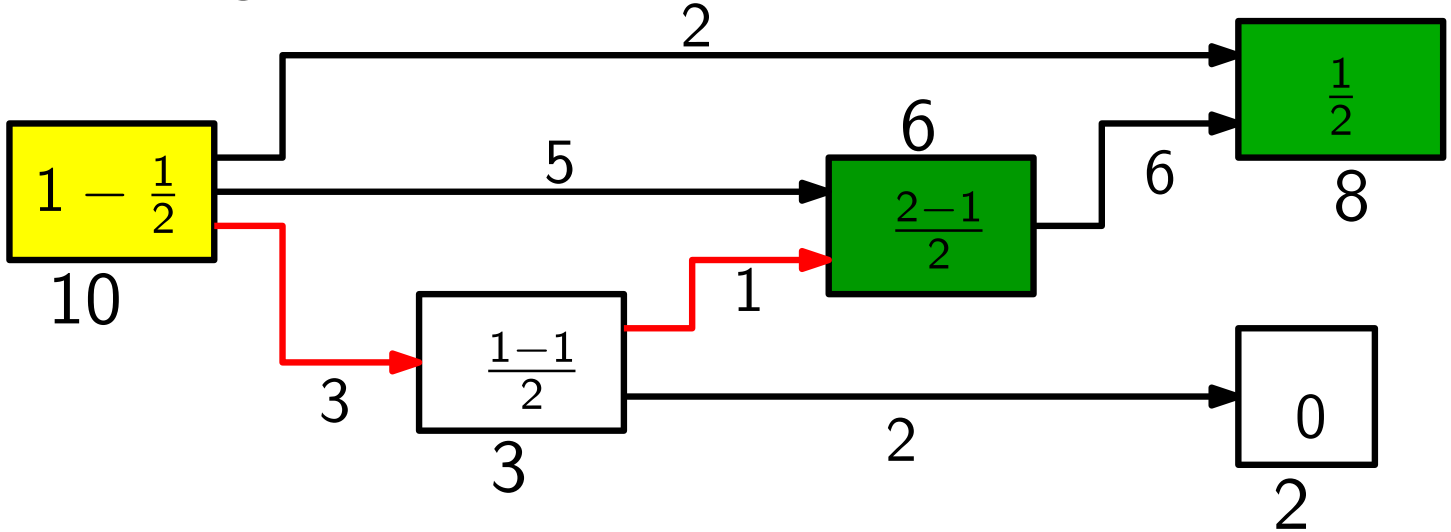
- directed graph of calculation steps



- source: studies with many students
- often very large (500 – 1000 vertices)

Motivation – Calculation Graphs

- directed graph of calculation steps

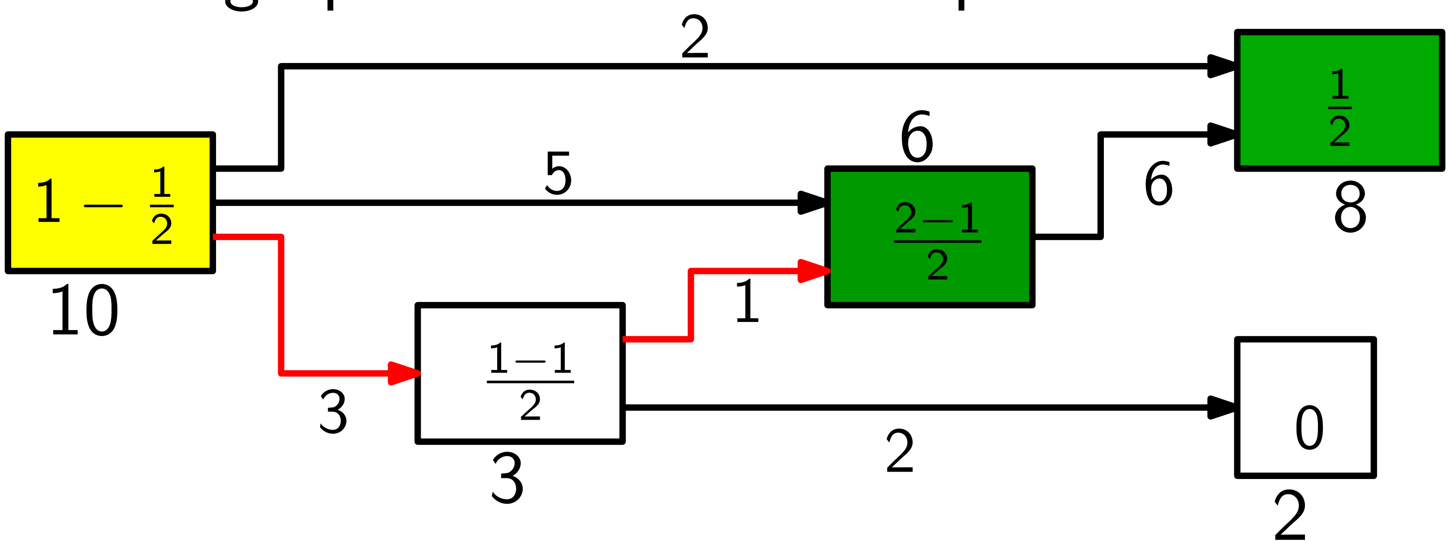


- source: studies with many students
- often very large (500 – 1000 vertices)

task: allow human expert to analyze for typical calculation steps and frequent mistakes

Motivation – Calculation Graphs

- directed graph of calculation steps



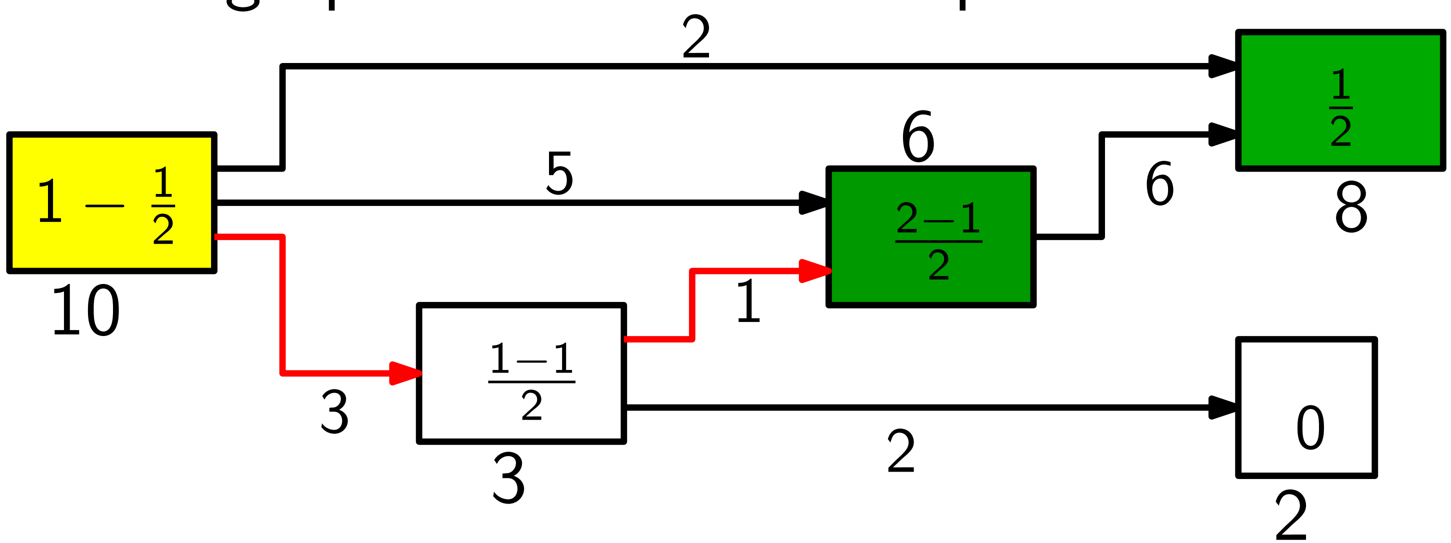
- source: studies with many students

– draw heavy subgraph in prescribed drawing area
– given vertex sizes

task: allow human expert to analyze for typical calculation steps and frequent mistakes

Motivation – Calculation Graphs

- directed graph of calculation steps



- source: studies with many students

– draw heavy subgraph in prescribed drawing area
– given vertex sizes

NP-hard

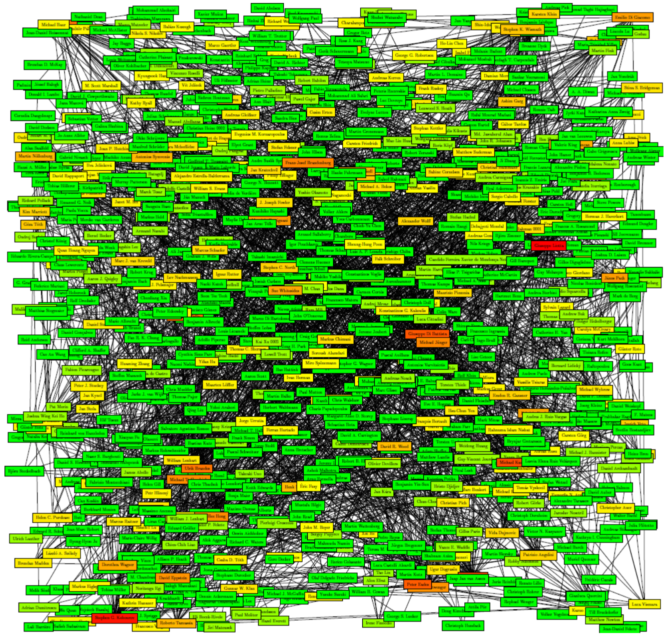
task: allow human exper e for typical calculation steps and frequent mistakes

Motivation cont.

- social networks

Motivation cont.

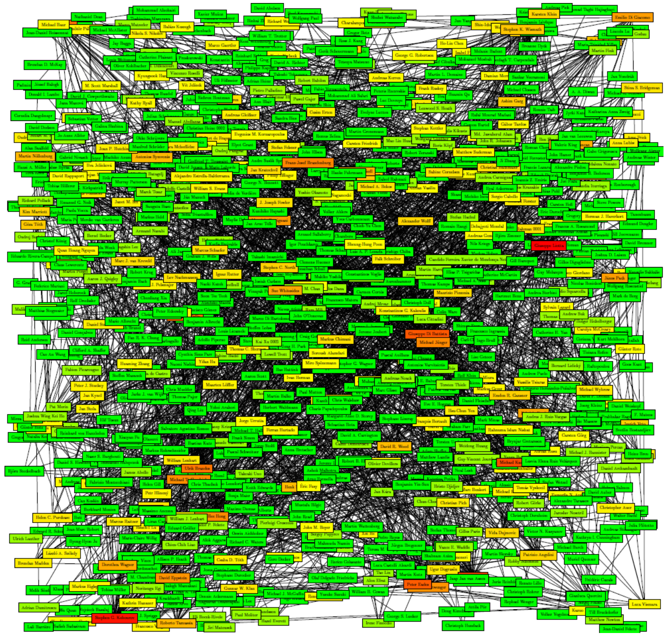
- social networks
- collaboration/coauthor graphs



Motivation cont.

- social networks
- collaboration/coauthor graphs

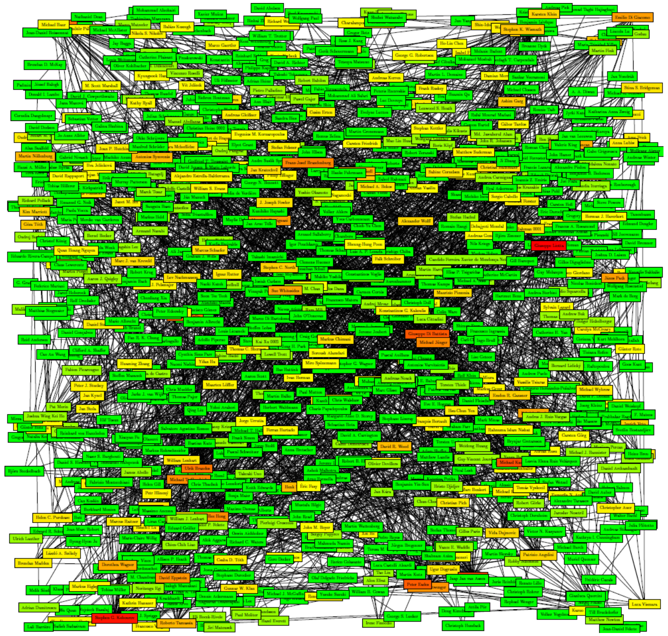
example: invited talk yesterday



Motivation cont.

- social networks
- collaboration/coauthor graphs

example: invited talk yesterday



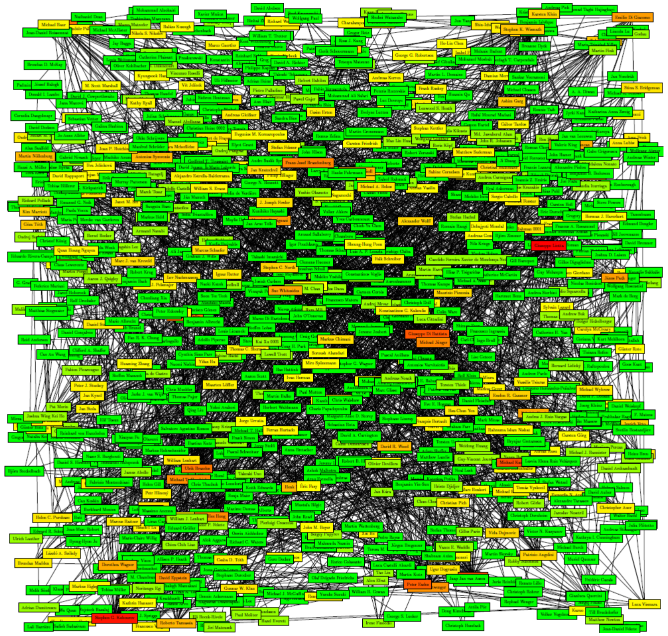
Formally:

Input: – weighted graph $G = (V, E)$
– vertex sizes $w(v)$ and $h(v)$ for each $v \in V$
– prescribed drawing area of width W and height H

Motivation cont.

- social networks
- collaboration/coauthor graphs

example: invited talk yesterday



Formally:

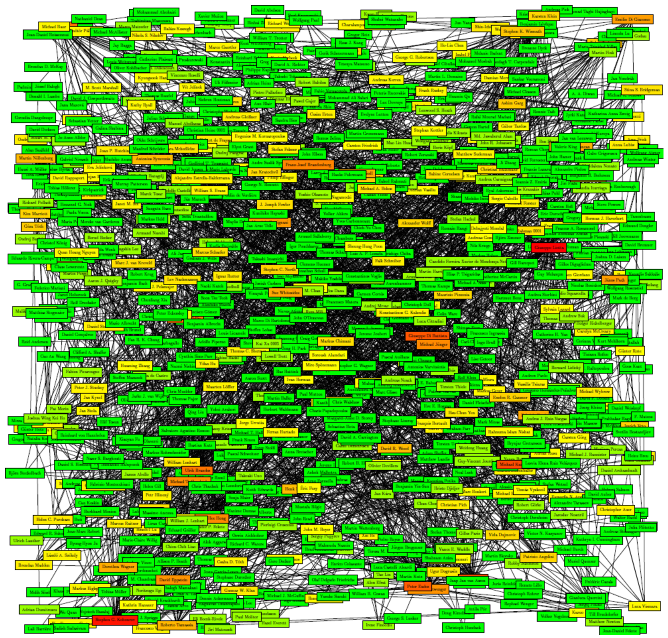
Input: – weighted graph $G = (V, E)$
– vertex sizes $w(v)$ and $h(v)$ for each $v \in V$
– prescribed drawing area of width W and height H

Output: *nice* drawing of **heavy** subgraph G' of G within the drawing area

Motivation cont.

- social networks
- collaboration/coauthor graphs

example: invited talk yesterday



Formally:

Input: – weighted graph $G = (V, E)$

depends on application) and $h(v)$ for each $v \in V$

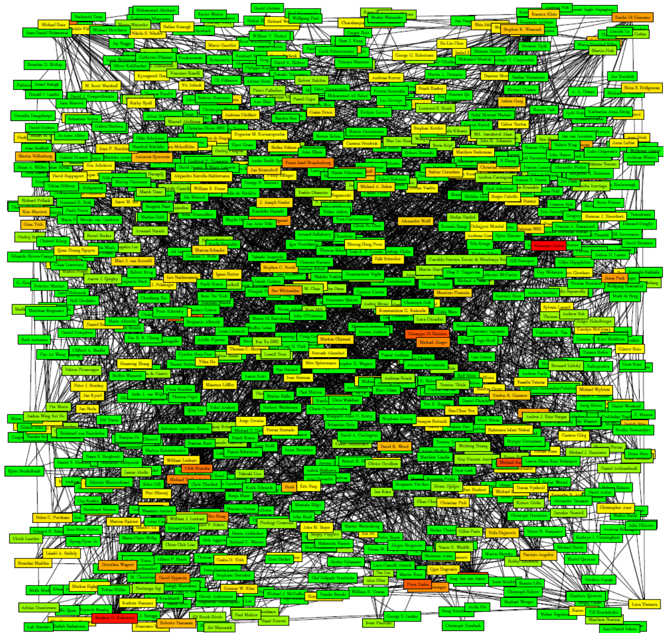
– prescribed drawing area of width W and height H

Output: *nice* drawing of **heavy** subgraph G' of G within the drawing area

Motivation cont.

- social networks
- collaboration/coauthor graphs

example: invited talk yesterday



Formally:

Input: – weighted graph $G = (V, E)$

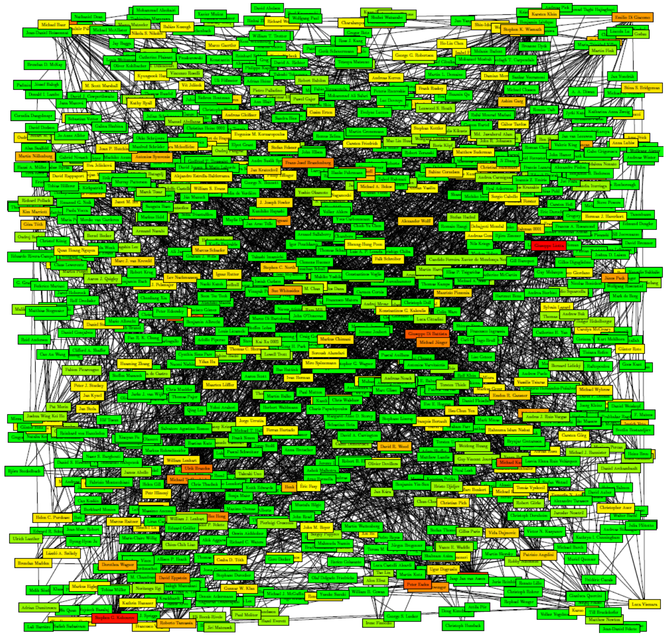
depends on application) and maximize weight V

– prescribed drawing area of width W and height H

Output: *nice* drawing of **heavy** subgraph G' of G within the drawing area

Motivation cont.

- social networks
- collaboration/coauthor graphs



We present heuristics for
– calculation graphs
– straight-line drawings of
general weighted graphs

Formally:

Input: – weighted graph $G = (V, E)$

depends on application) and maximize weight $\in V$

– prescribed drawing area of width W and height H

Output: *nice* drawing of **heavy** subgraph G' of G within the drawing area

Related Work

- [Fruchterman, Reingold 1991]:
  prescribed rectangular drawing area
  but: vertices can be made arbitrarily small

Related Work

- [Fruchterman, Reingold 1991]:
  prescribed rectangular drawing area
  but: vertices can be made arbitrarily small

- constrained graph layout, e.g., [Dwyer et al. 2006]:
  constrain vertex position to prescribed rectangle
  but: no strategy for dropping vertices

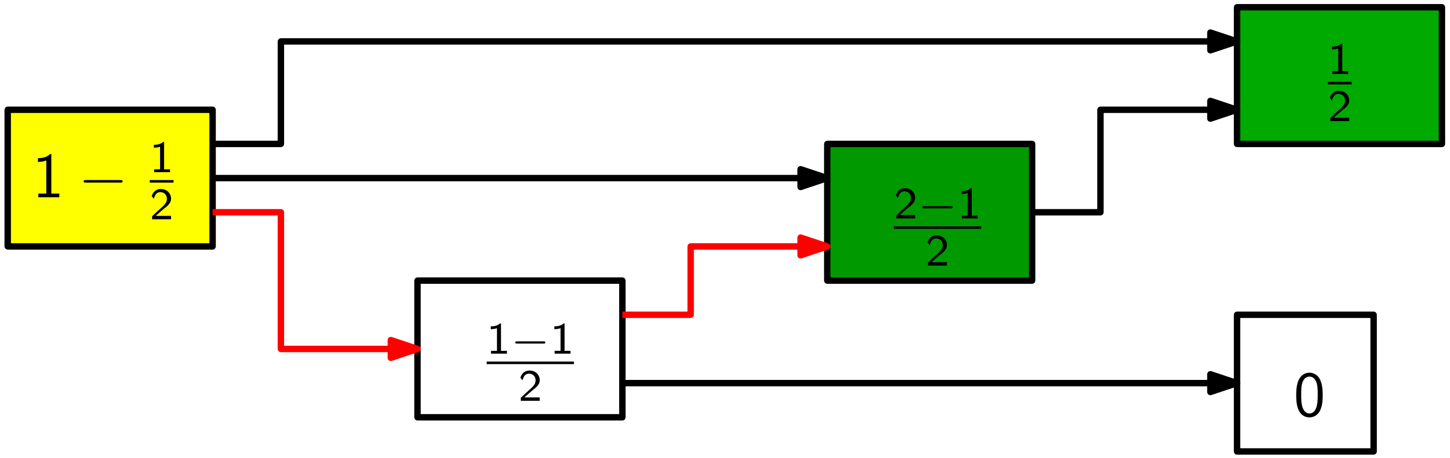
Related Work

- [Fruchterman, Reingold 1991]:
  prescribed rectangular drawing area
  but: vertices can be made arbitrarily small

- constrained graph layout, e.g., [Dwyer et al. 2006]:
  constrain vertex position to prescribed rectangle
  but: no strategy for dropping vertices

- interactive methods for graph exploration:
  – [Dwyer et al. 2008]: overview+detail
  – [Da Lozzo et al. 2011]: graph exploration on smartphone

Drawing Calculation Graphs



- hierarchical drawing (left to right)
- start vertex left
- orthogonal edges

Calculation Graphs – Our Approach

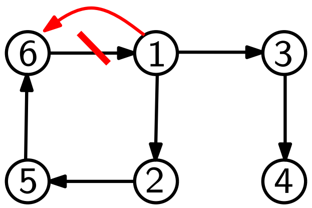
- based on Sugiyama framework
- extra phases for removing vertices/edges

Calculation Graphs – Our Approach

- based on Sugiyama framework
- extra phases for removing vertices/edges
- take weights into account

Calculation Graphs – Our Approach

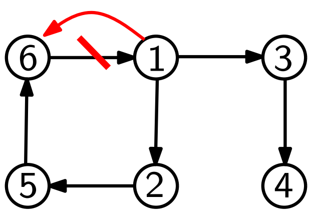
- based on Sugiyama framework
- extra phases for removing vertices/edges
- take weights into account



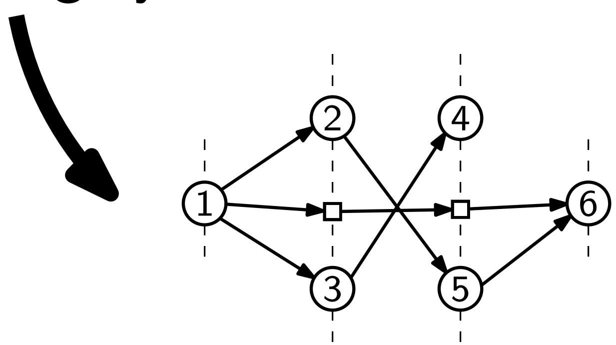
breaking cycles

Calculation Graphs – Our Approach

- based on Sugiyama framework
- extra phases for removing vertices/edges
- take weights into account



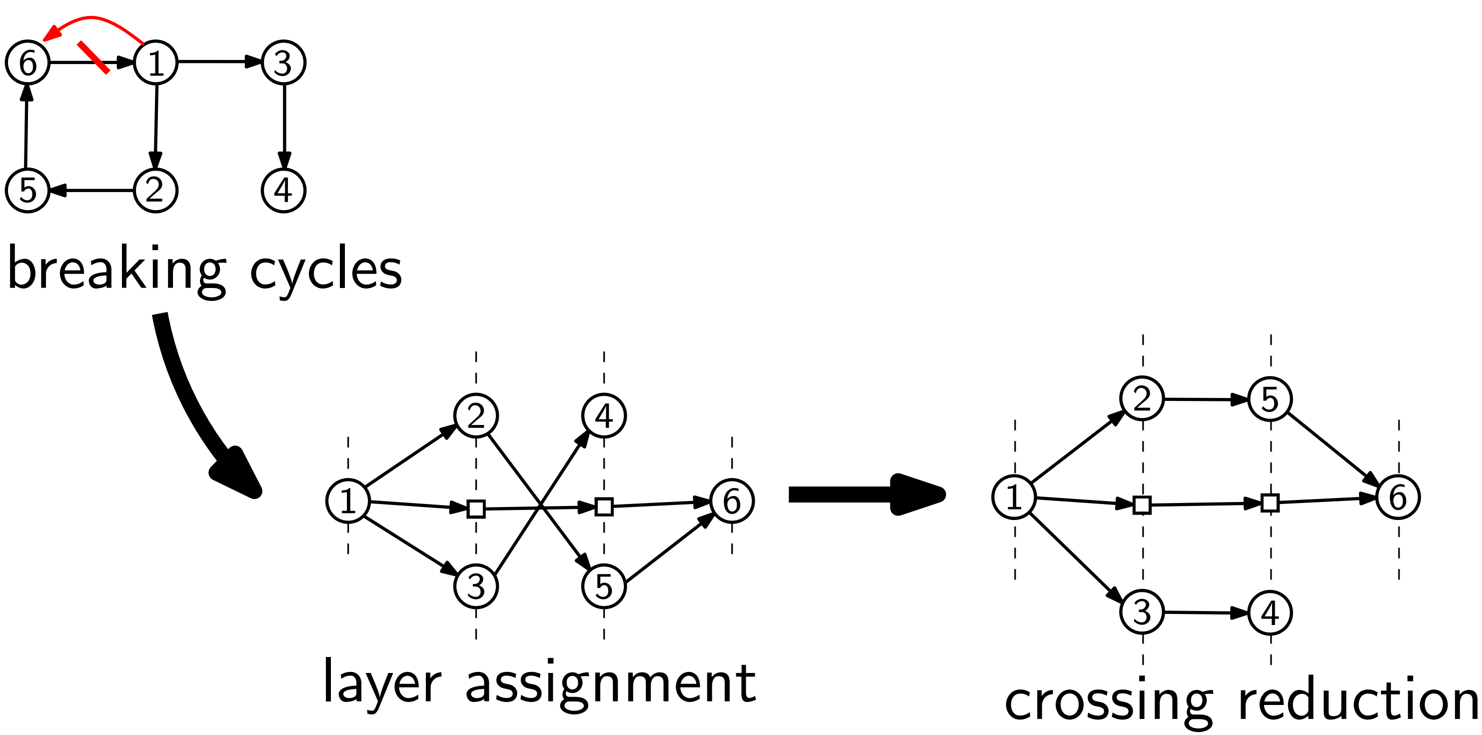
breaking cycles



layer assignment

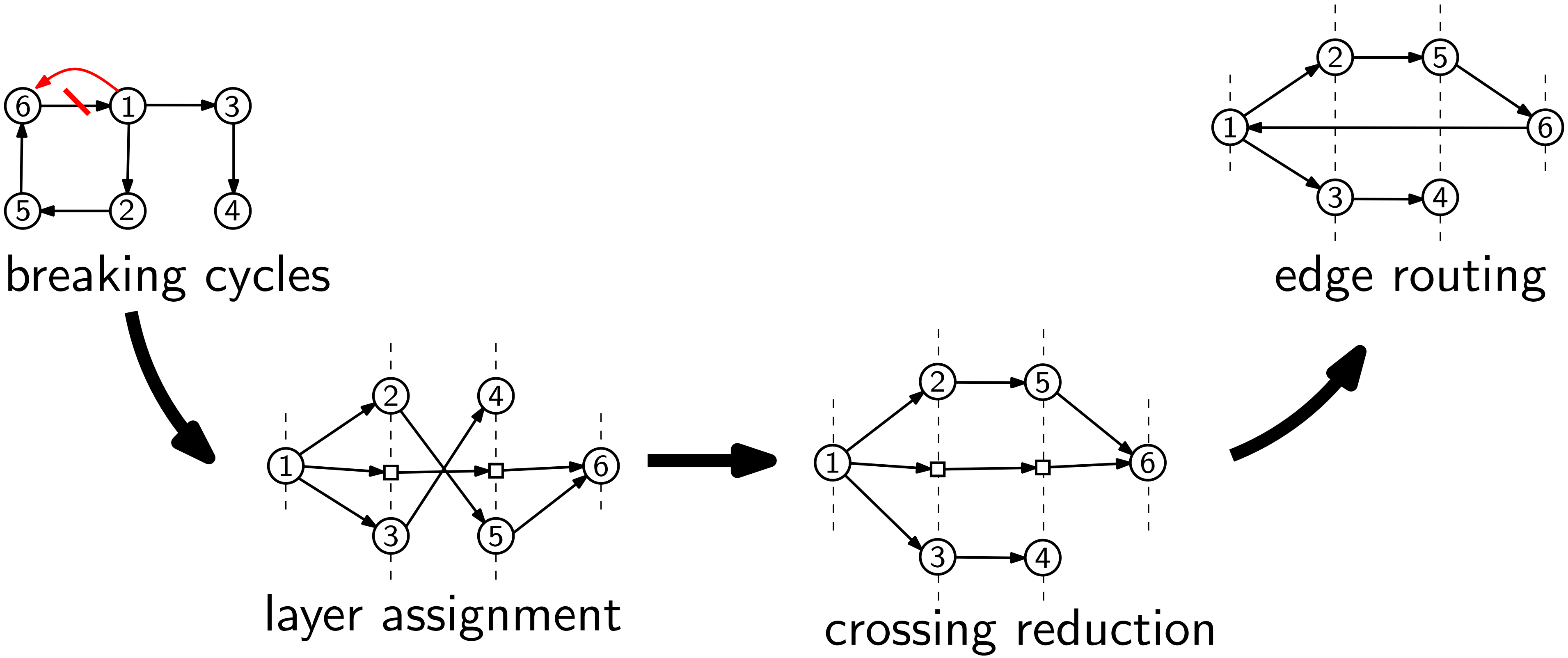
Calculation Graphs – Our Approach

- based on Sugiyama framework
- extra phases for removing vertices/edges
- take weights into account



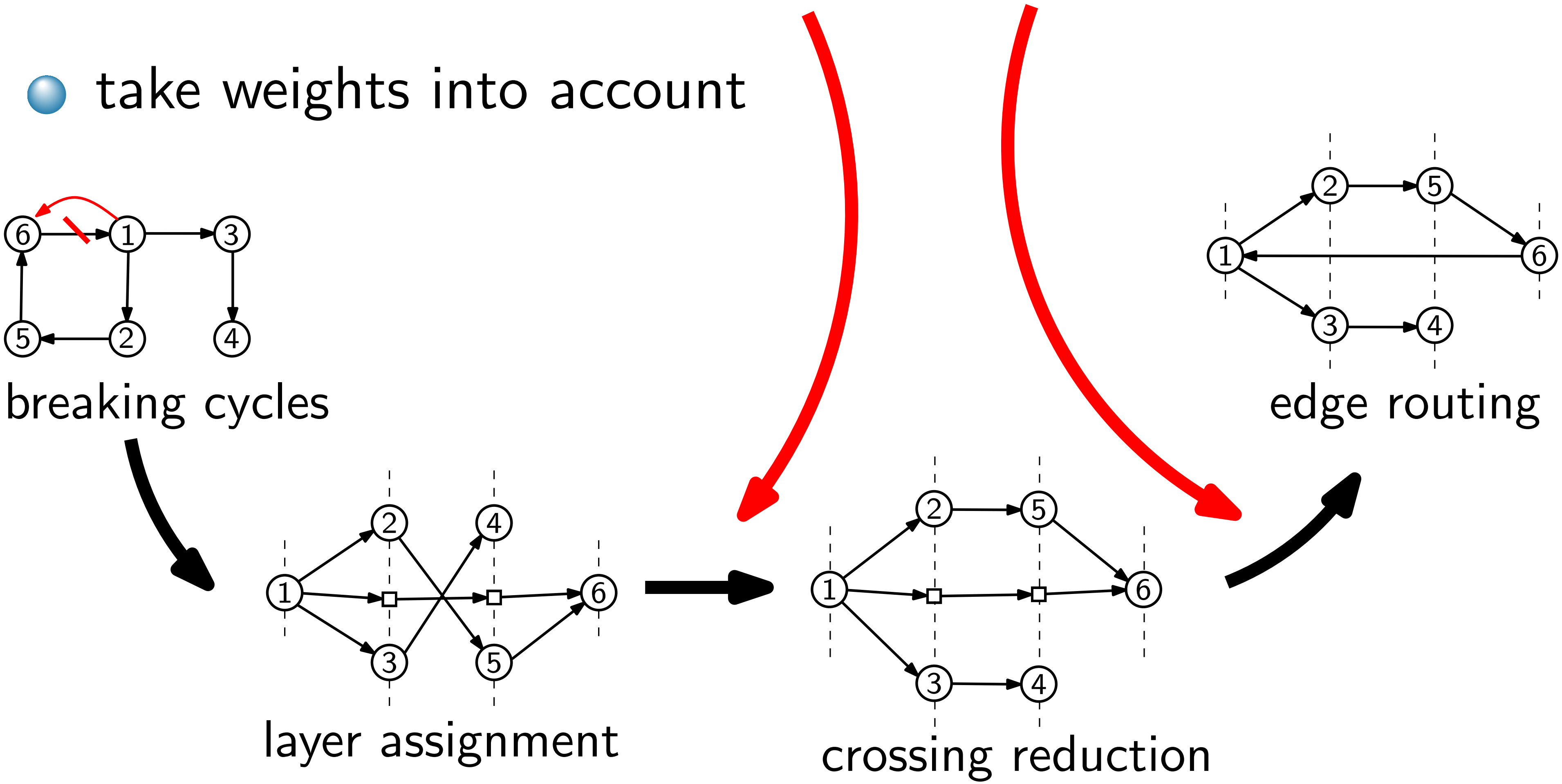
Calculation Graphs – Our Approach

- based on Sugiyama framework
- extra phases for removing vertices/edges
- take weights into account



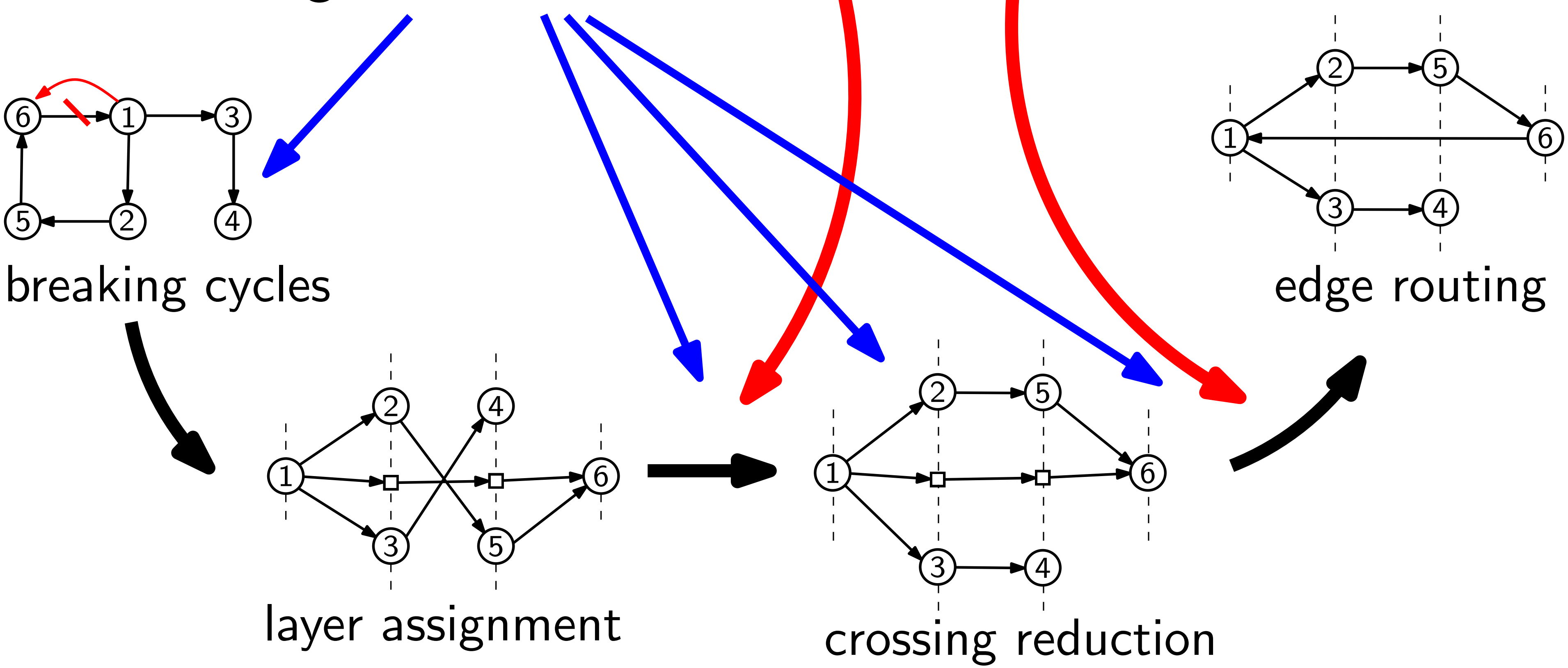
Calculation Graphs – Our Approach

- based on Sugiyama framework
- extra phases for removing vertices/edges
- take weights into account



Calculation Graphs – Our Approach

- based on Sugiyama framework
- extra phases for removing vertices/edges
- take weights into account

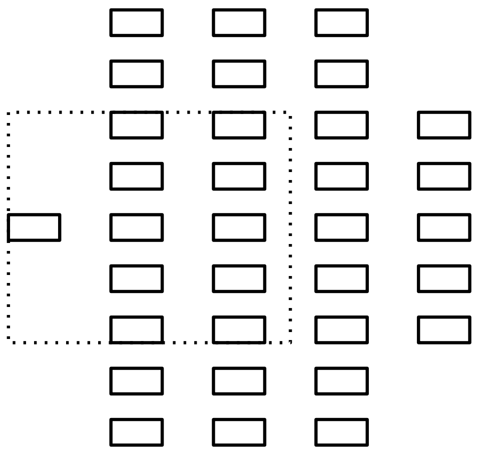


Our Adjustments

- breaking cycles: minimize **weight** of reverted edges
  can afford ILP solution

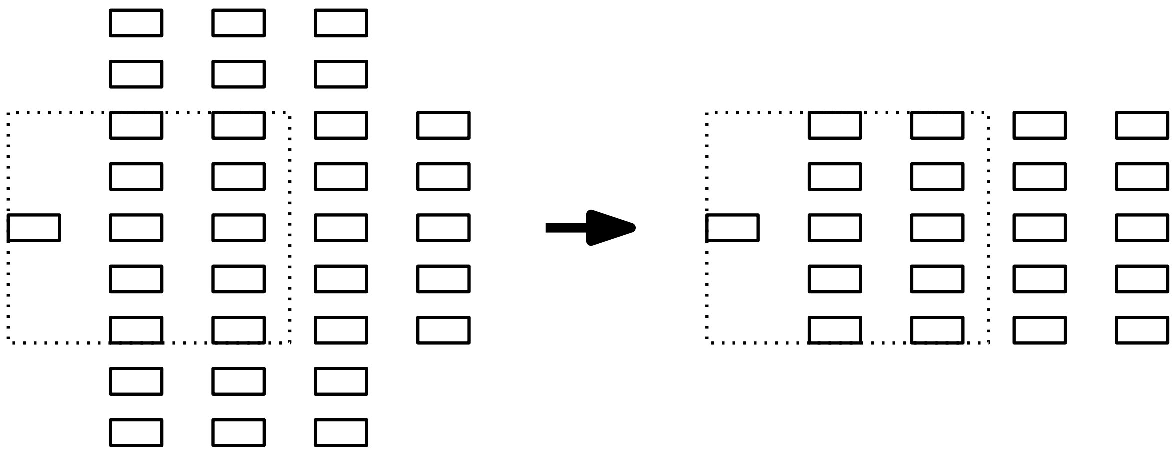
Our Adjustments

- breaking cycles: minimize **weight** of reverted edges
  can afford ILP solution
- layer assignment: first use overfull layers



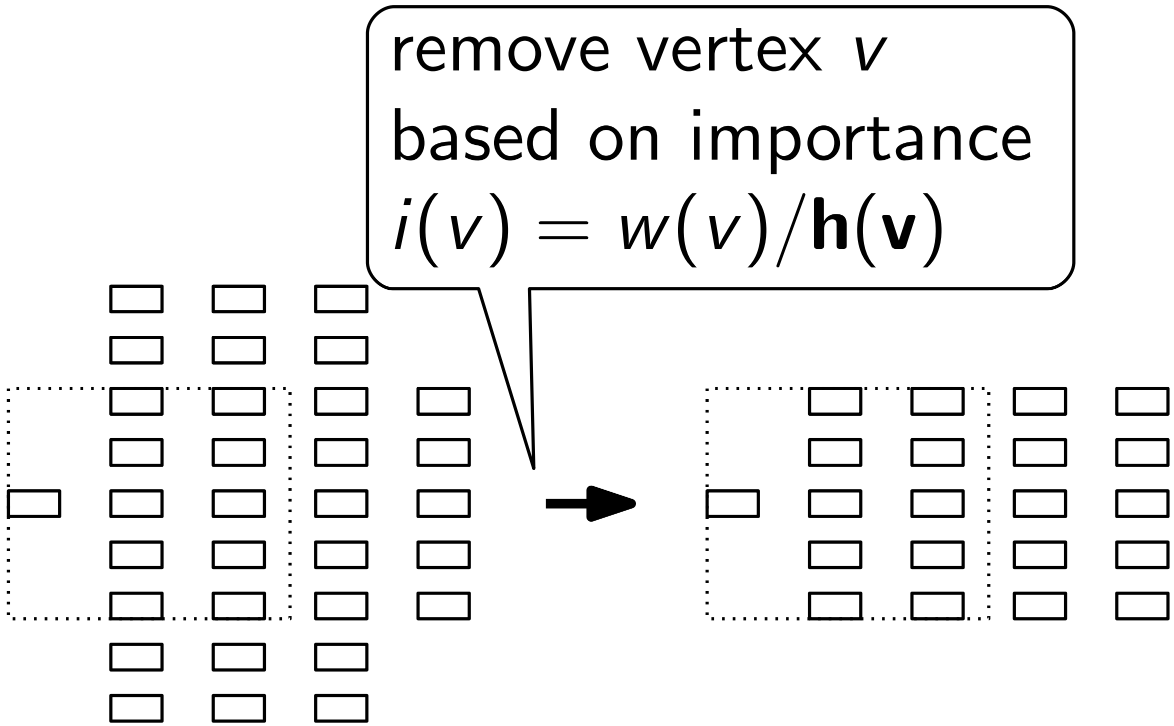
Our Adjustments

- breaking cycles: minimize **weight** of reverted edges
  can afford ILP solution
- layer assignment: first use overfull layers
- from left to right: remove vertices until layer small enough



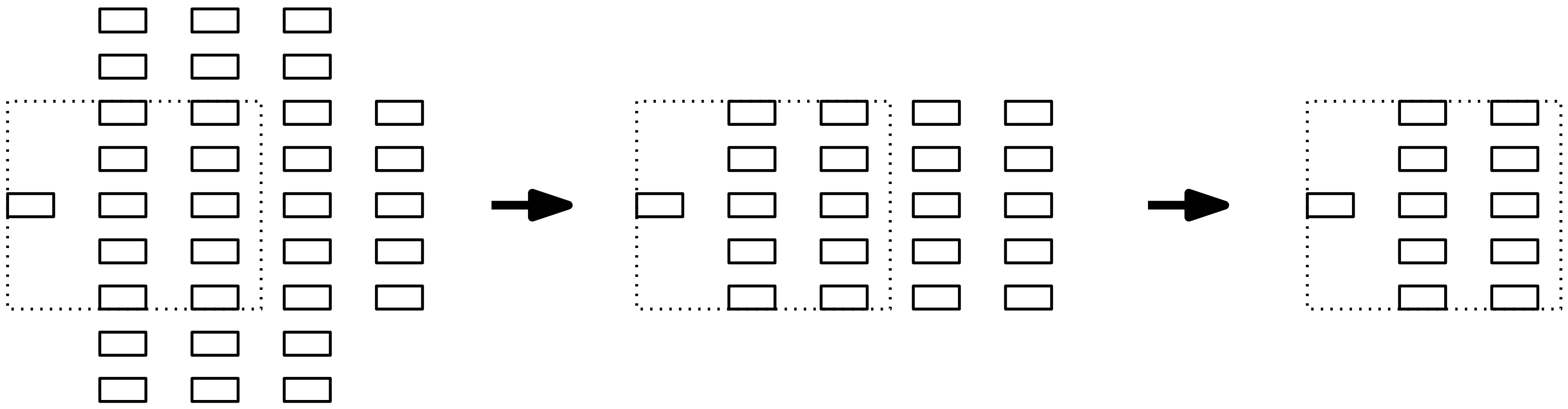
Our Adjustments

- breaking cycles: minimize **weight** of reverted edges
  can afford ILP solution
- layer assignment: first use overfull layers
- from left to right: remove vertices until layer small enough



Our Adjustments

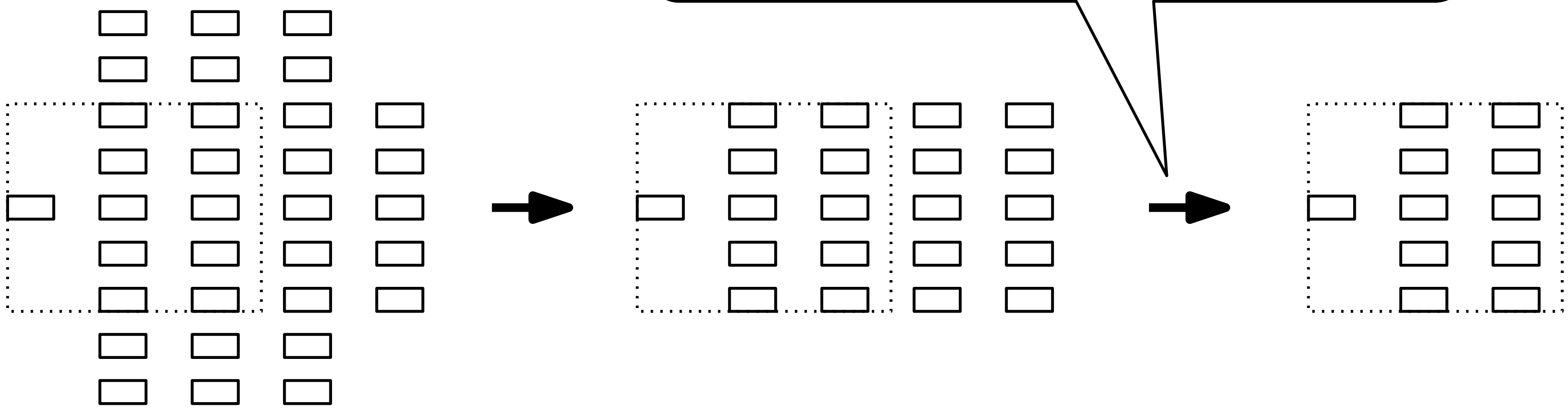
- breaking cycles: minimize **weight** of reverted edges
  can afford ILP solution
- layer assignment: first use overfull layers
- from left to right: remove vertices until layer small enough
- remove layers until all vertices fit into drawing area



Our Adjustments

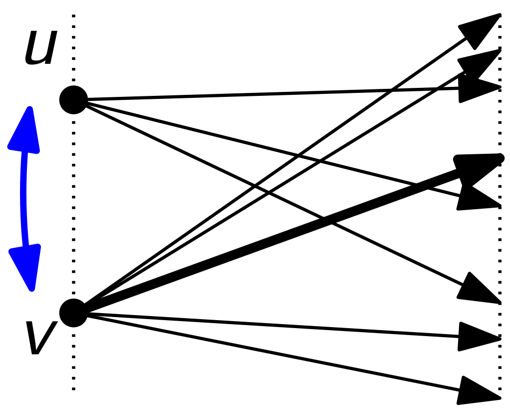
- breaking cycles: minimize **weight** of reverted edges
  can afford ILP solution
- layer assignment: first use overfull layers
- from left to right: remove vertices until layer small enough
- remove layers until all

remove layer L based on importance
$i(L) = \sum_{v \in L} w(v)/\text{width}(L)$



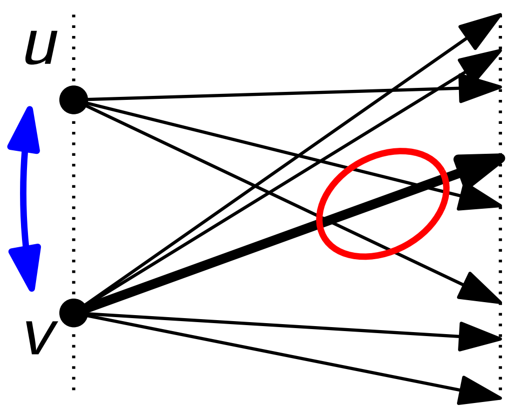
Our Adjustments

- breaking cycles: minimize **weight** of reverted edges
  can afford ILP solution
- layer assignment: first use overfull layers
- from left to right: remove vertices until layer small enough
- remove layers until all vertices fit into drawing area
- crossing minimization: adjacent-exchange heuristic can
  take weights of crossing edges into account



Our Adjustments

- breaking cycles: minimize **weight** of reverted edges
  can afford ILP solution
- layer assignment: first use overfull layers
- from left to right: remove vertices until layer small enough
- remove layers until all vertices fit into drawing area
- crossing minimization: adjacent-exchange heuristic can
  take weights of crossing edges into account



higher cost for crossing heavy edge

Our Adjustments

- breaking cycles: minimize **weight** of reverted edges
  can afford ILP solution
- layer assignment: first use overfull layers
- from left to right: remove vertices until layer small enough
- remove layers until all vertices fit into drawing area
- crossing minimization: adjacent-exchange heuristic can
  take weights of crossing edges into account
- too many crossings: remove edge e based on importance

$$i(e) = \frac{w(e)}{\text{total weight of edges crossing } e}$$

Our Adjustments

- breaking cycles: minimize **weight** of reverted edges
  can afford ILP solution
- layer assignment: first use overfull layers
- from left to right: remove vertices until layer small enough
- remove layers until all vertices fit into drawing area
- crossing ristic can
  take we

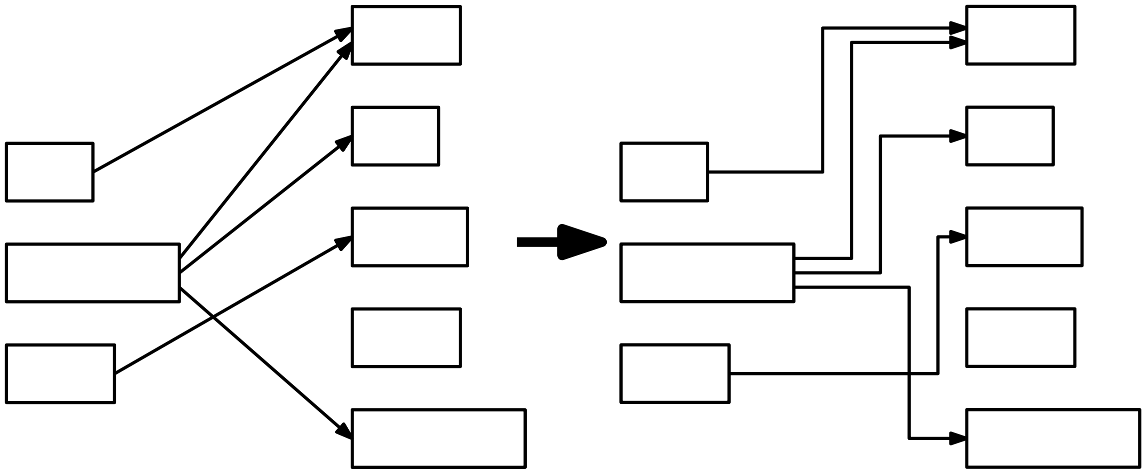
extension: try to reinsert removed
objects if space available

- too many crossings: remove edge e based on importance

$$i(e) = \frac{w(e)}{\text{total weight of edges crossing } e}$$

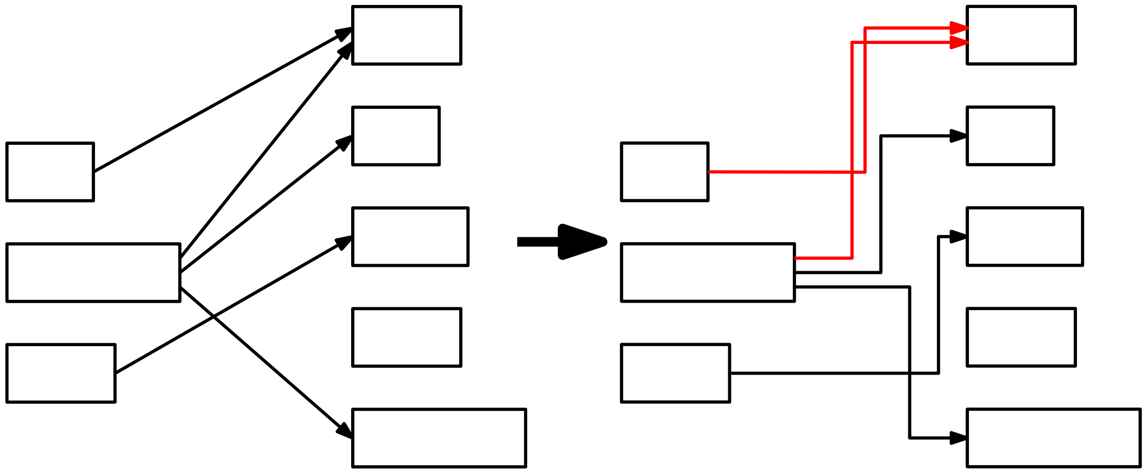
Edge Routing

- place vertical segments between consecutive layers



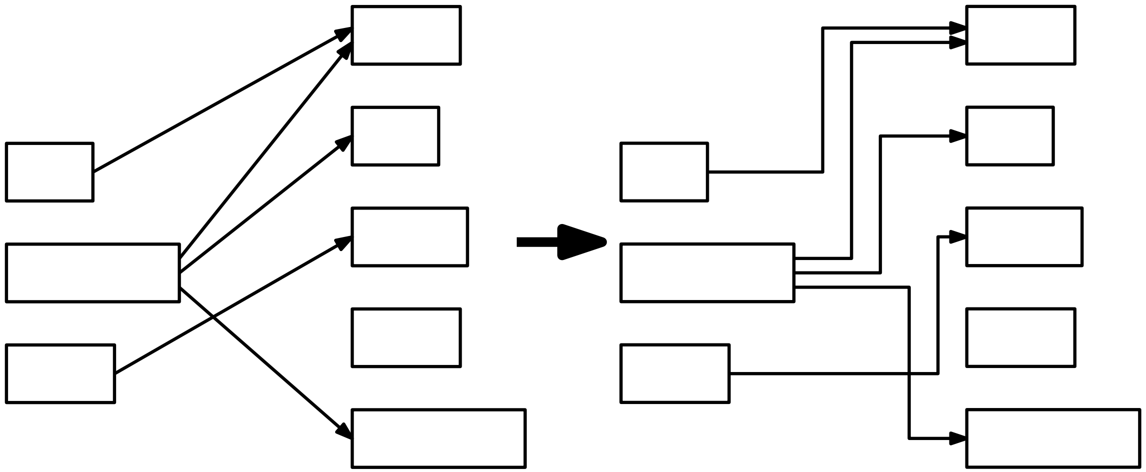
Edge Routing

- place vertical segments between consecutive layers
- no unnecessary crossings



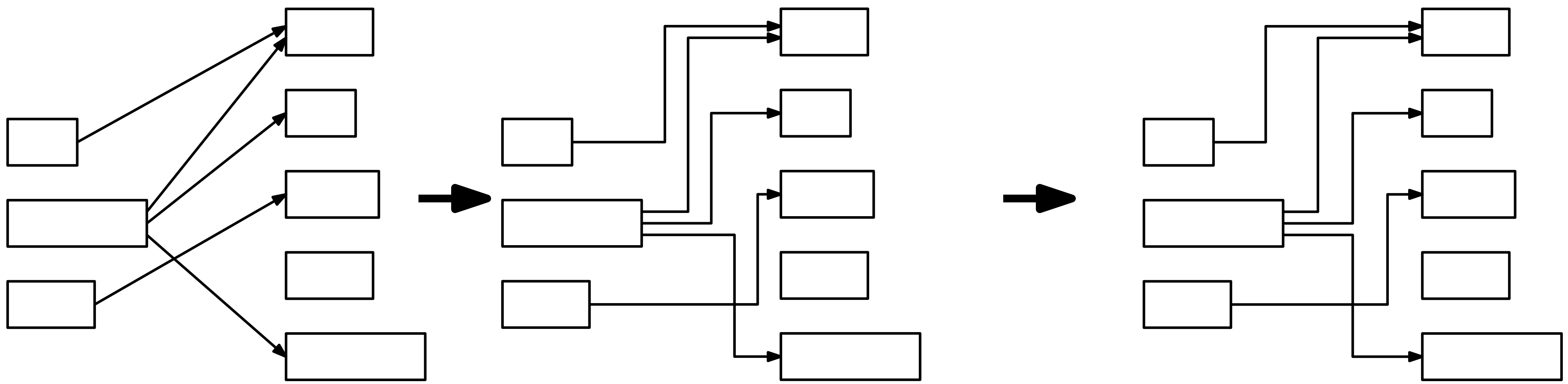
Edge Routing

- place vertical segments between consecutive layers
- no unnecessary crossings



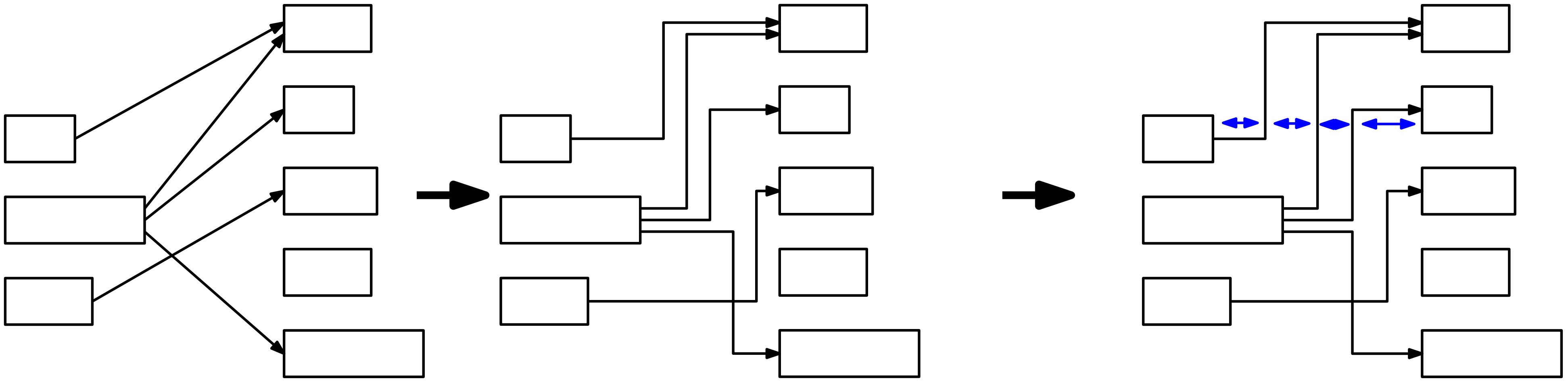
Edge Routing

- place vertical segments between consecutive layers
- no unnecessary crossings
- improve spacing of segments by force-directed method



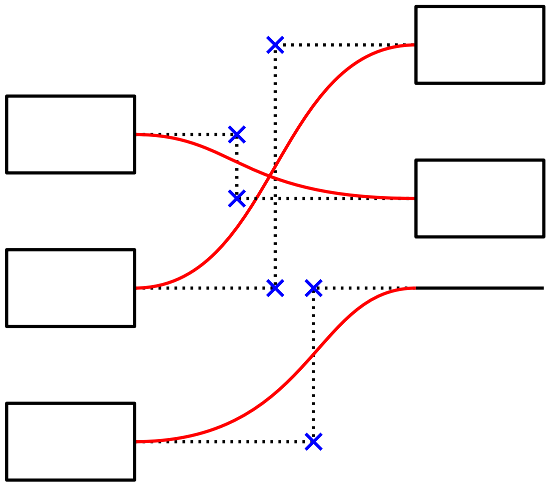
Edge Routing

- place vertical segments between consecutive layers
- no unnecessary crossings
- improve spacing of segments by force-directed method



Edge Routing

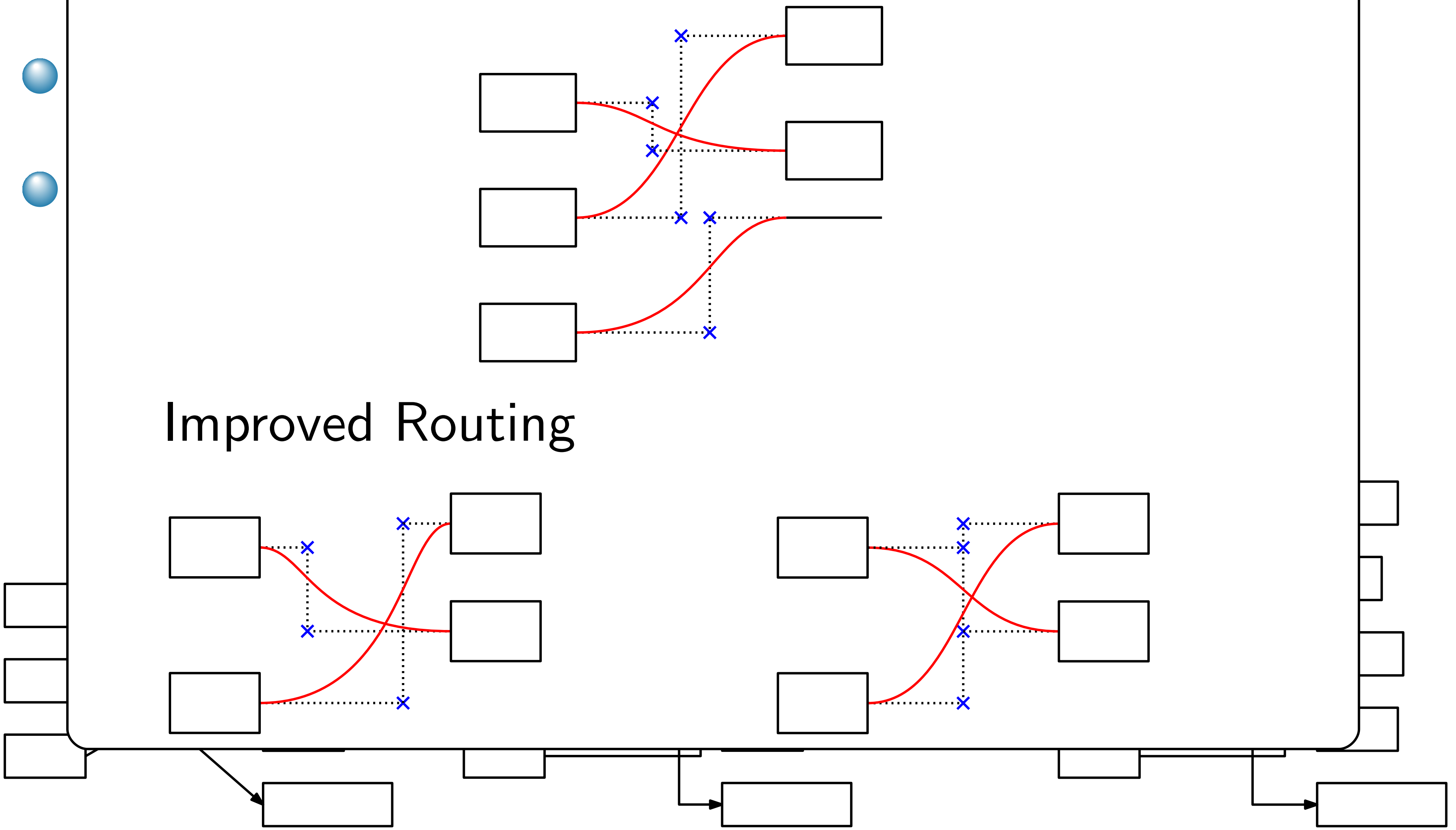
Extension: Bézier Curves



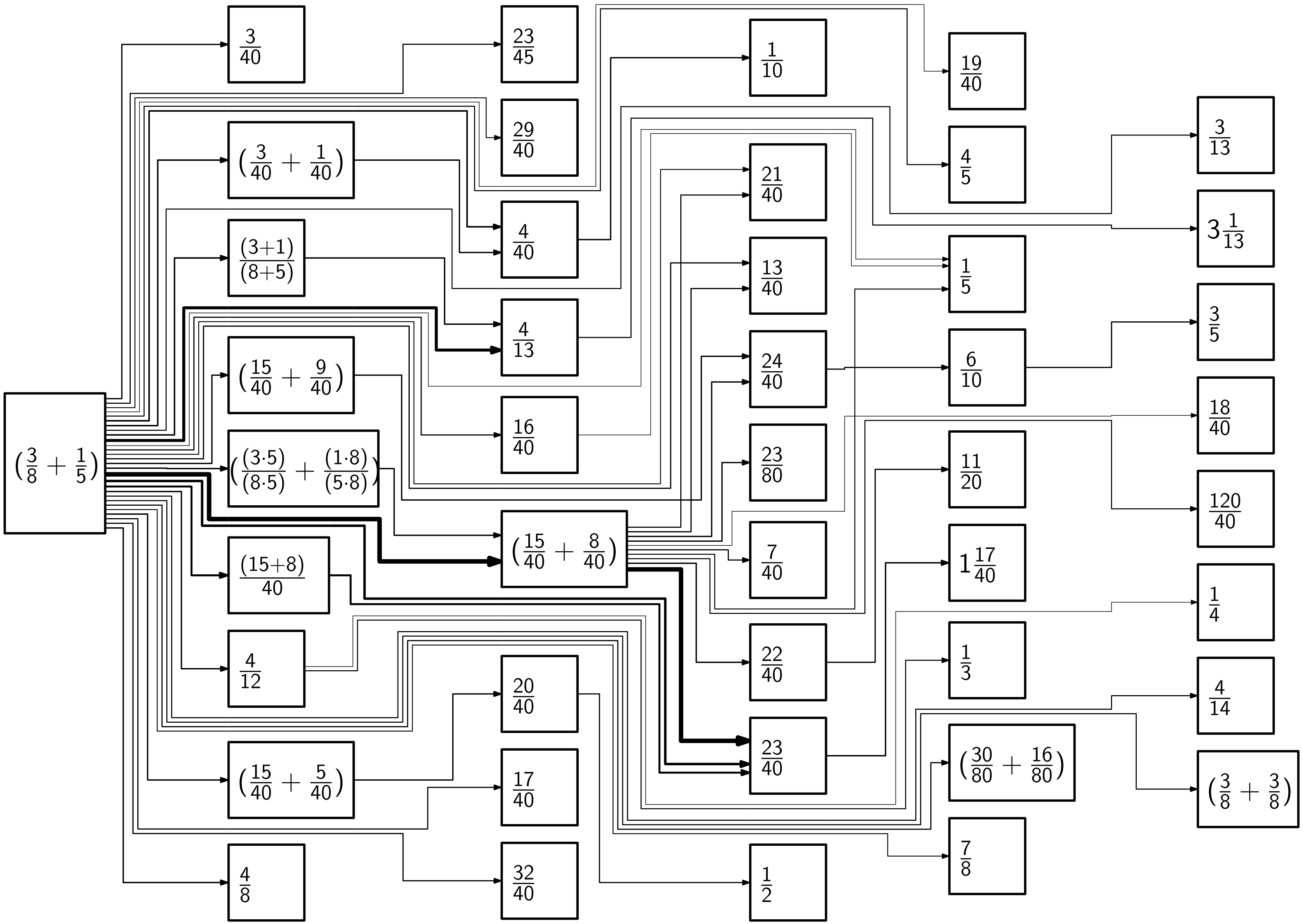
Edge Routing

Extension: Bézier Curves

Improved Routing

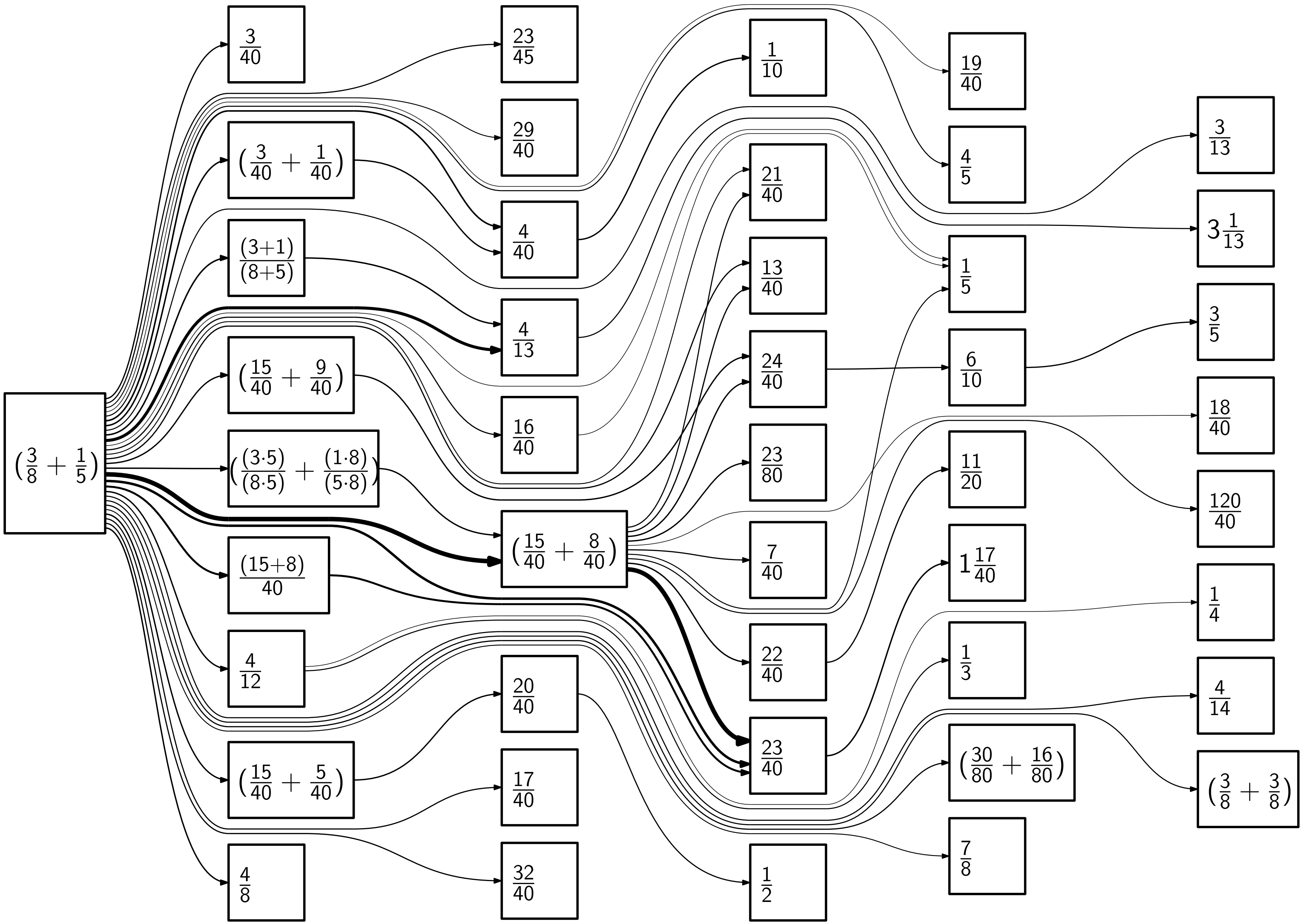


Example



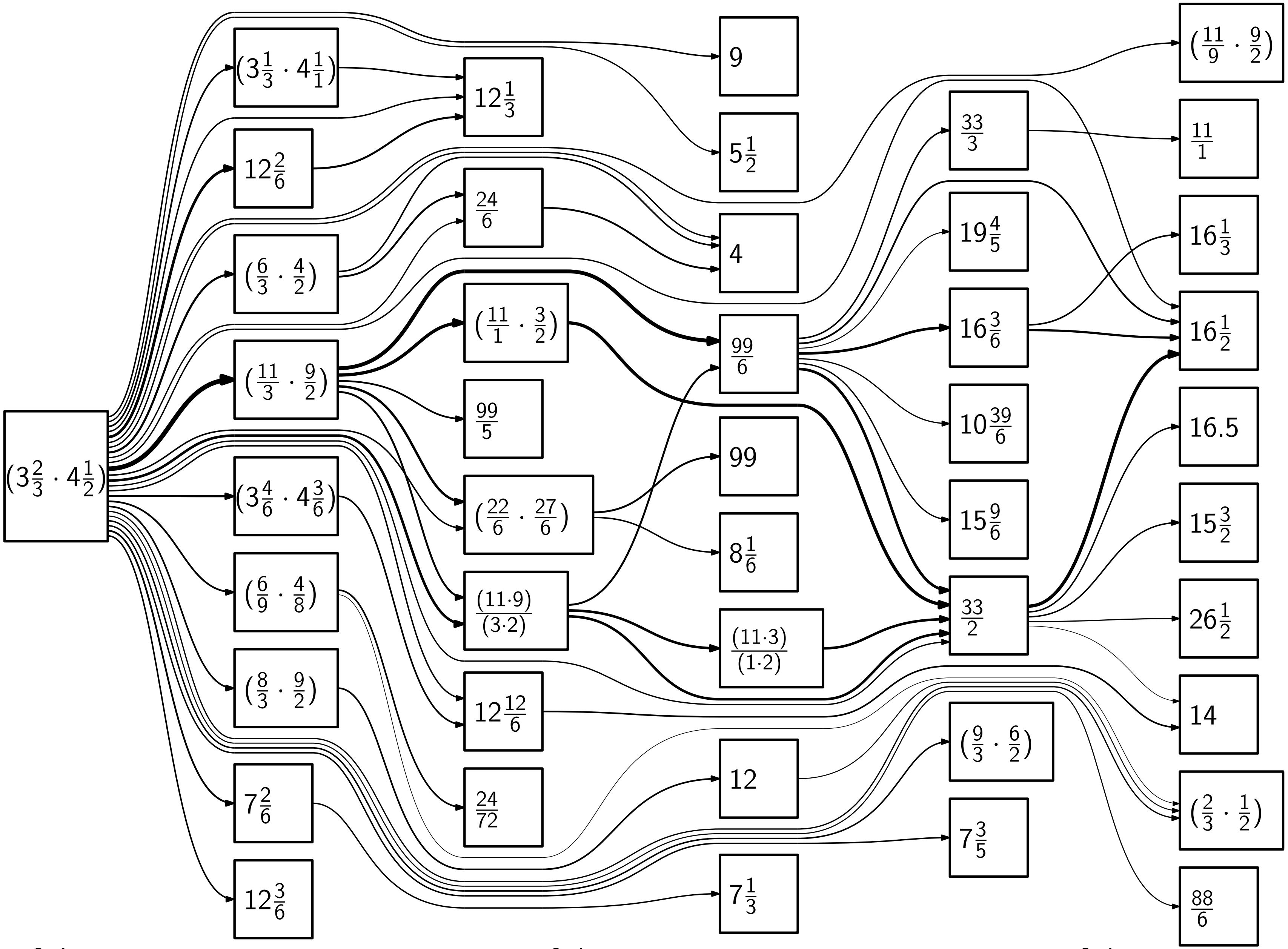
38 of 358 vertices (10.6 %); 91.1 % of the vertex weight

Example



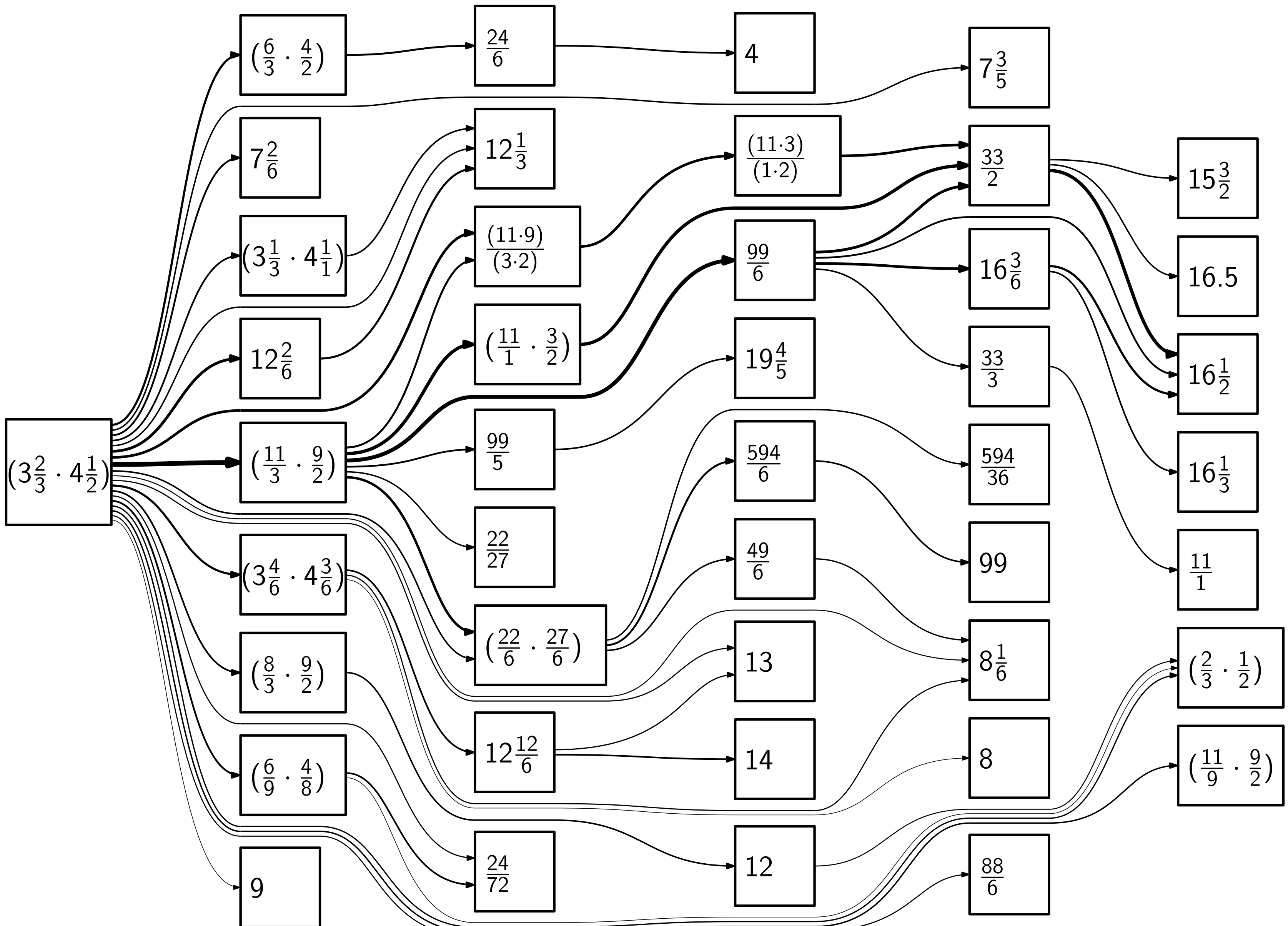
38 of 358 vertices (10.6 %); 91.1 % of the vertex weight

Example 2



4.7 % of 1031 vert.; 75.9 % vertex weight; 74.1 % edge weight

Example 2 – Planar Output



4.3 % of 1031 vert.; 76.6 % vertex weight; 70.2 % edge weight

Remarks

- runtime for graph with 1031 vertices and 1549 edges to A4 paper: $\approx 3s$

Remarks

- runtime for graph with 1031 vertices and 1549 edges to A4 paper: $\approx 3s$

  most of it for edge routing

Remarks

- runtime for graph with 1031 vertices and 1549 edges to A4 paper: $\approx 3s$

  most of it for edge routing

- preprocessing removing very light vertices yielded speedup + heavier outputs

Remarks

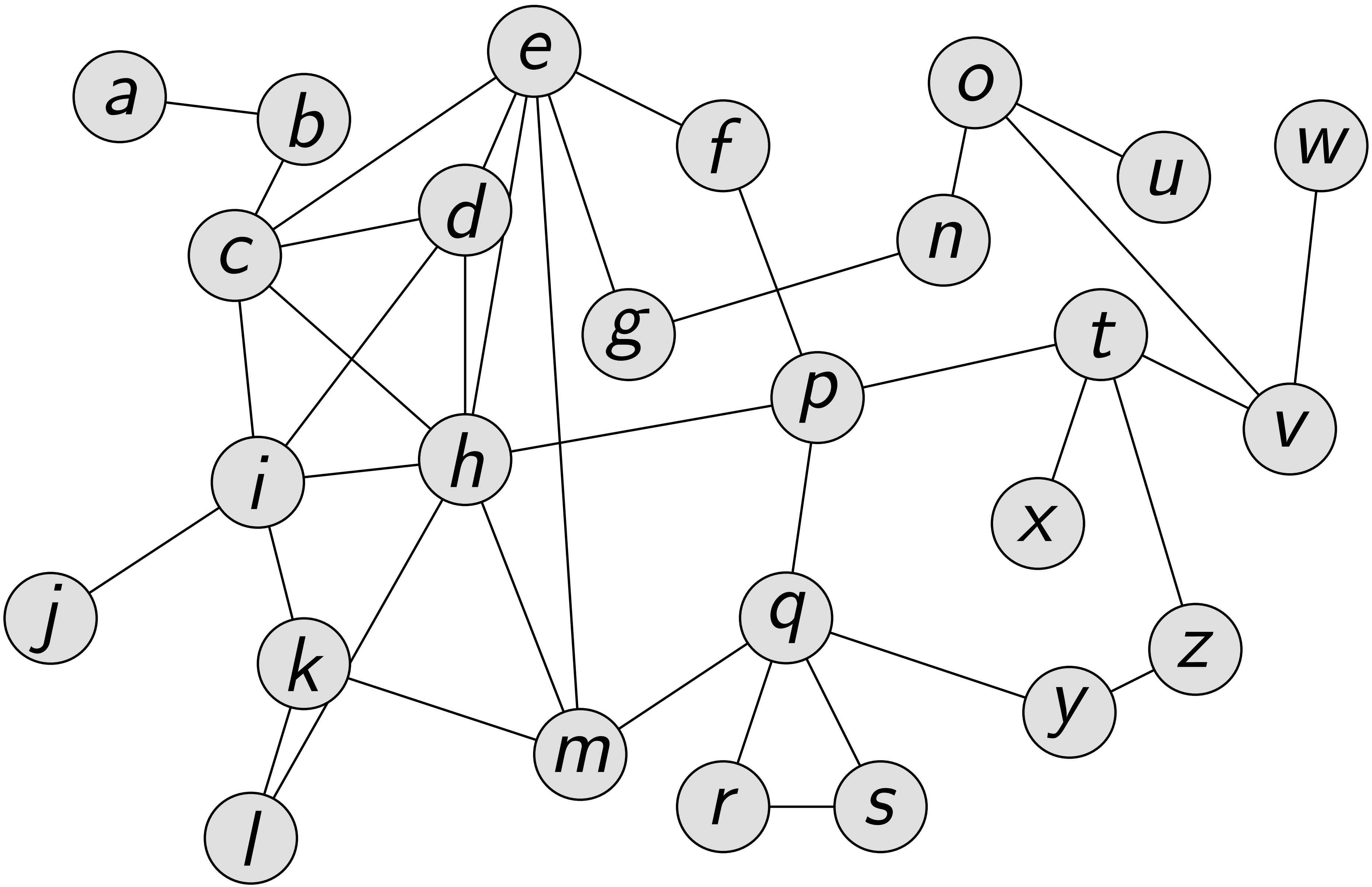
- runtime for graph with 1031 vertices and 1549 edges to A4 paper: $\approx 3s$

  most of it for edge routing

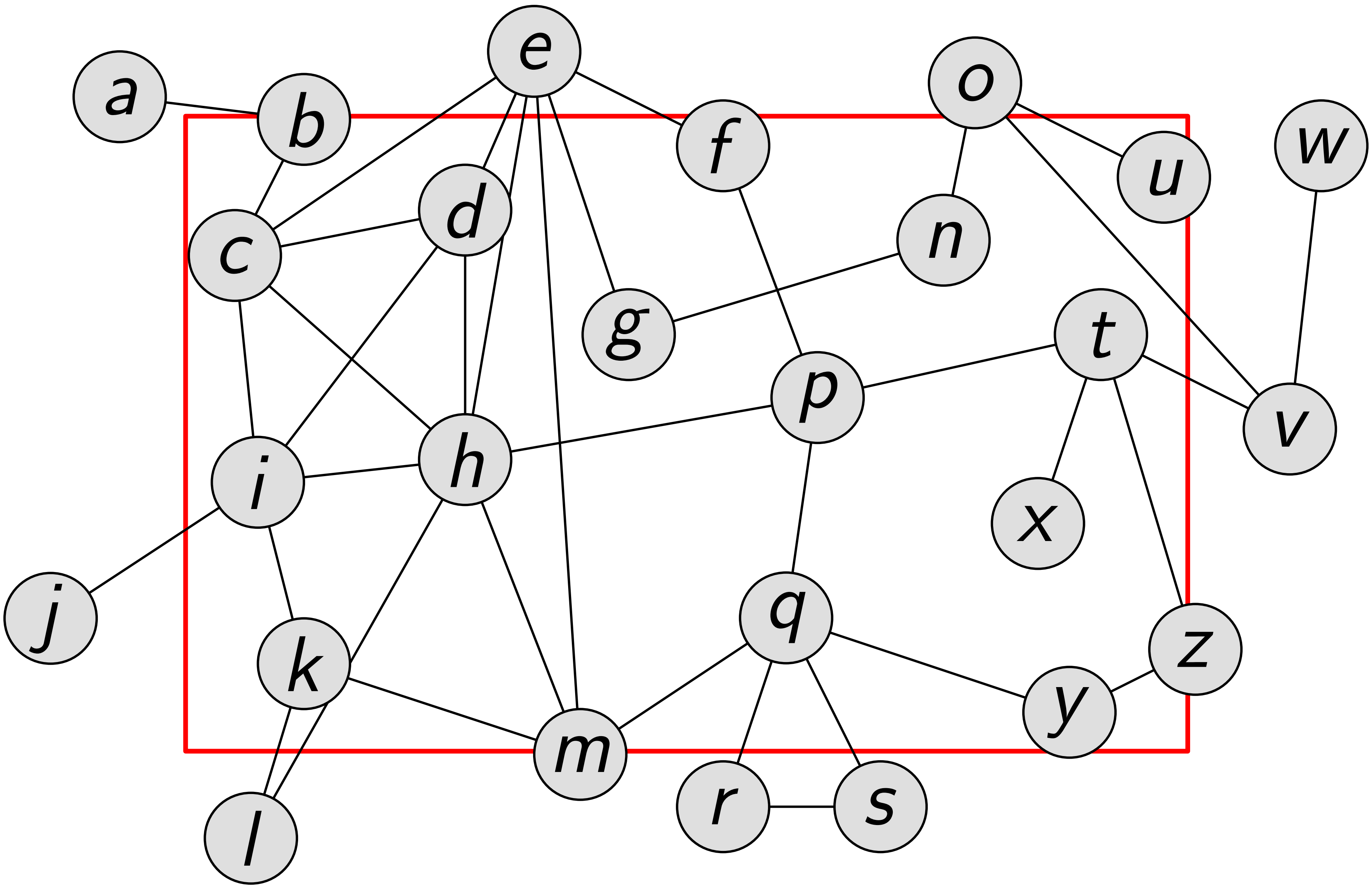
- preprocessing removing very light vertices yielded speedup + heavier outputs

- especially with vertex/edge reinsertion: no significant influence of layer assignment algorithm

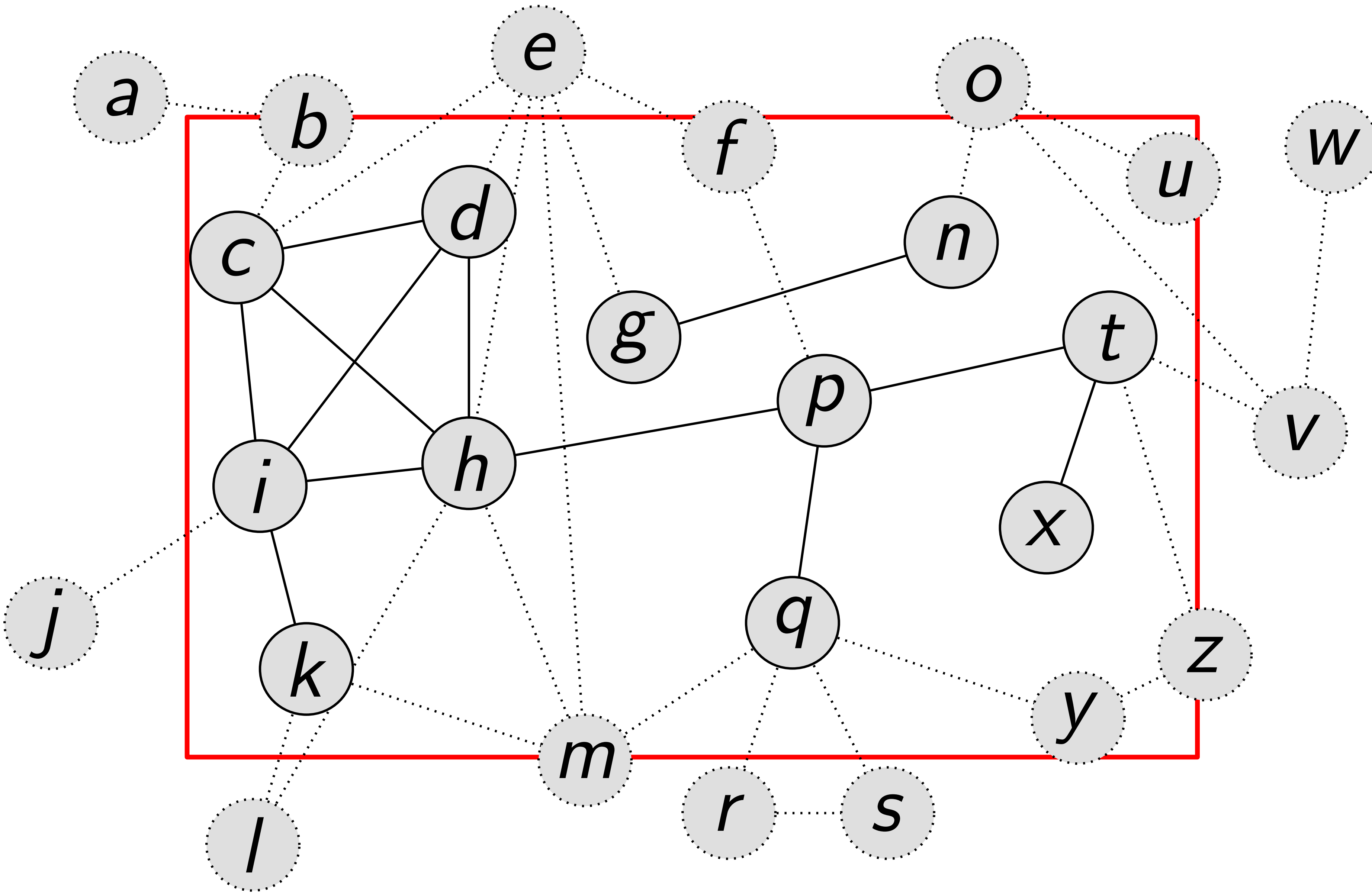
General Graphs



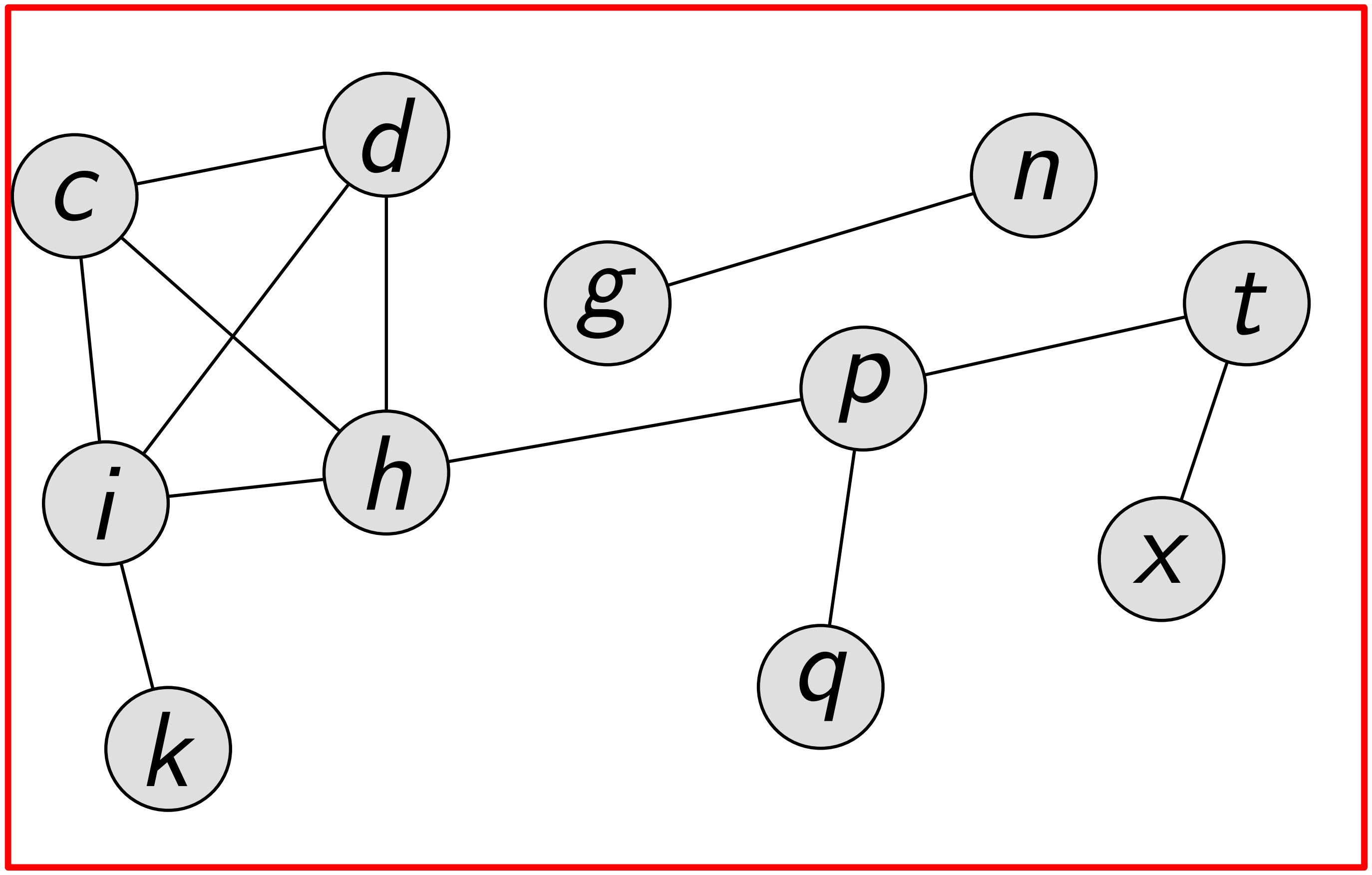
General Graphs



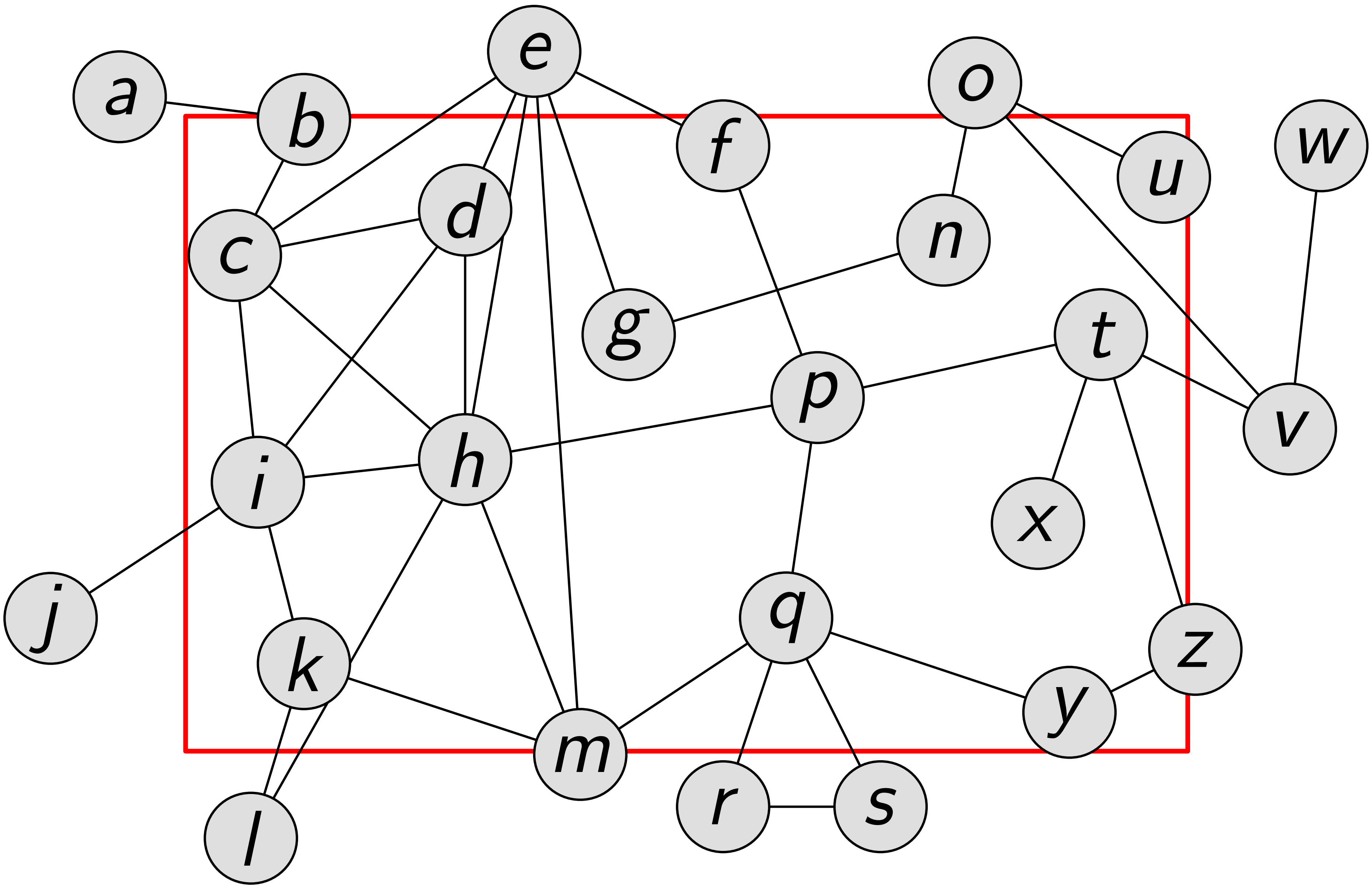
General Graphs



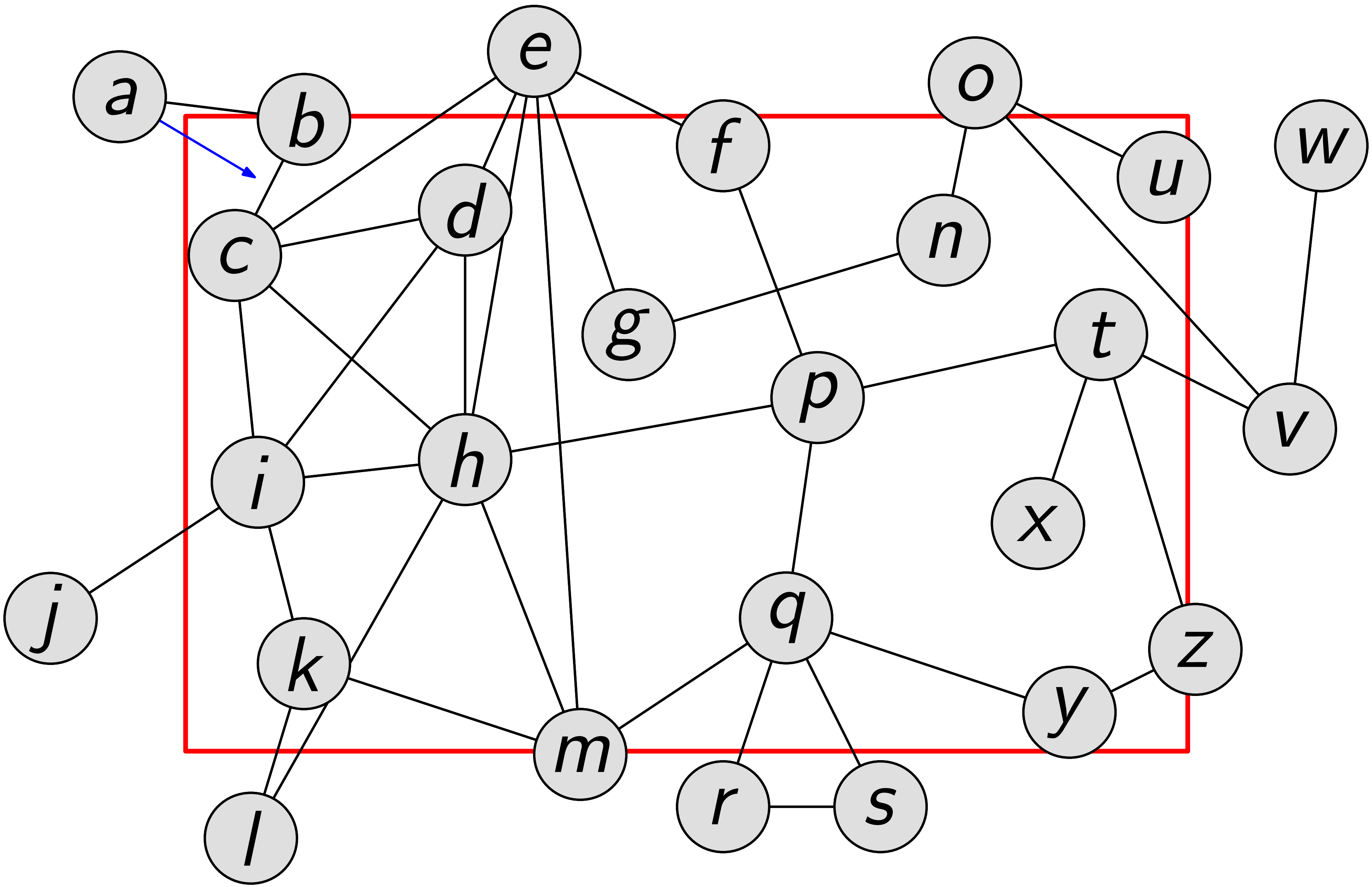
General Graphs



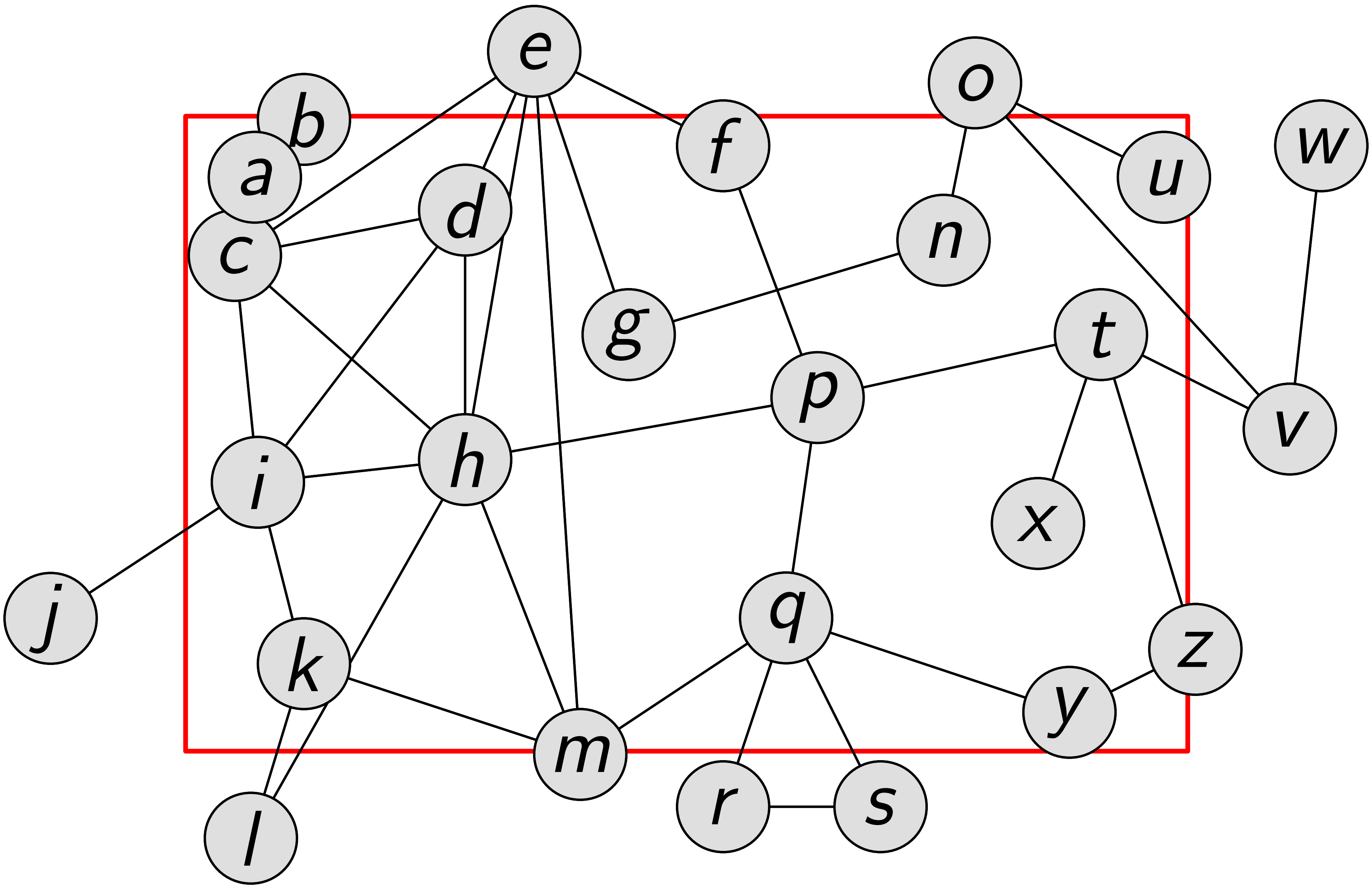
General Graphs



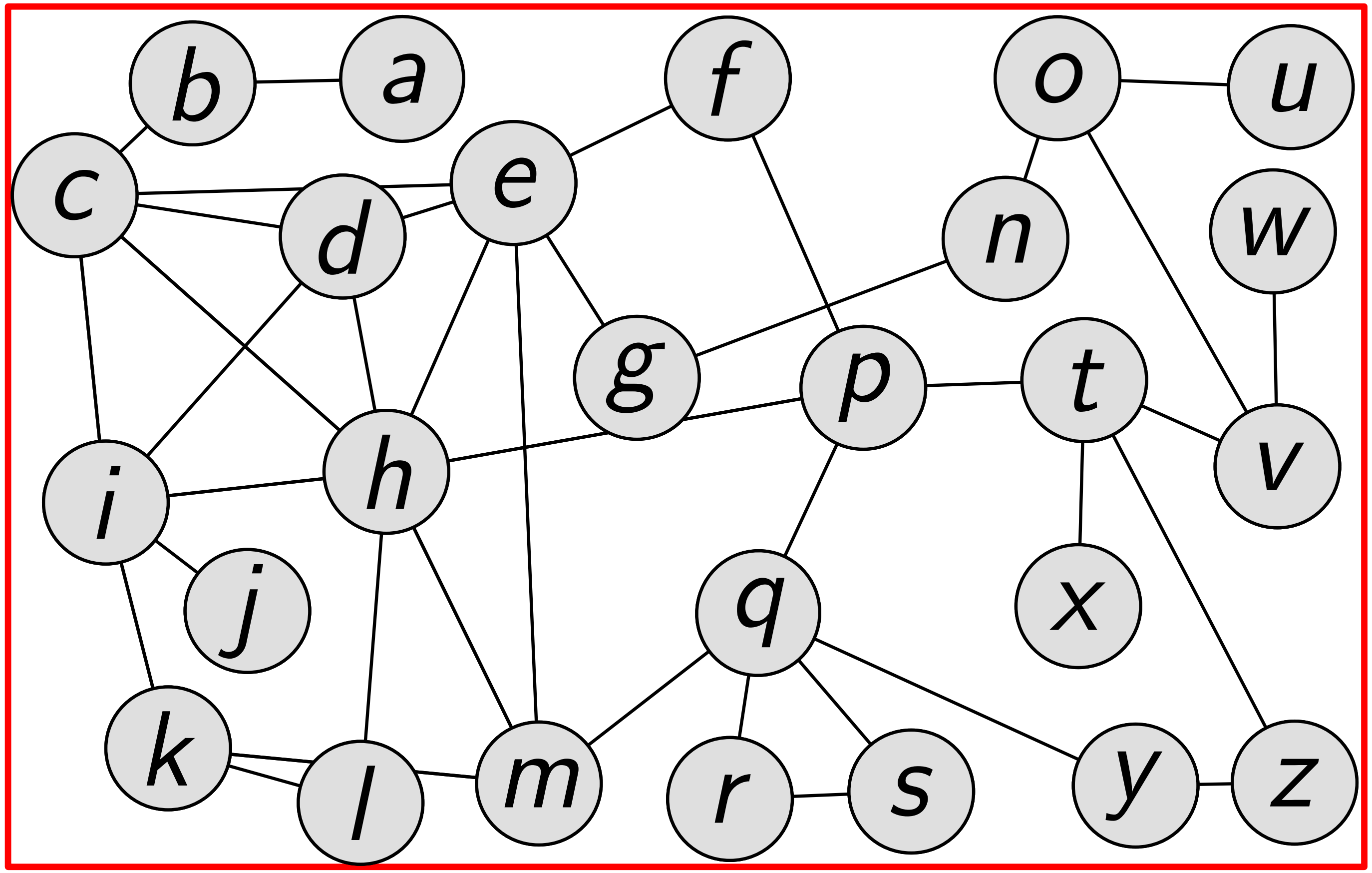
General Graphs



General Graphs

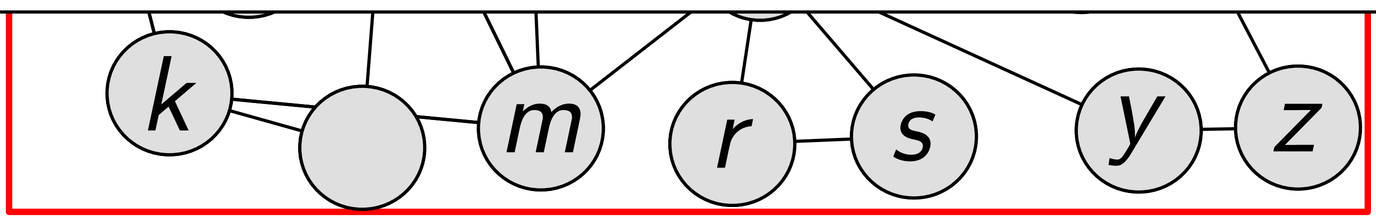


General Graphs



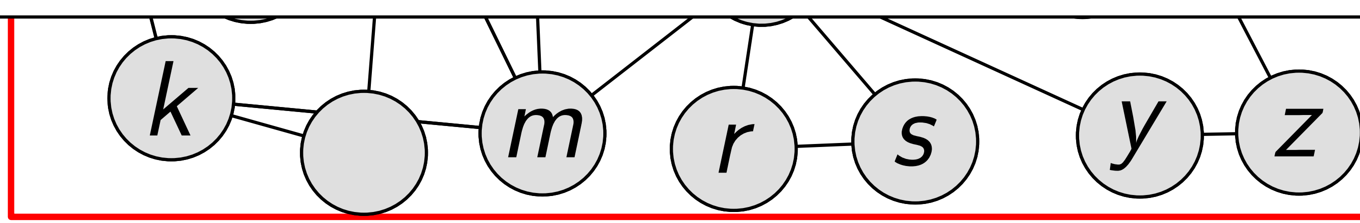
General Graphs – Basic Ideas

- use force-directed approach + standard forces



General Graphs – Basic Ideas

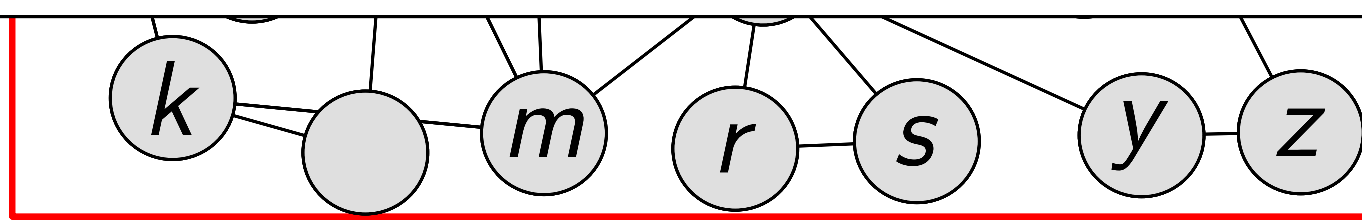
- use force-directed approach + standard forces
- keep drawing in prescribed *frame*



General Graphs – Basic Ideas

- use force-directed approach + standard forces
- keep drawing in prescribed *frame*

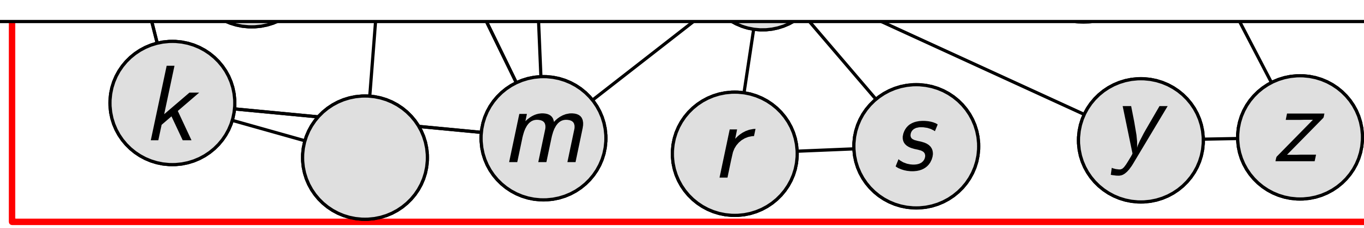
compare [Fruchterman, Reingold '91]



General Graphs – Basic Ideas

- use force-directed approach + standard forces
- ~~keep drawing in prescribed *frame*~~
- keep drawing in *frame*
- slowly shrink frame

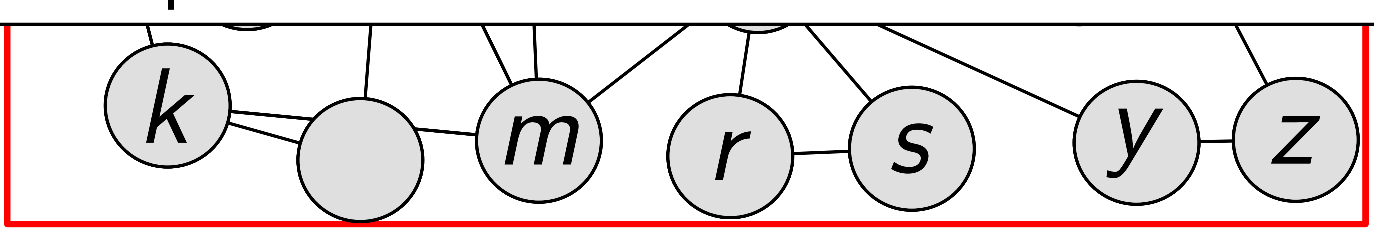
compare [Fruchterman, Reingold '91]



General Graphs – Basic Ideas

- use force-directed approach + standard forces
- ~~keep drawing in prescribed *frame*~~
- keep drawing in *frame*
- slowly shrink frame
- remove vertices/edges if necessary after shrinking
- find new equilibrium

compare [Fruchterman, Reingold '91]



General Graphs – Outline

- start with some initial drawing

General Graphs – Outline

- start with some initial drawing
- compute equilibrium layout

General Graphs – Outline

- start with some initial drawing
- compute equilibrium layout
- initialize frame F around drawing

General Graphs – Outline

- start with some initial drawing
- compute equilibrium layout
- initialize frame F around drawing
- while F too large:

General Graphs – Outline

- start with some initial drawing
- compute equilibrium layout
- initialize frame F around drawing
- while F too large:
  - shrink F
  - find new equilibrium

General Graphs – Outline

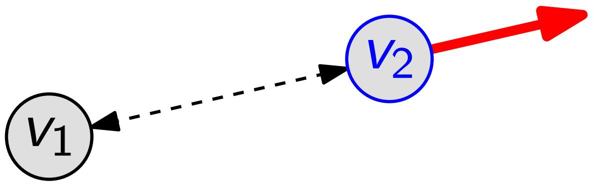
- start with some initial drawing
- compute equilibrium layout
- initialize frame F around drawing
- while F too large:
  - shrink F
  - find new equilibrium
  - if necessary remove vertices/edges

General Graphs – Outline

- start with some initial drawing
- compute equilibrium layout
- initialize frame F around drawing
- while F too large:
  - shrink F
  - find new equilibrium
  - if necessary remove vertices/edges
- postprocessing: transform straight-line edges to Bézier curves

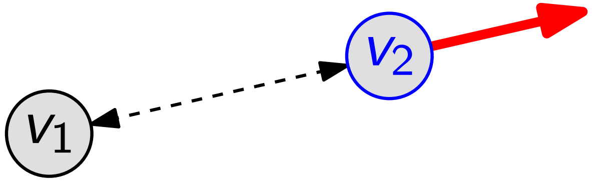
Forces

- repulsion between vertices

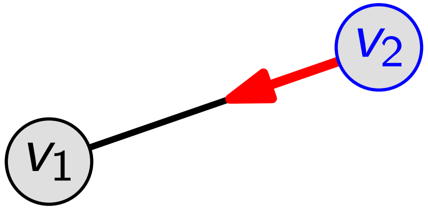


Forces

- repulsion between vertices

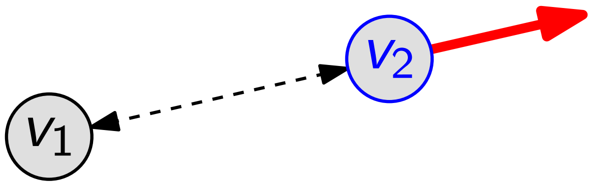


- attraction between *adjacent* vertices

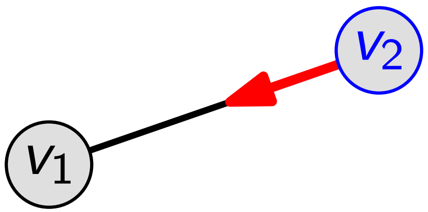


Forces

- repulsion between vertices



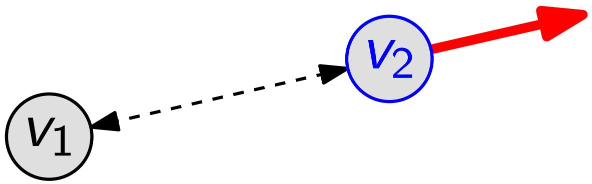
- attraction between *adjacent* vertices



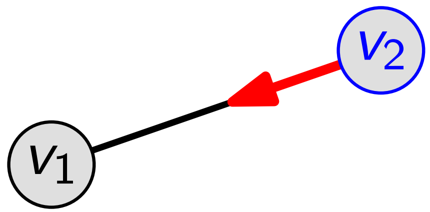
[Fruchterman, Reingold 1991]
parameterized by
desired edge length

Forces

- repulsion between vertices

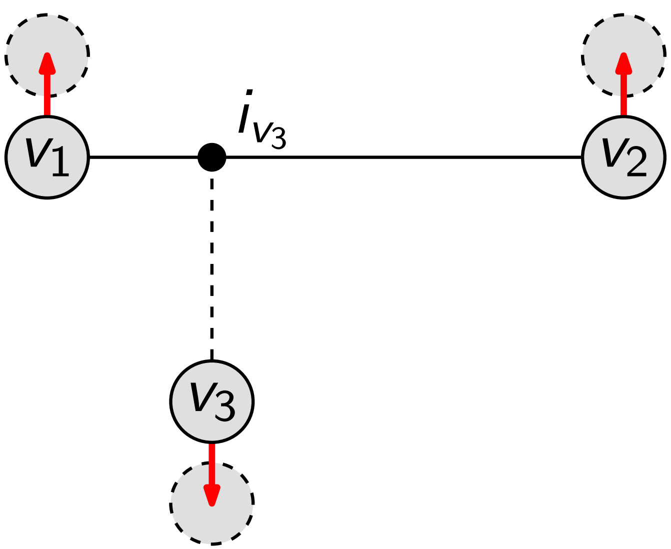


- attraction between *adjacent* vertices



[Fruchterman, Reingold 1991] parameterized by desired edge length

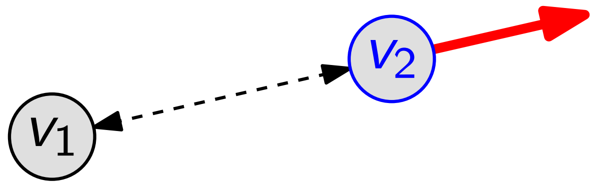
- edge-vertex repulsion



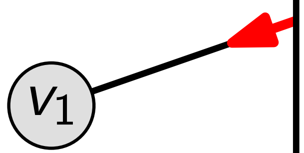
[Bertault, 2000]

Forces

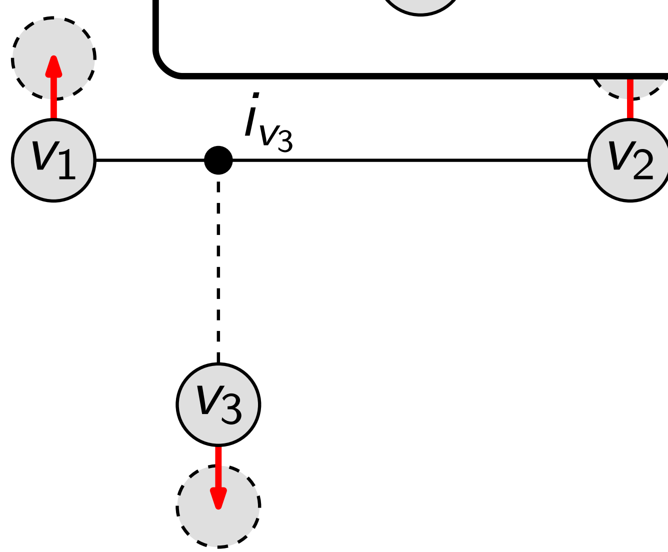
- repulsion between vertices



- attraction between a



- edge-vertex repulsio

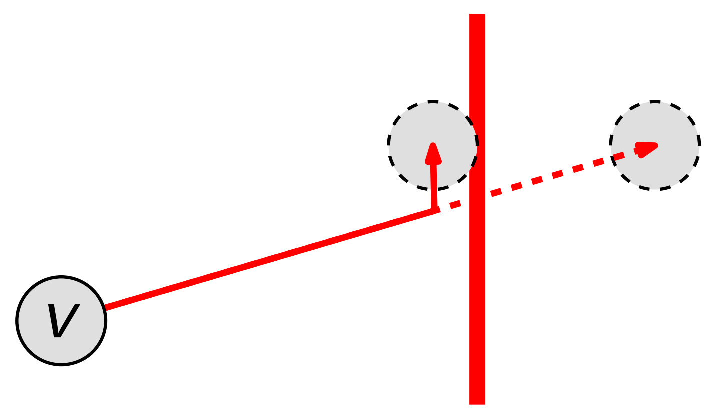


[Fruchterman, Reingold 1991]

parameterized by

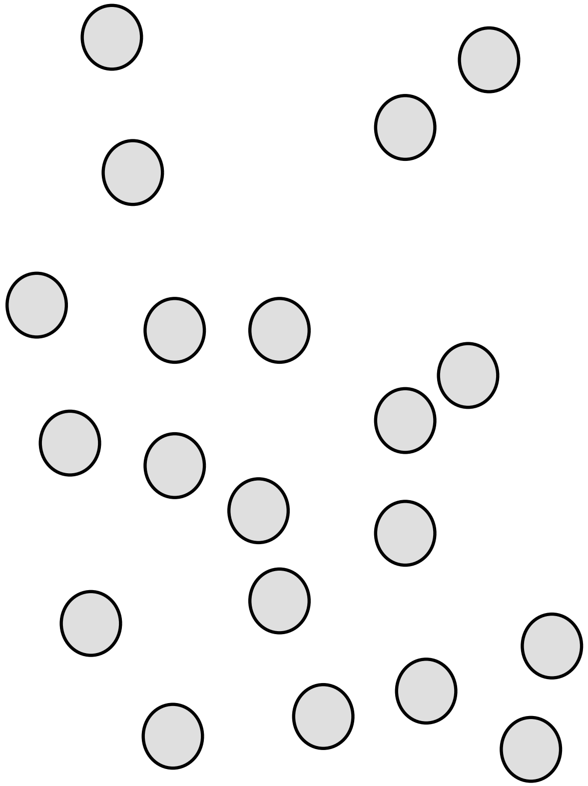
gth

frame boundary: cut off movement

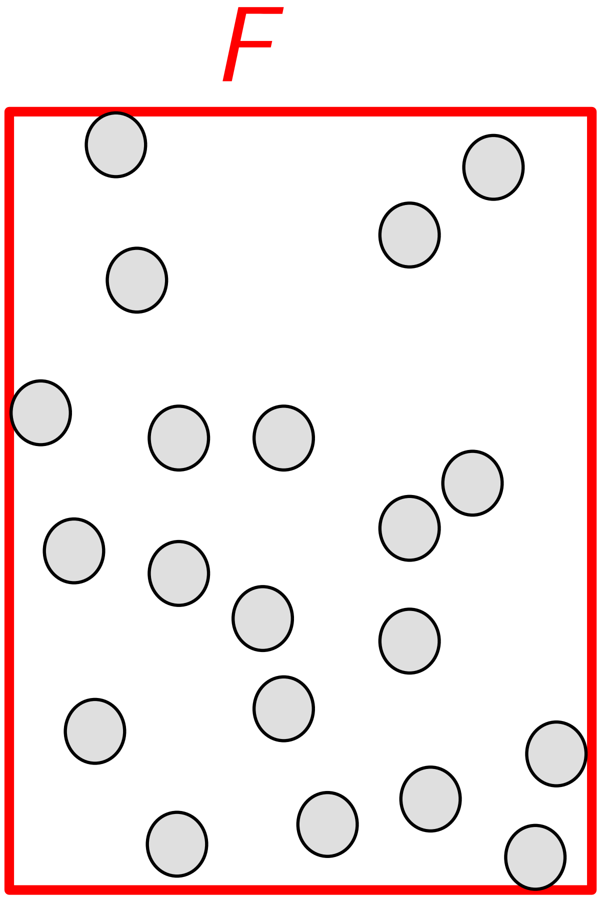


[Bertault, 2000]

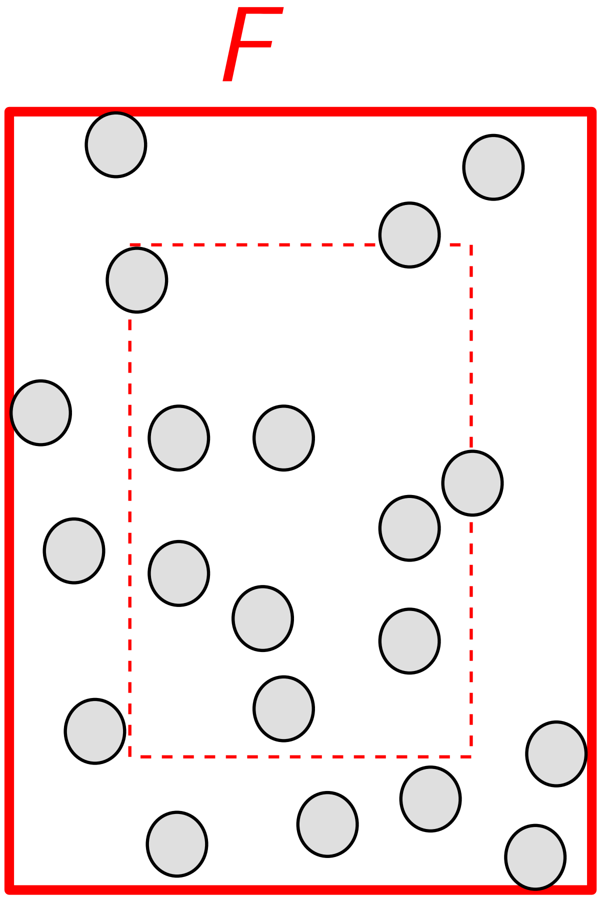
Handling the Frame



Handling the Frame

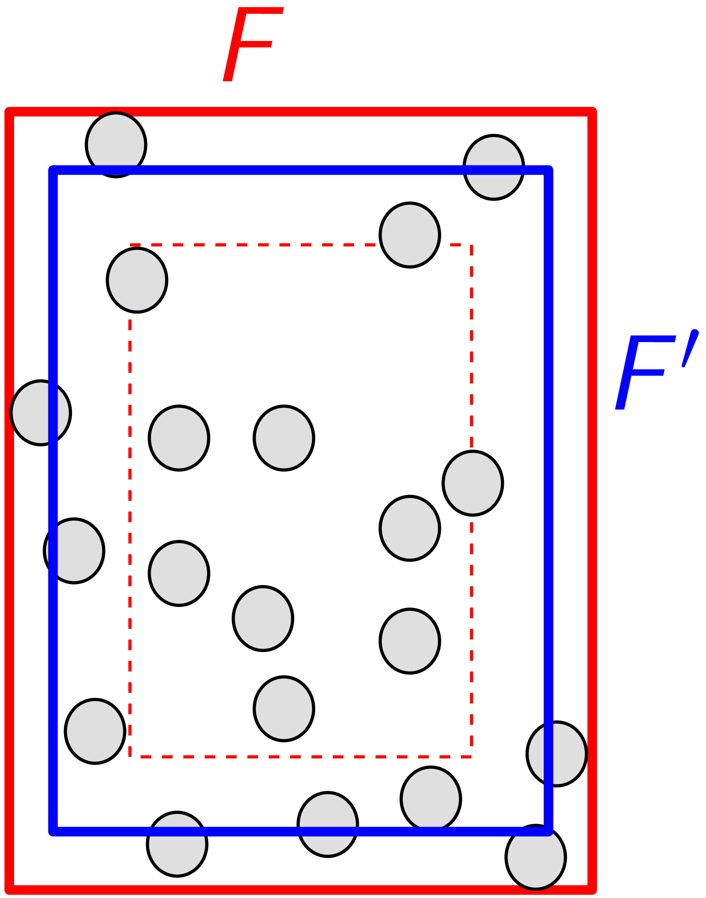


Handling the Frame



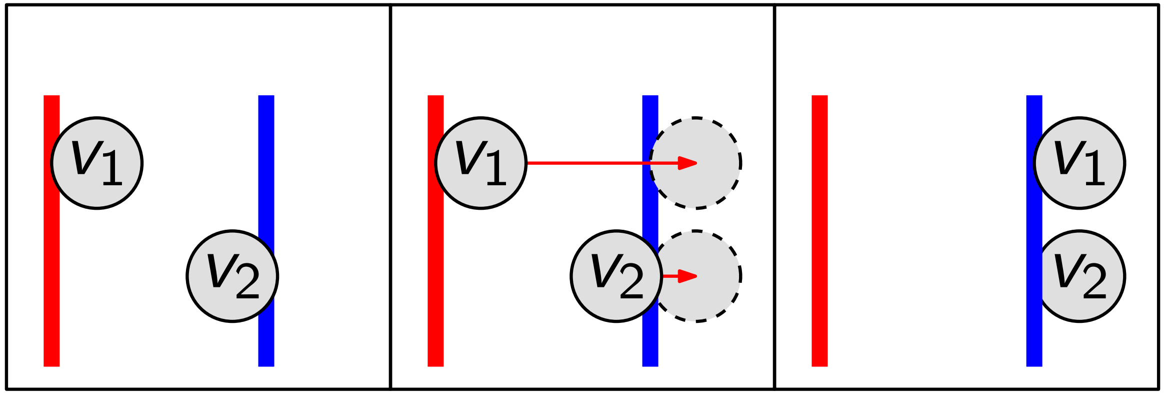
Handling the Frame

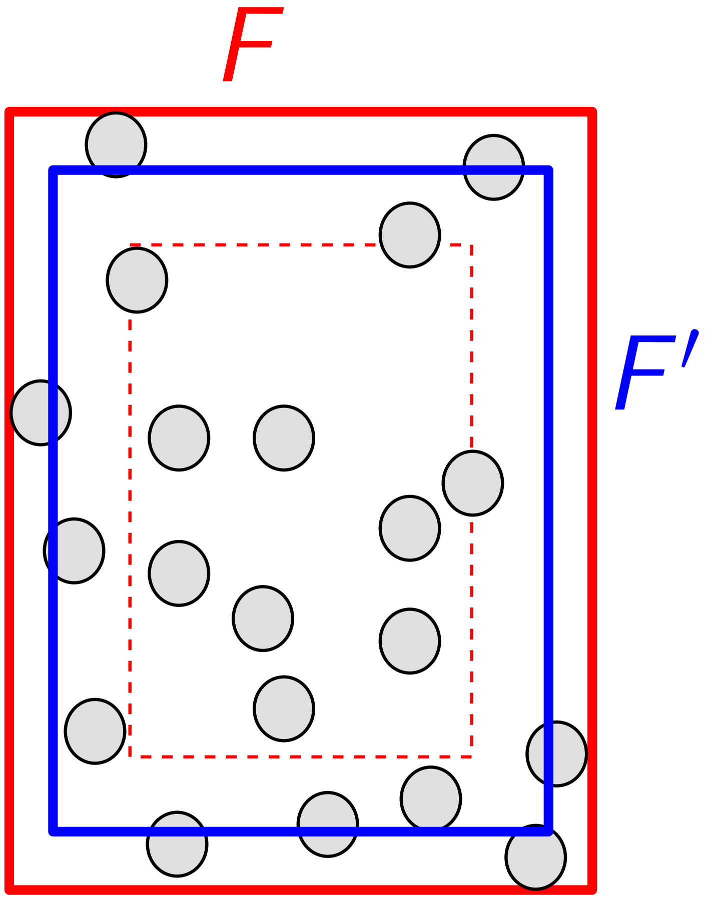
- iteratively shrink frame



Handling the Frame

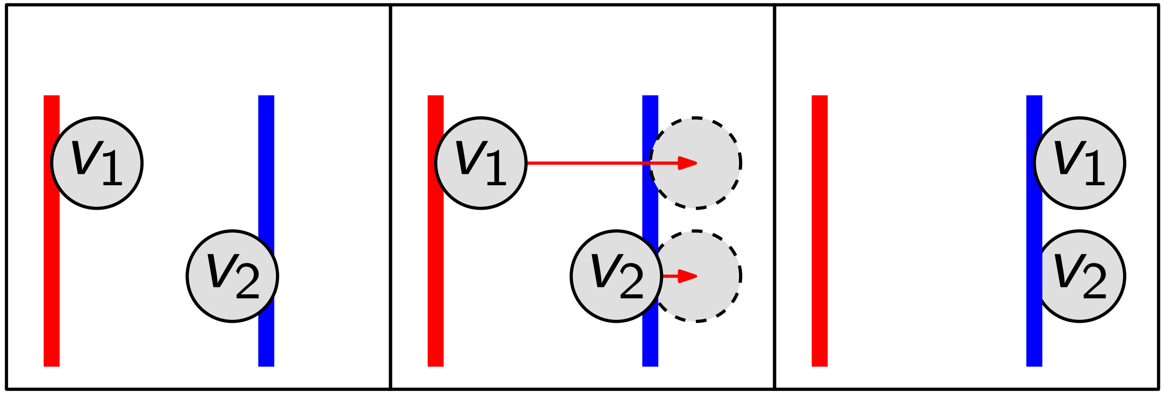
- iteratively shrink frame
- push vertices into frame

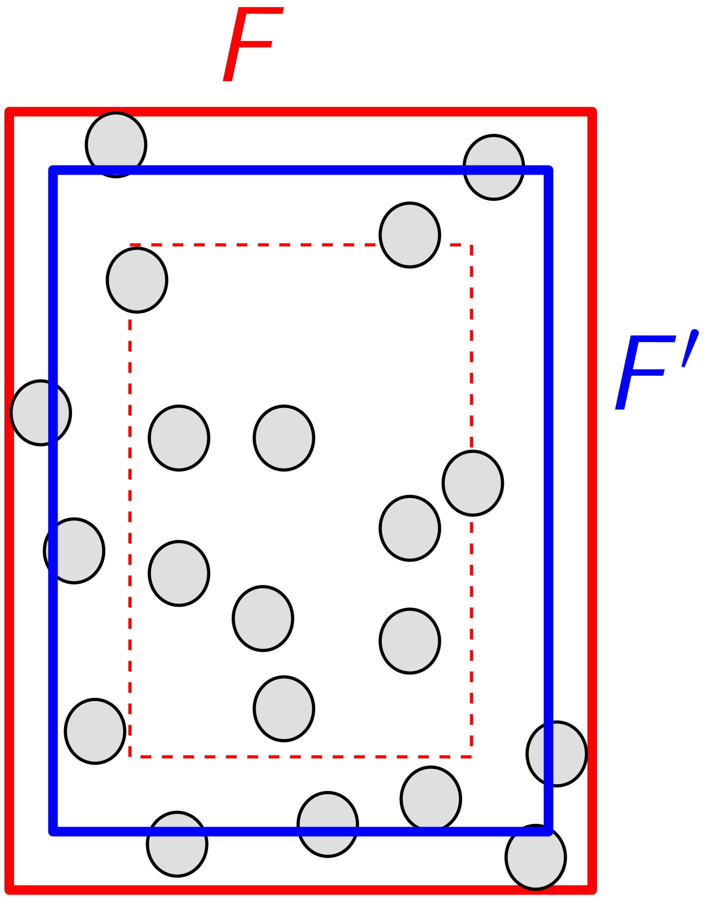




Handling the Frame

- iteratively shrink frame
- push vertices into frame
- compute new equilibrium layout

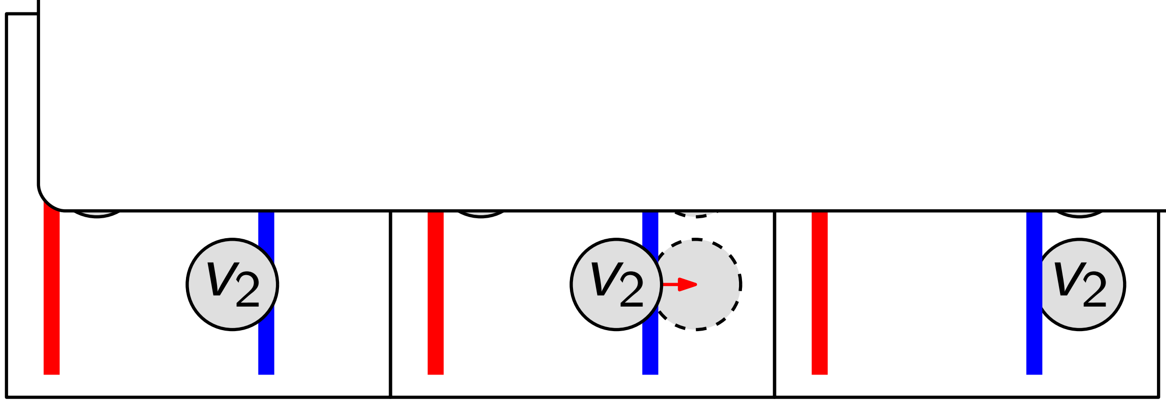


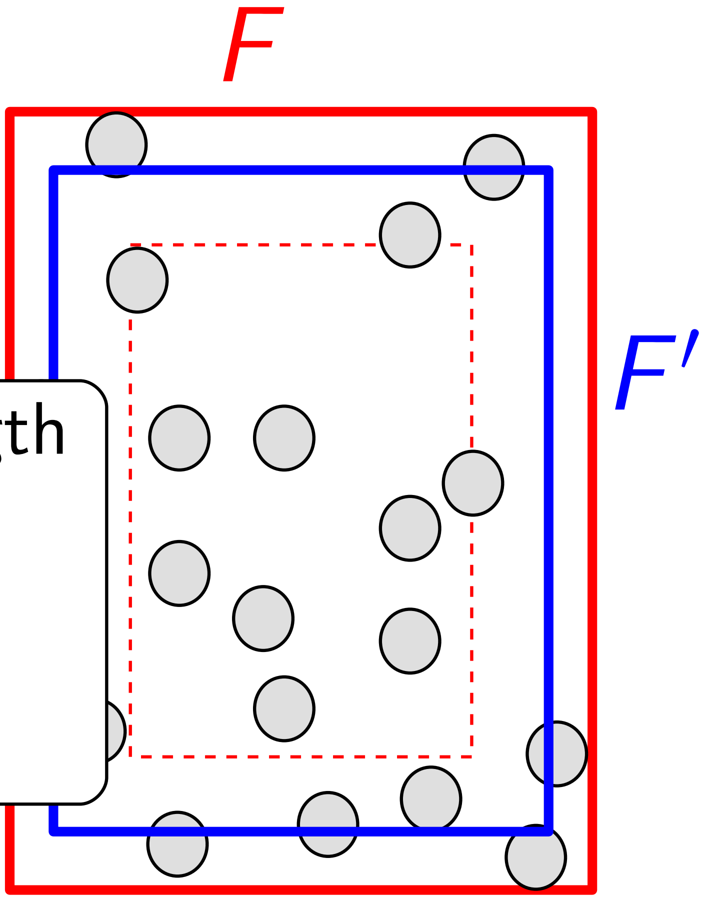


Handling the Frame

- iteratively shrink frame
- push vertices into frame
- compute new equilibrium layout
- remove vertices/edges if *necessary*

distances too short ↔ desired edge length



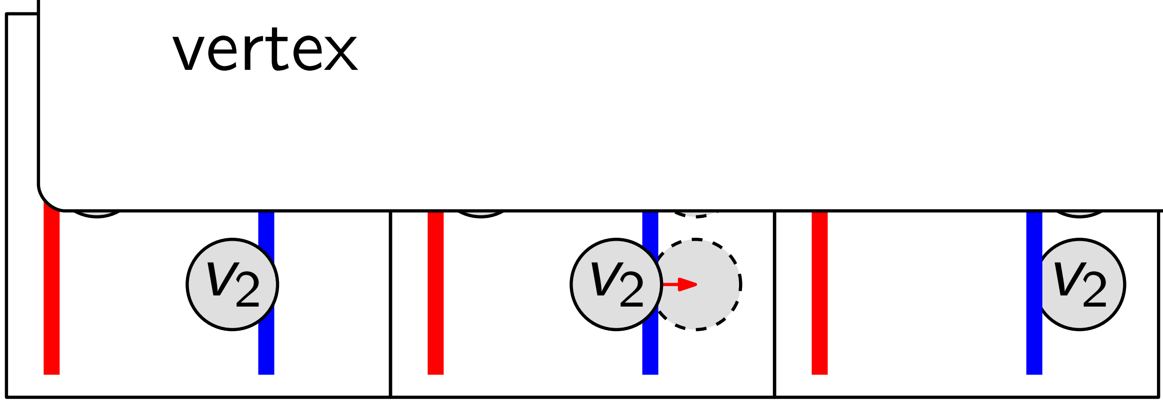


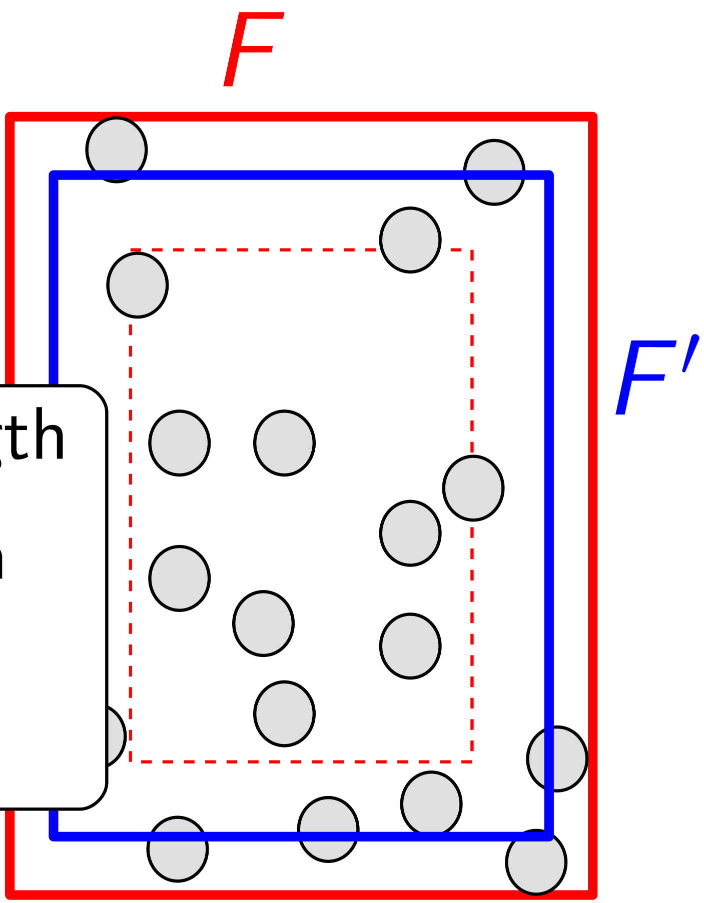
Handling the Frame

- iteratively shrink frame
- push vertices into frame
- compute new equilibrium layout
- remove vertices/edges if *necessary*

distances too short ↔ desired edge length

- average edge length small: remove a vertex



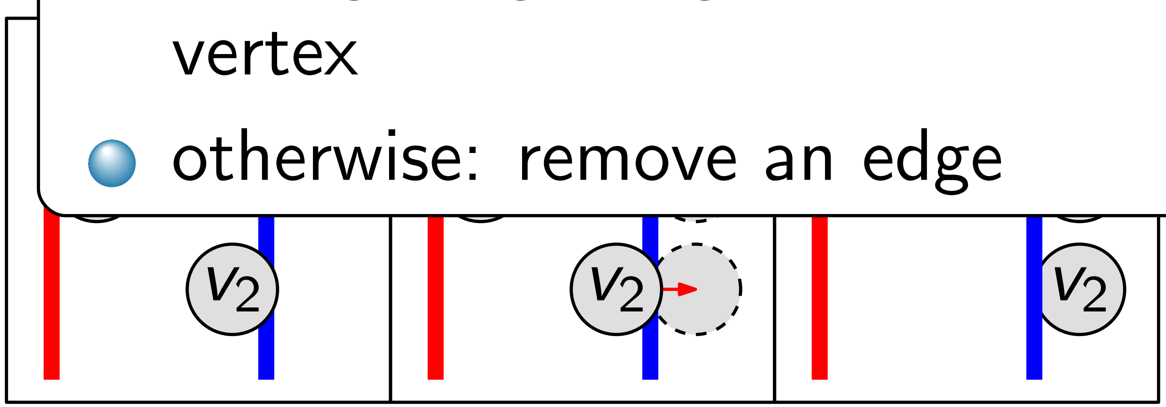


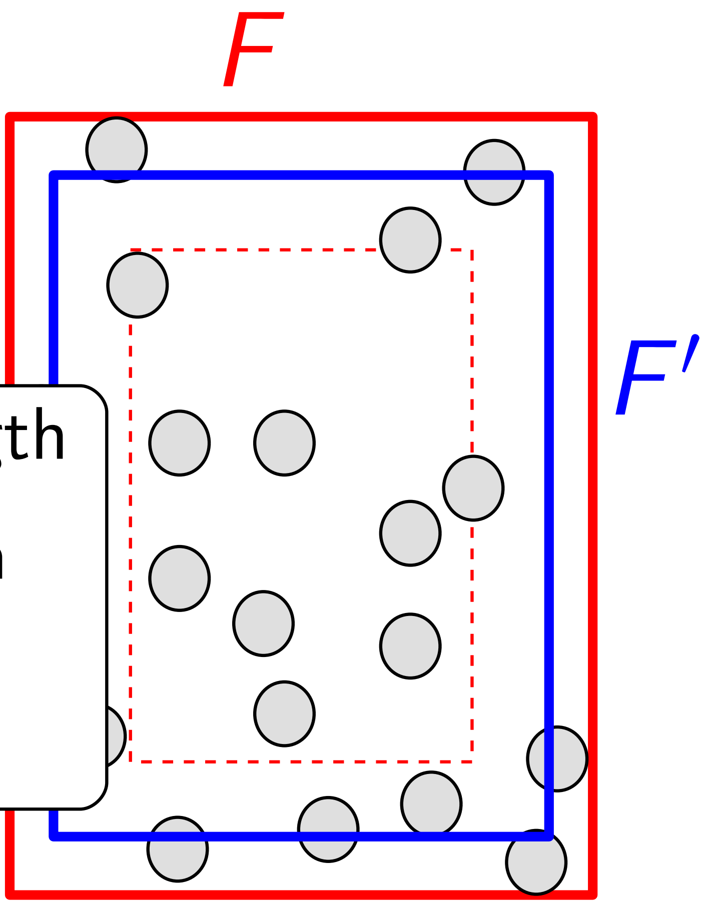
Handling the Frame

- iteratively shrink frame
- push vertices into frame
- compute new equilibrium layout
- remove vertices/edges if *necessary*

distances too short ↔ desired edge length

- average edge length small: remove a vertex
- otherwise: remove an edge





Removing vertices – Pressure and Stress

- remove lightest vertex
- remove vertex with worst weight-area ratio

Removing vertices – Pressure and Stress

- remove lightest vertex ✗
- remove vertex with worst weight-area ratio

Removing vertices – Pressure and Stress

- remove lightest vertex ✗
- remove vertex with worst weight-area ratio ✗

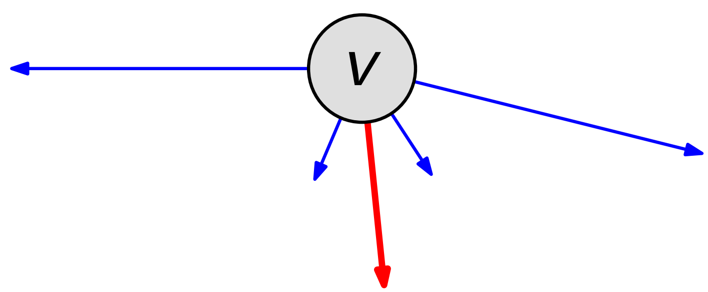
Removing vertices – Pressure and Stress

- remove lightest vertex ✗
- remove vertex with worst weight-area ratio ✗
- also take current drawing into account:
  compute pressure based on forces

Removing vertices – Pressure and Stress

- remove lightest vertex ✗
- remove vertex with worst weight-area ratio ✗
- also take current drawing into account:
  compute pressure based on forces

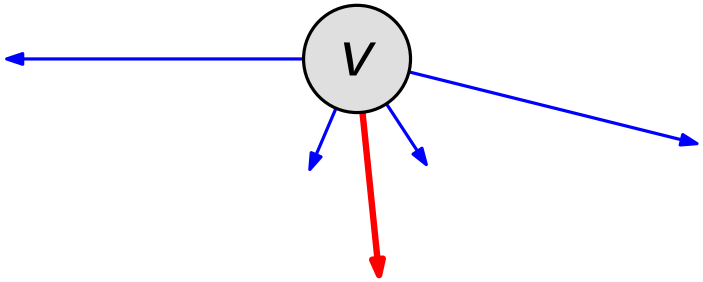
idea:



Removing vertices – Pressure and Stress

- remove lightest vertex ✗
- remove vertex with worst weight-area ratio ✗
- also take current drawing into account:
  compute pressure based on forces

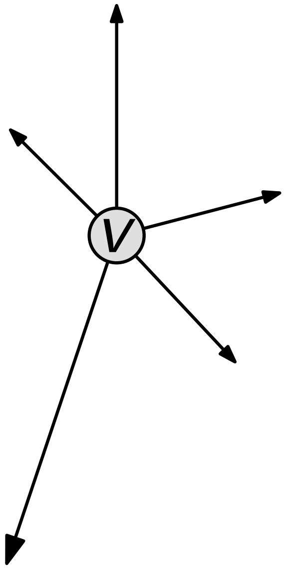
idea: forces in opposite directions



pressure $P(v)$

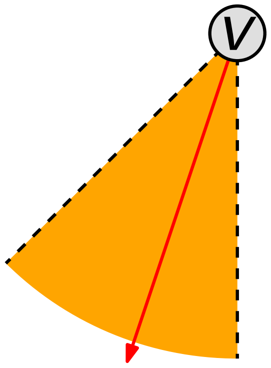
Removing vertices – Pressure and Stress

- remove lightest vertex ✗
- remove vertex with worst weight-area ratio ✗
- also take current drawing into account: compute pressure based on forces



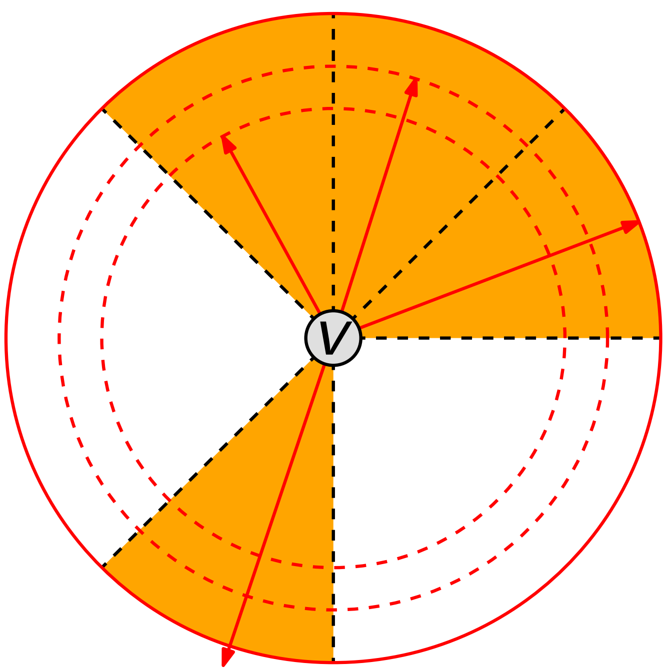
Removing vertices – Pressure and Stress

- remove lightest vertex ✗
- remove vertex with worst weight-area ratio ✗
- also take current drawing into account:
  compute pressure based on forces



Removing vertices – Pressure and Stress

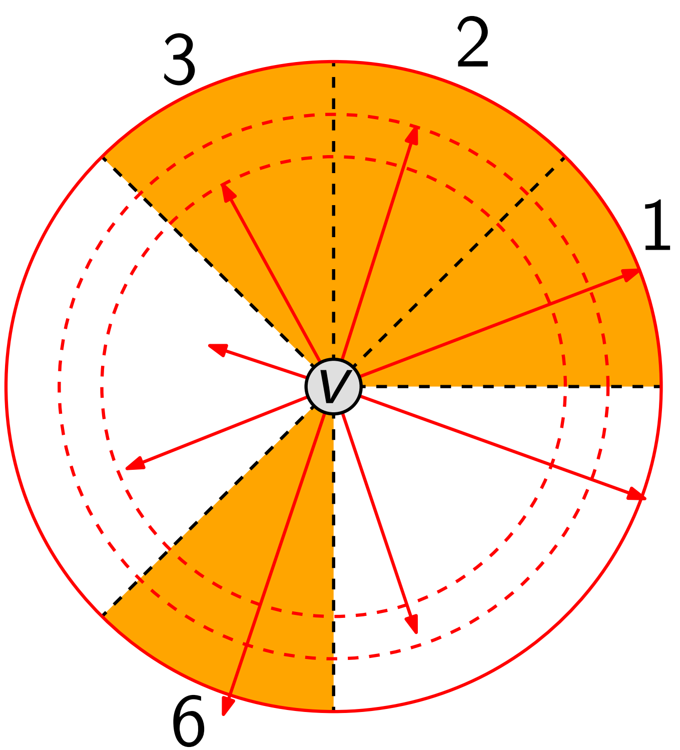
- remove lightest vertex ✗
- remove vertex with worst weight-area ratio ✗
- also take current drawing into account:
  compute pressure based on forces



– compare with opposite octants

Removing vertices – Pressure and Stress

- remove lightest vertex ✗
- remove vertex with worst weight-area ratio ✗
- also take current drawing into account:
  compute pressure based on forces



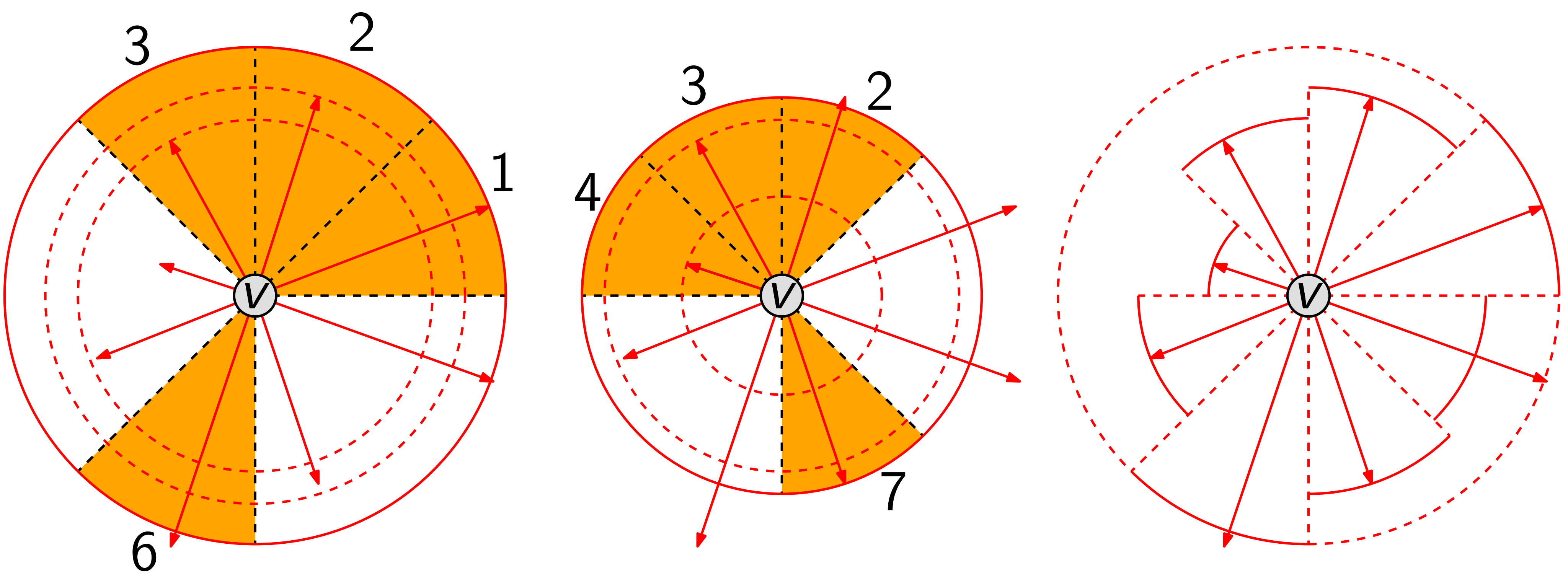
– compare with opposite octants

– worst pair $\rightarrow$ pressure

Removing vertices – Pressure and Stress

- remove lightest vertex ✗
- remove vertex with worst weight-area ratio ✗

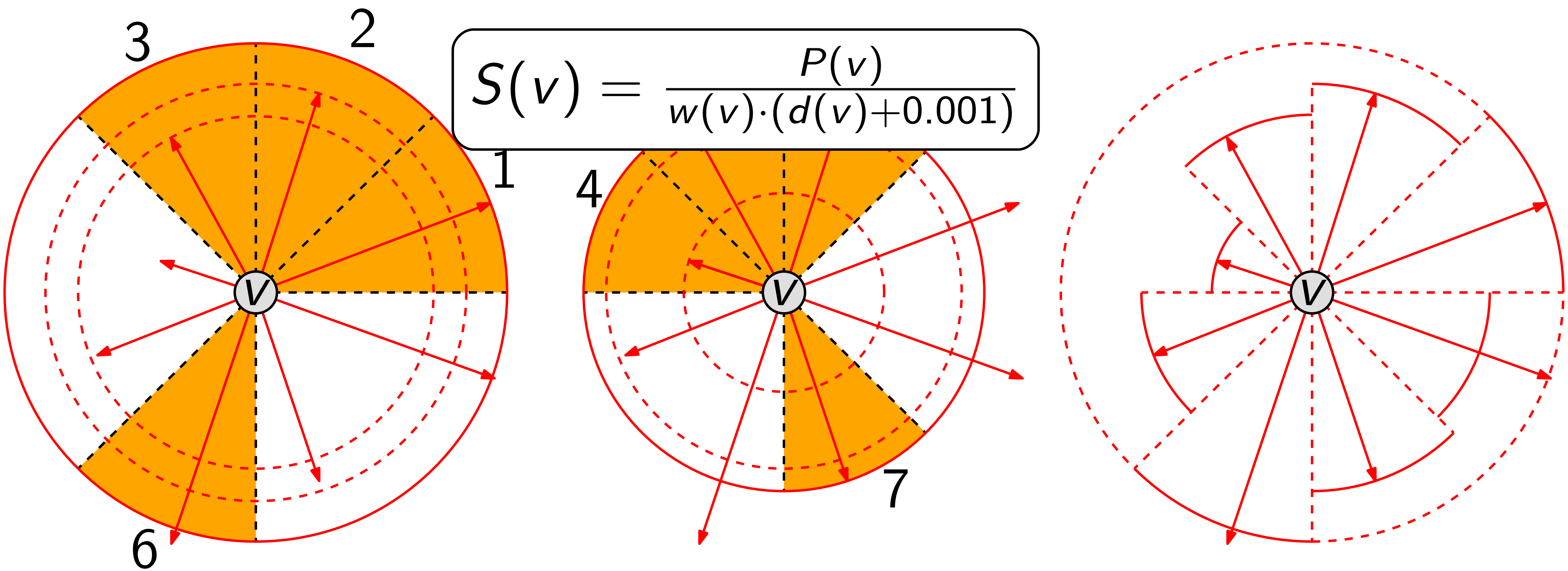
- also take current drawing into account:
  compute pressure based on forces



Removing vertices – Pressure and Stress

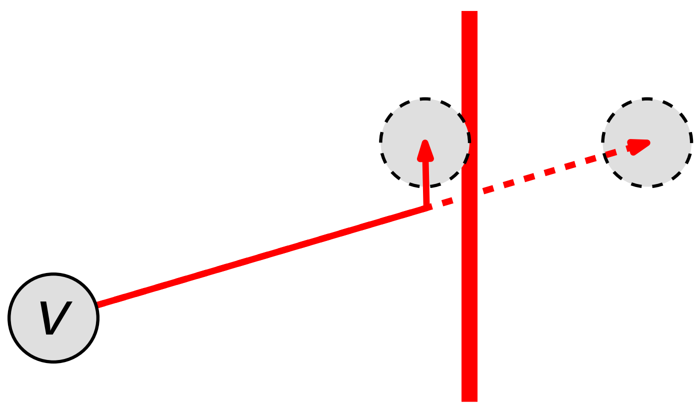
- remove lightest vertex ✗
- remove vertex with worst weight-area ratio ✗
- also take current drawing into account:
  compute pressure based on forces
- remove vertex with highest stress value

$$S(v) = \frac{P(v)}{w(v)\cdot(d(v)+0.001)}$$



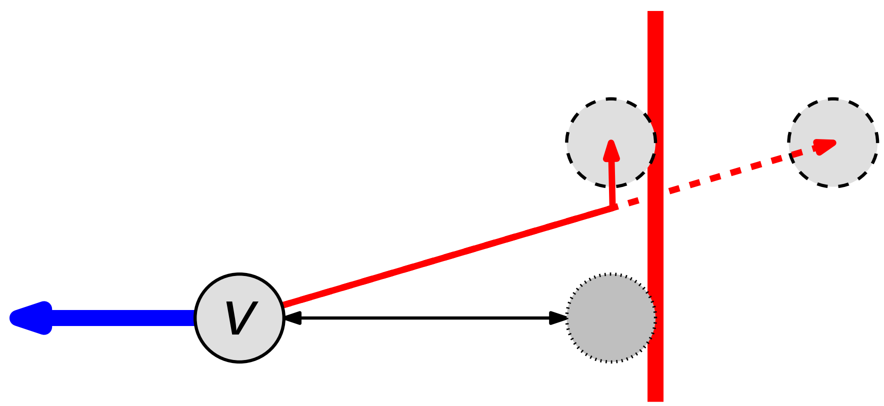
Remark: Boundary Vertices

- movement cut off



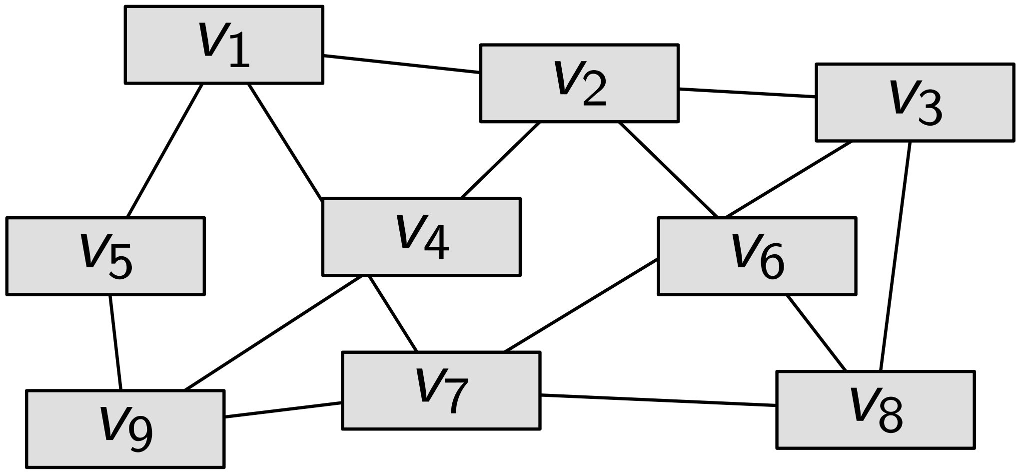
Remark: Boundary Vertices

- movement cut off
- model as force
- only considered for pressure computation



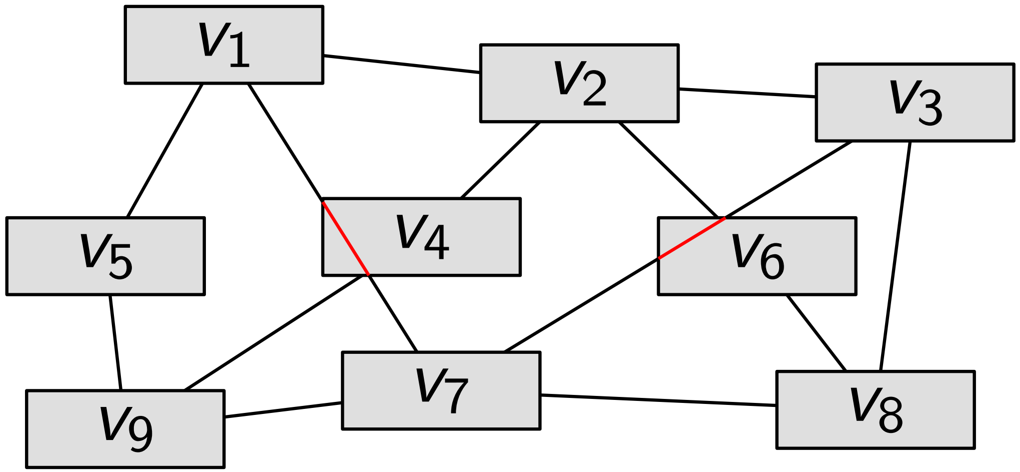
Postprocessing: Curved Edges

- avoid edge-vertex intersections



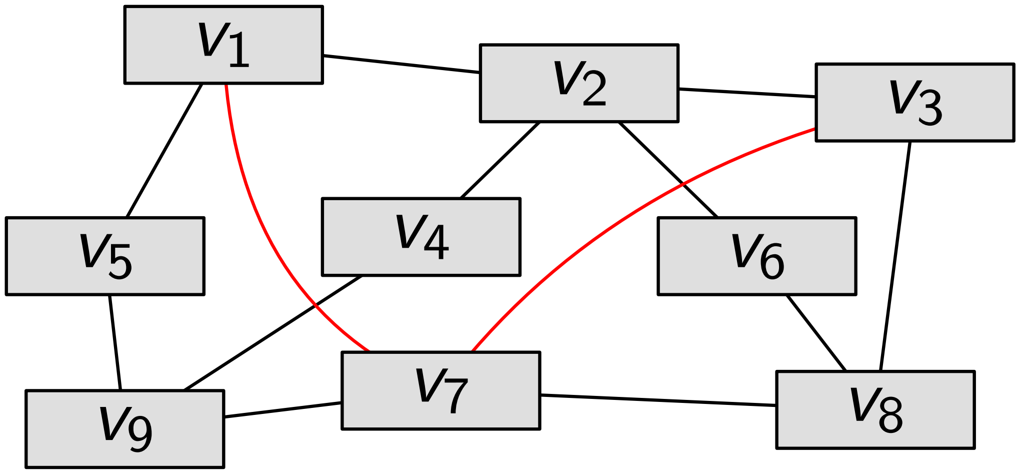
Postprocessing: Curved Edges

- avoid edge-vertex intersections



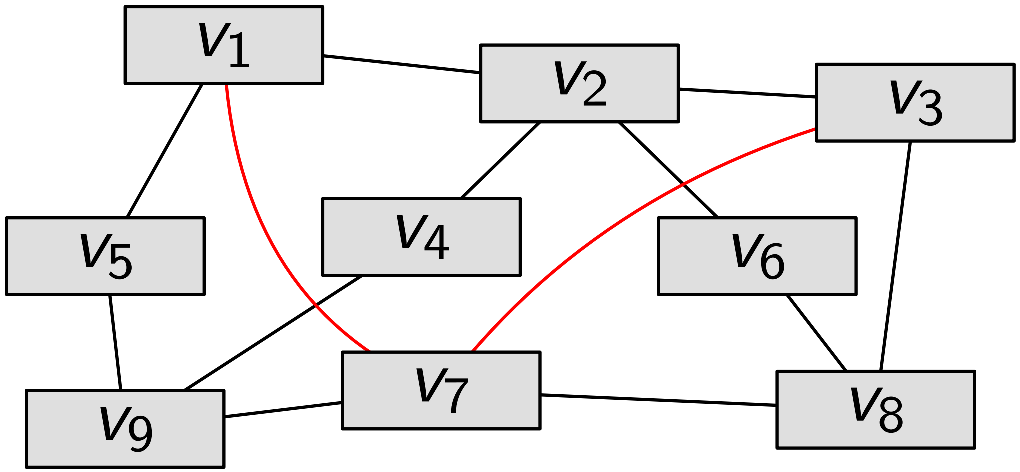
Postprocessing: Curved Edges

- avoid edge-vertex intersections

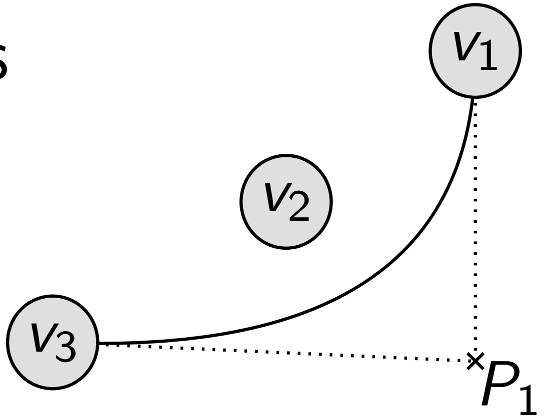


Postprocessing: Curved Edges

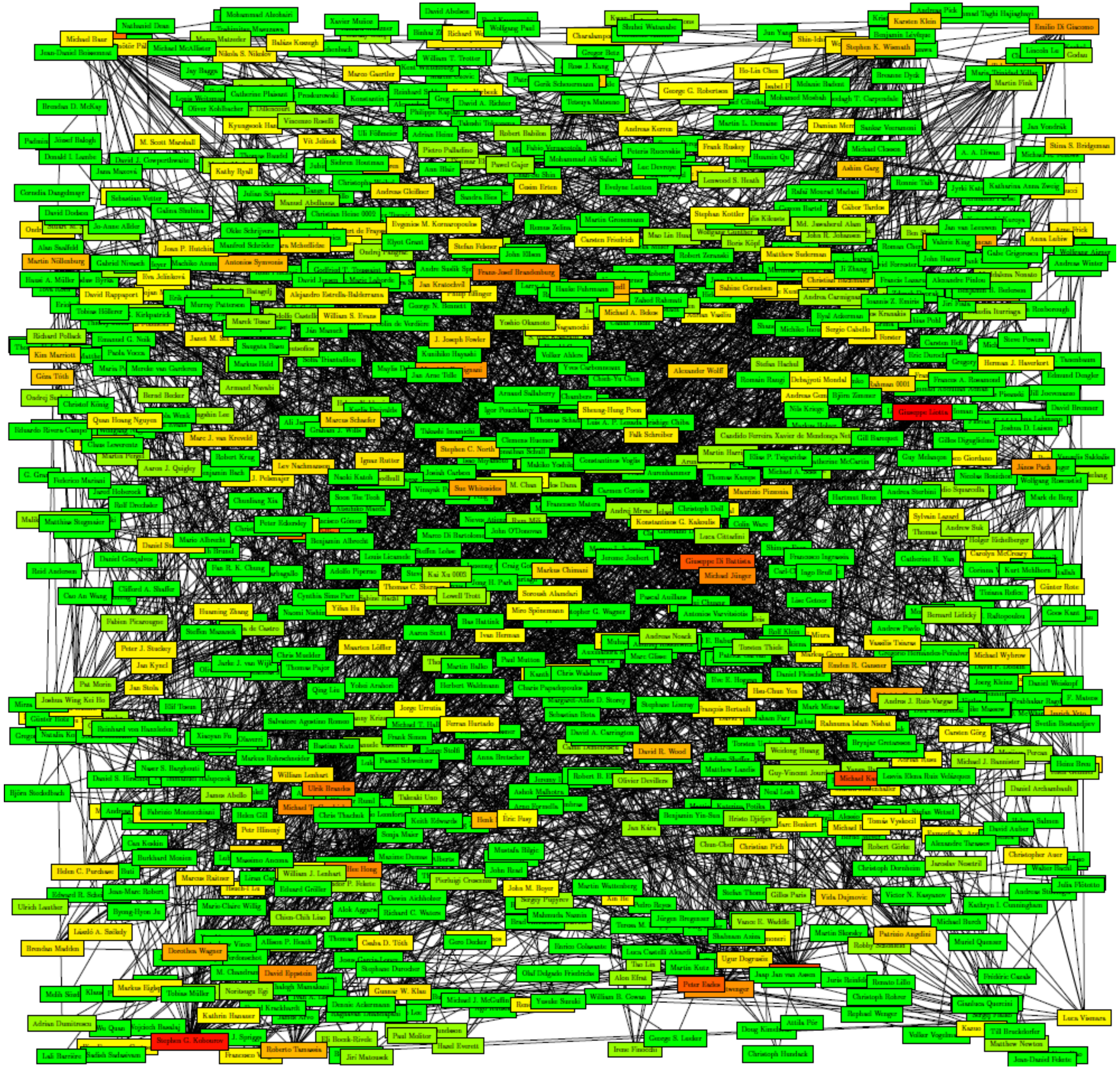
- avoid edge-vertex intersections



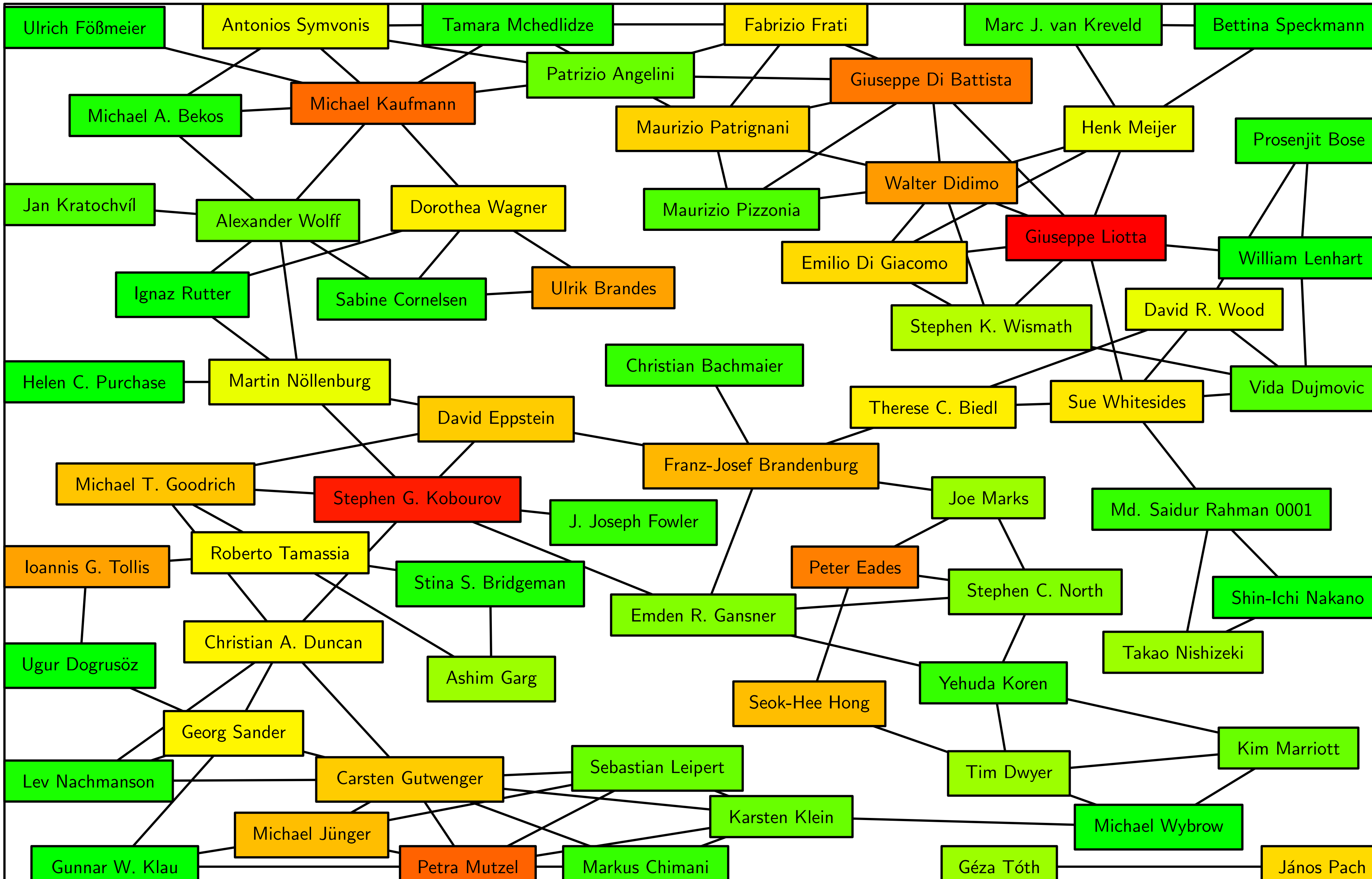
- draw some edges as quadratic Bézier curves
- force-directed computation of curve



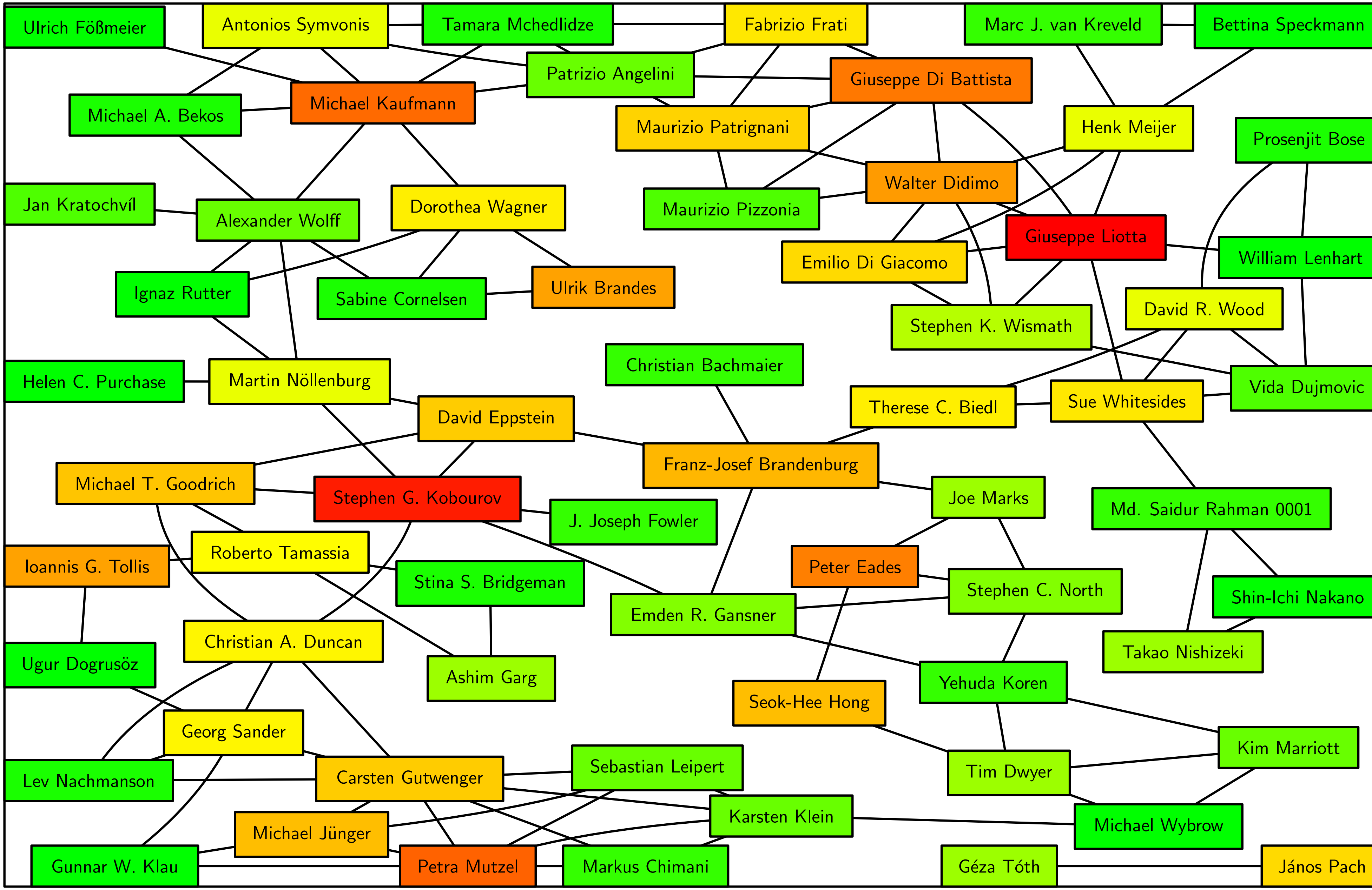
Example – Input



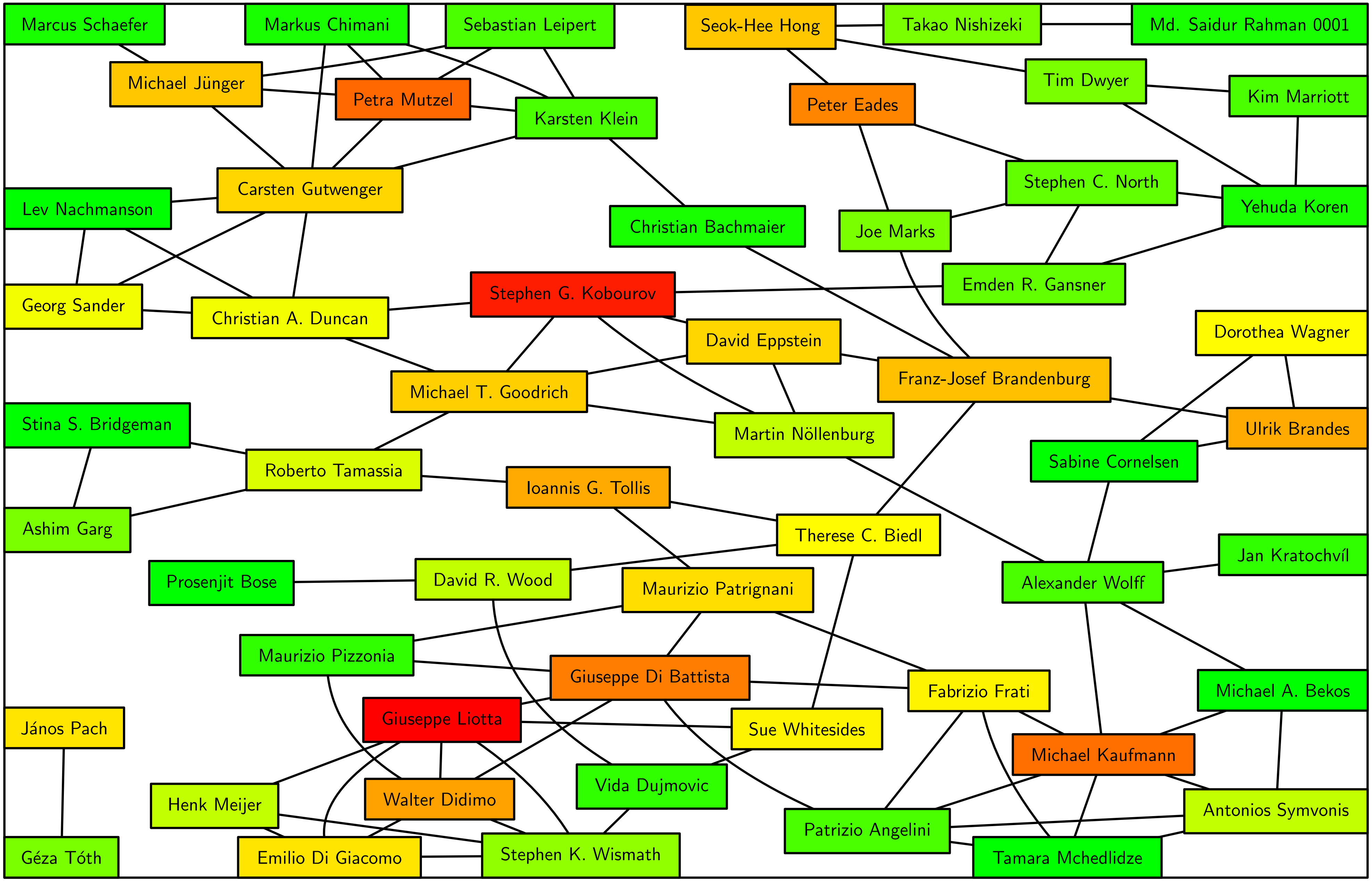
Example – Output



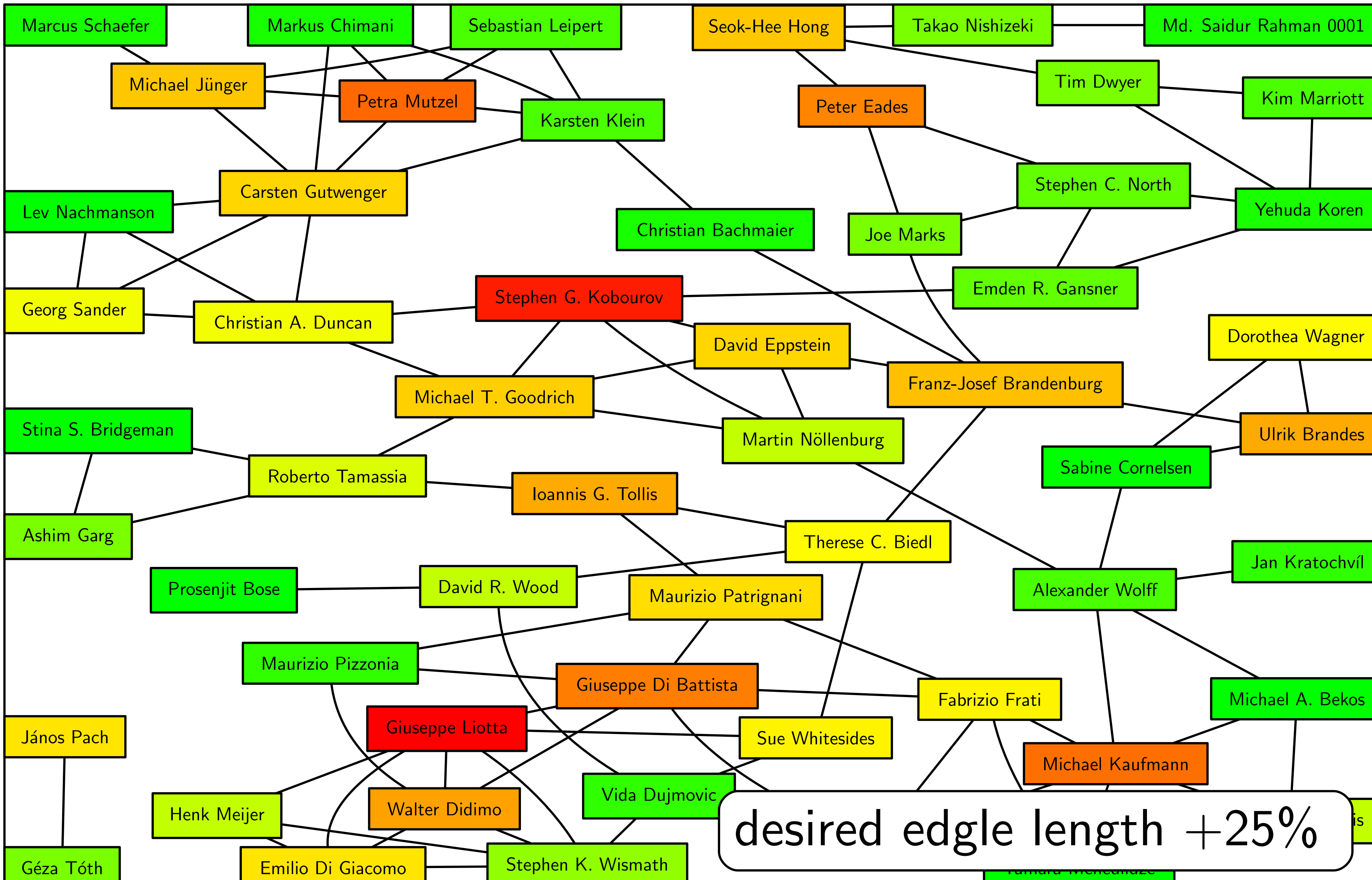
Example – Curved Edges



Example – Sparser Output



Example – Sparser Output



desired edgle length +25%

Remarks

- GD coauthor graph (1994 - 2012): 950, vertices 2559 edges $\rightarrow$ A4 paper, 10pt font:
  – runtime $\approx$ 2 minutes

Remarks

- GD coauthor graph (1994 - 2012): 950, vertices 2559 edges $\rightarrow$ A4 paper, 10pt font:
  – runtime $\approx$ 2 minutes
- preprocessing removing very light vertices yielded speedup $+$ heavier outputs

Remarks

- GD coauthor graph (1994 - 2012): 950, vertices 2559 edges $\rightarrow$ A4 paper, 10pt font:
  – runtime $\approx$ 2 minutes
- preprocessing removing very light vertices yielded speedup $+$ heavier outputs
- activating edge-vertex repulsion only at the end of force-directed phase:
  weight of edges in output $+80\%$

Conclusion

- new problem: find + draw heavy subgraph within prescribed area

Conclusion

- new problem: find + draw heavy subgraph within prescribed area
- heuristic for calculation graphs
- force-directed approach for general graphs

Conclusion

- new problem: find + draw heavy subgraph within prescribed area
- heuristic for calculation graphs
- force-directed approach for general graphs
- nice output drawings

Conclusion

- new problem: find + draw heavy subgraph within prescribed area
- heuristic for calculation graphs
- force-directed approach for general graphs
- nice output drawings
- Problems:
  - – no alternative methods to compare with: real performance?
  - – speedup