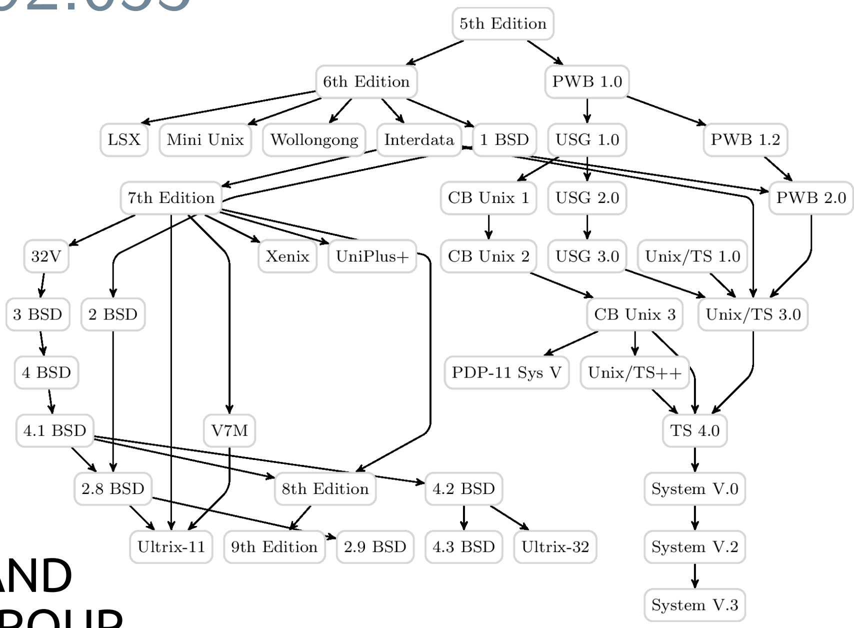
Layered Graph Drawing

Lecture Graph Drawing Algorithms · 192.053

Martin Nöllenburg

15.05.2018



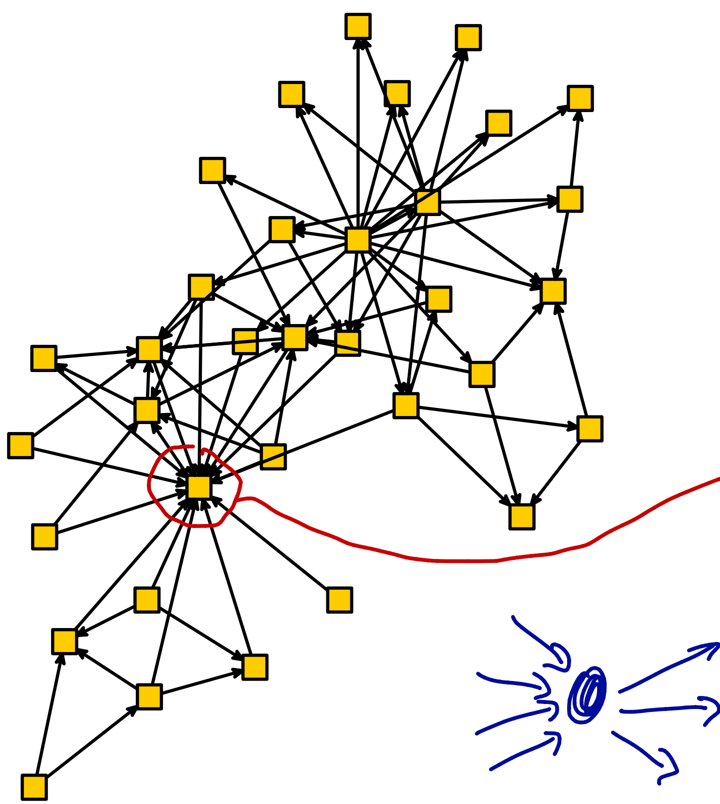
ALGORITHMS AND COMPLEXITY GROUP

Drawing Directed Graphs

Previously we studied mostly undirected graphs except for trees and series-parallel graphs. What if our input graph is directed but neither a tree nor series-parallel?

What is a suitable set of constraints and aesthetic criteria?

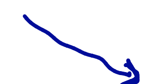
Assumptions: simple, directed graph



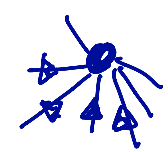
- straight edges
- few crossings
- large angles

also applies to undirected graphs

- general edge direction, e.g.



- arrow heads not at the end of edges

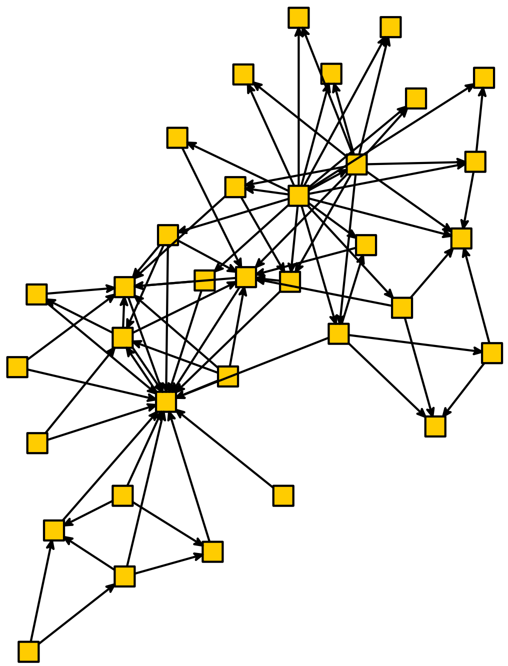


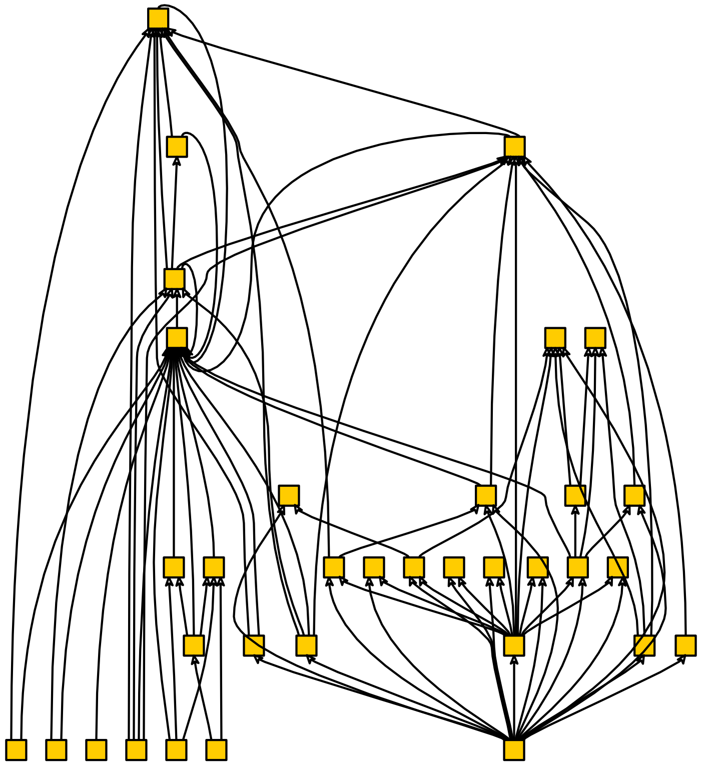
- identify and show structures like cycles ...

Layered Graph Layout

Input: directed graph $D = (V, A)$

Output: drawing of D that emphasizes its hierarchical structure





Layered Graph Layout

Input: directed graph $D = (V, A)$

Output: drawing of D that emphasizes its hierarchical structure

Criteria:

- many edges pointing upward (or some other direction)
- ideally short, straight and vertical edges
- vertices placed on (few) horizontal layers
- few edge crossings
- evenly distributed vertices

Layered Graph Layout

Input: directed graph $D = (V, A)$

Output: drawing of D that emphasizes its hierarchical structure

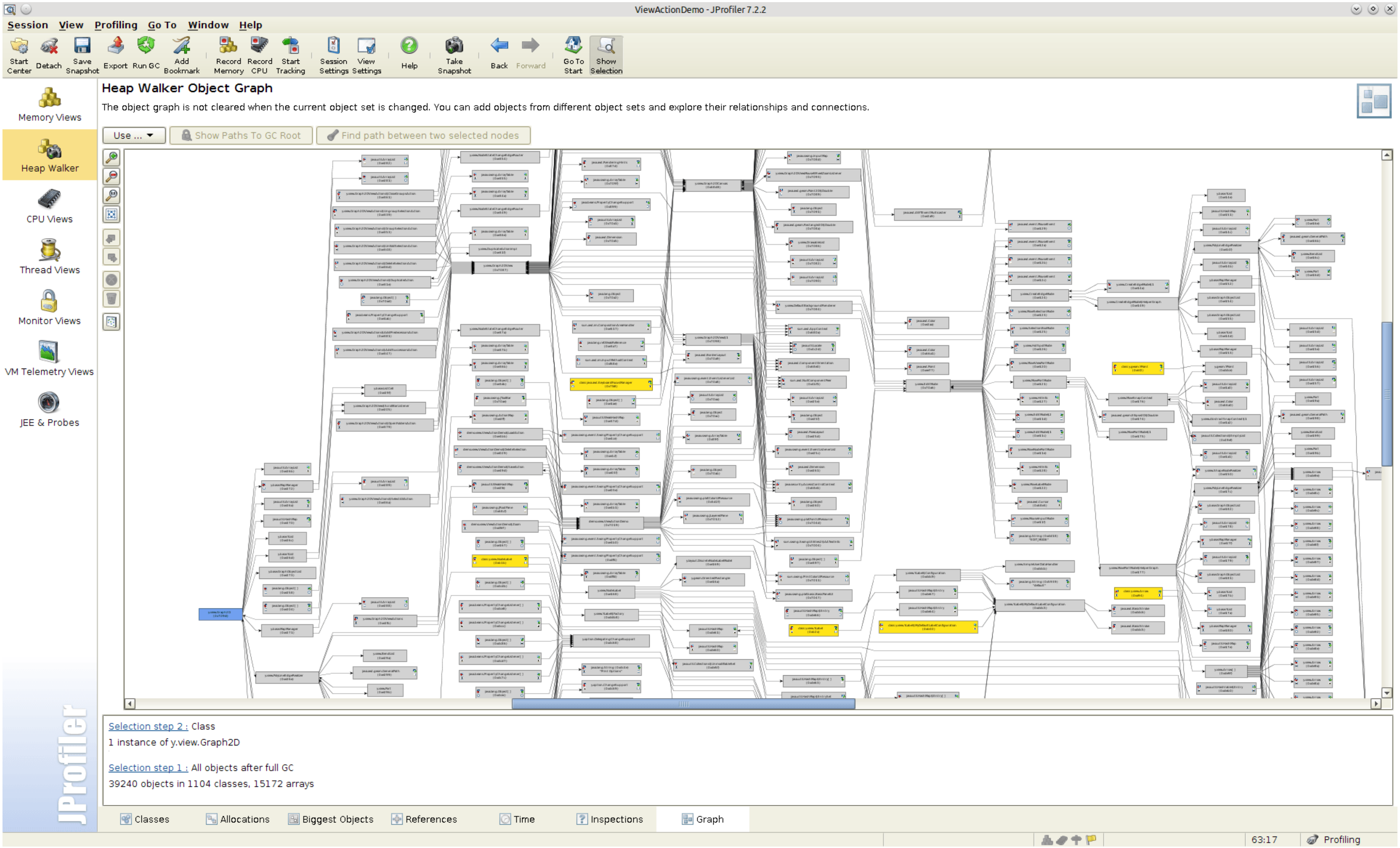
Criteria:

- many edges pointing upward (or some other direction)
- ideally short, straight and vertical edges
- vertices placed on (few) horizontal layers
- few edge crossings
- evenly distributed vertices

⚠ partially contradicting criteria

Examples

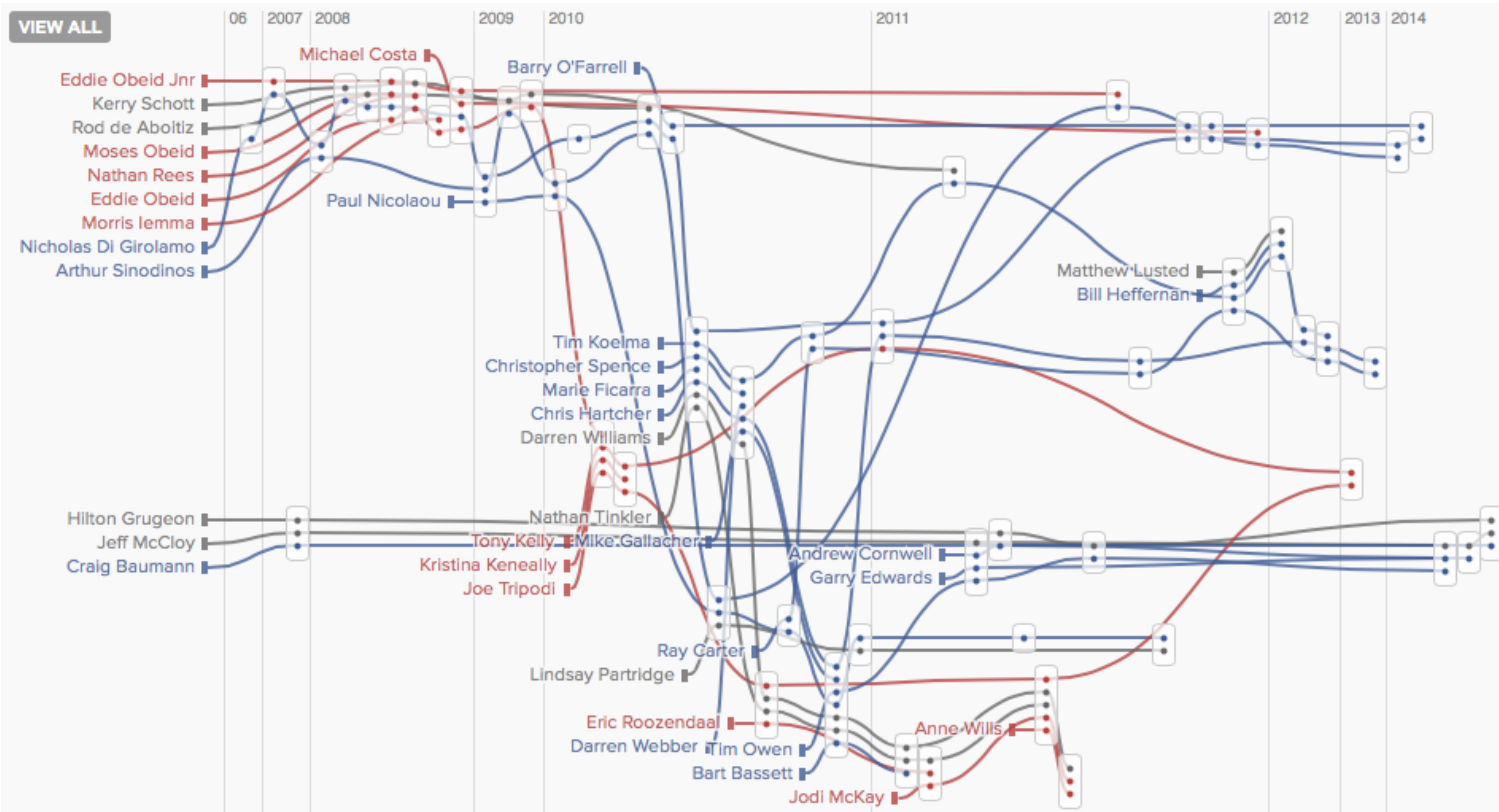
Java profiler



Java profiler JProfiler via yFiles

Examples

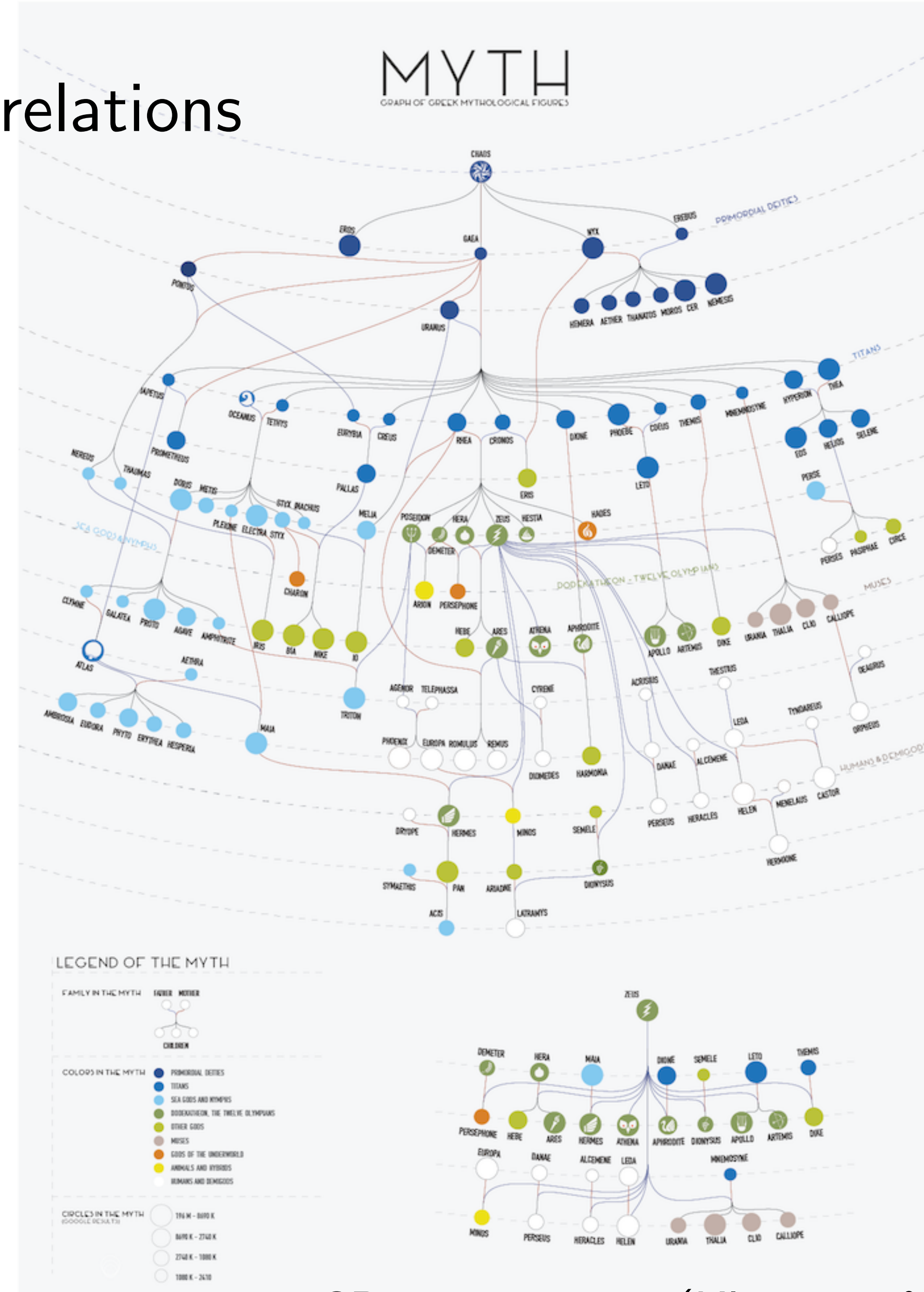
Storyline layout



Politics data visualization from ABC news, Australia

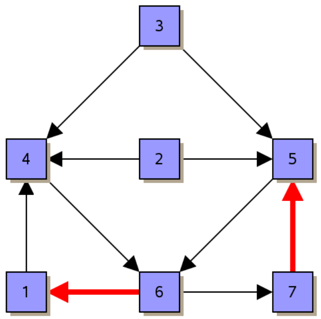
Examples

Greek mythology family relations



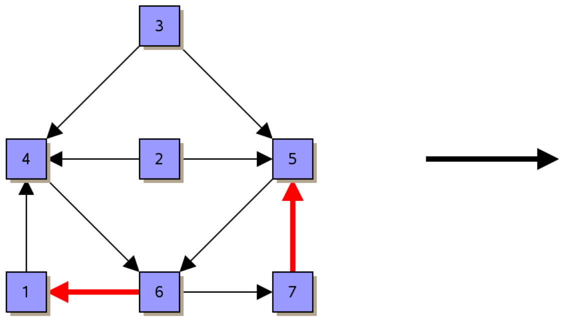
winner GD contest 2016 (Klawitter & Mchedlidze)

Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)

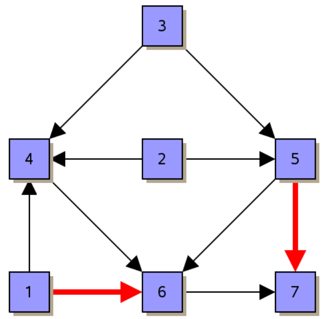


input

Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)

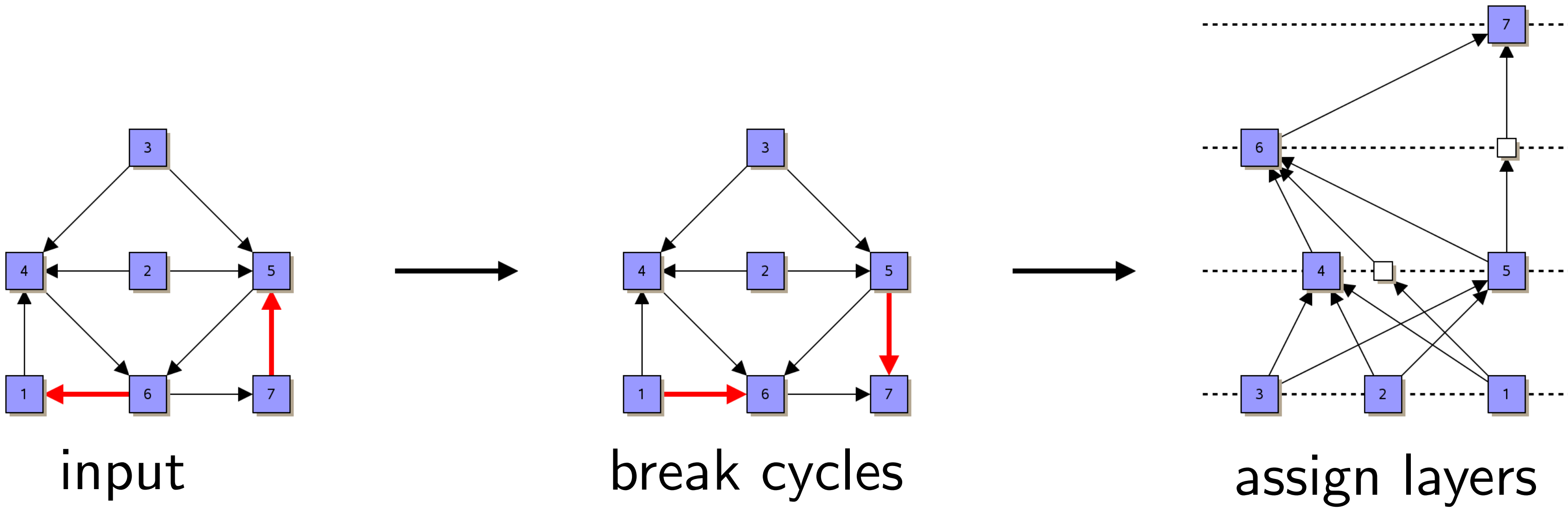


input

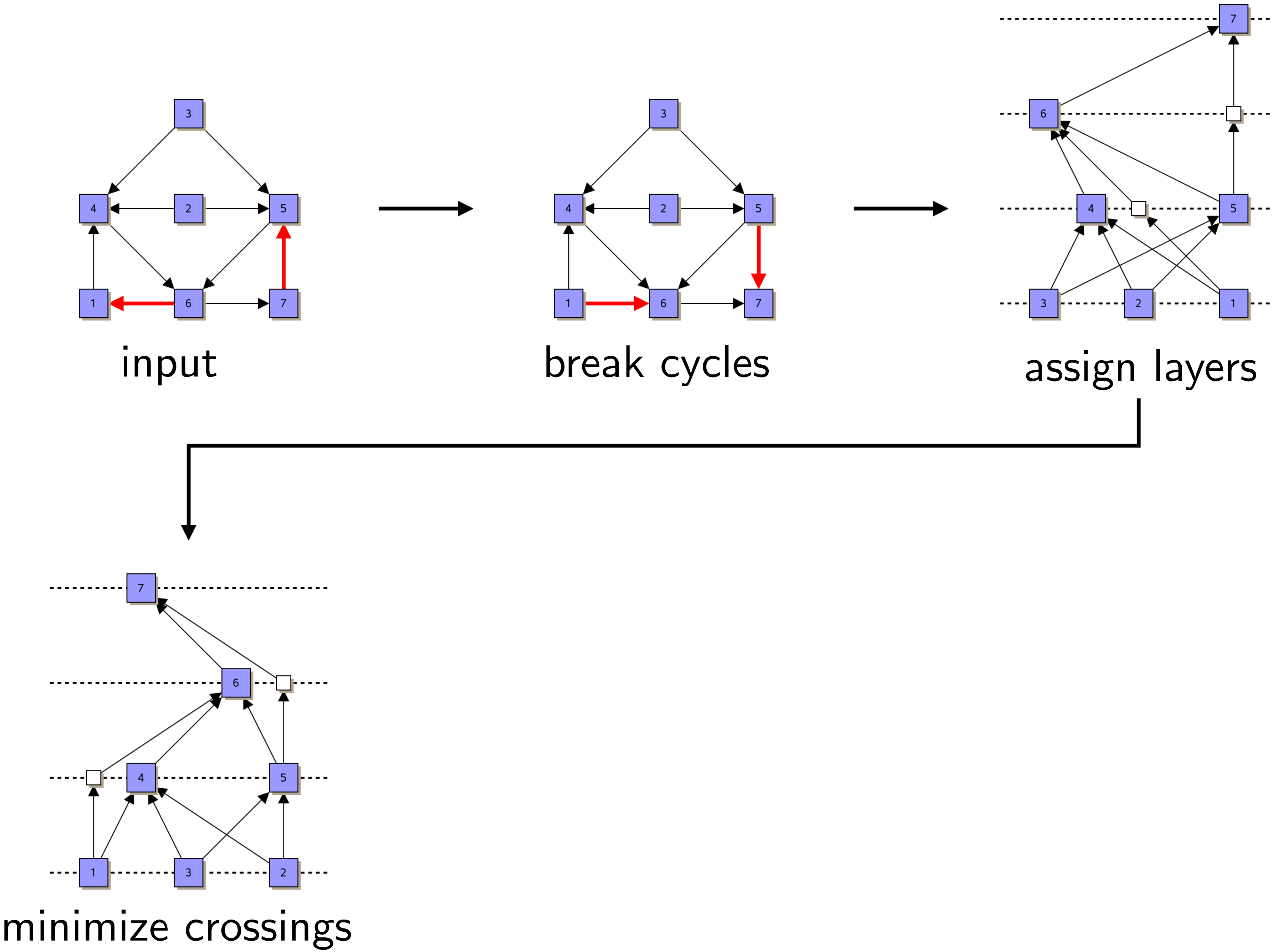


break cycles

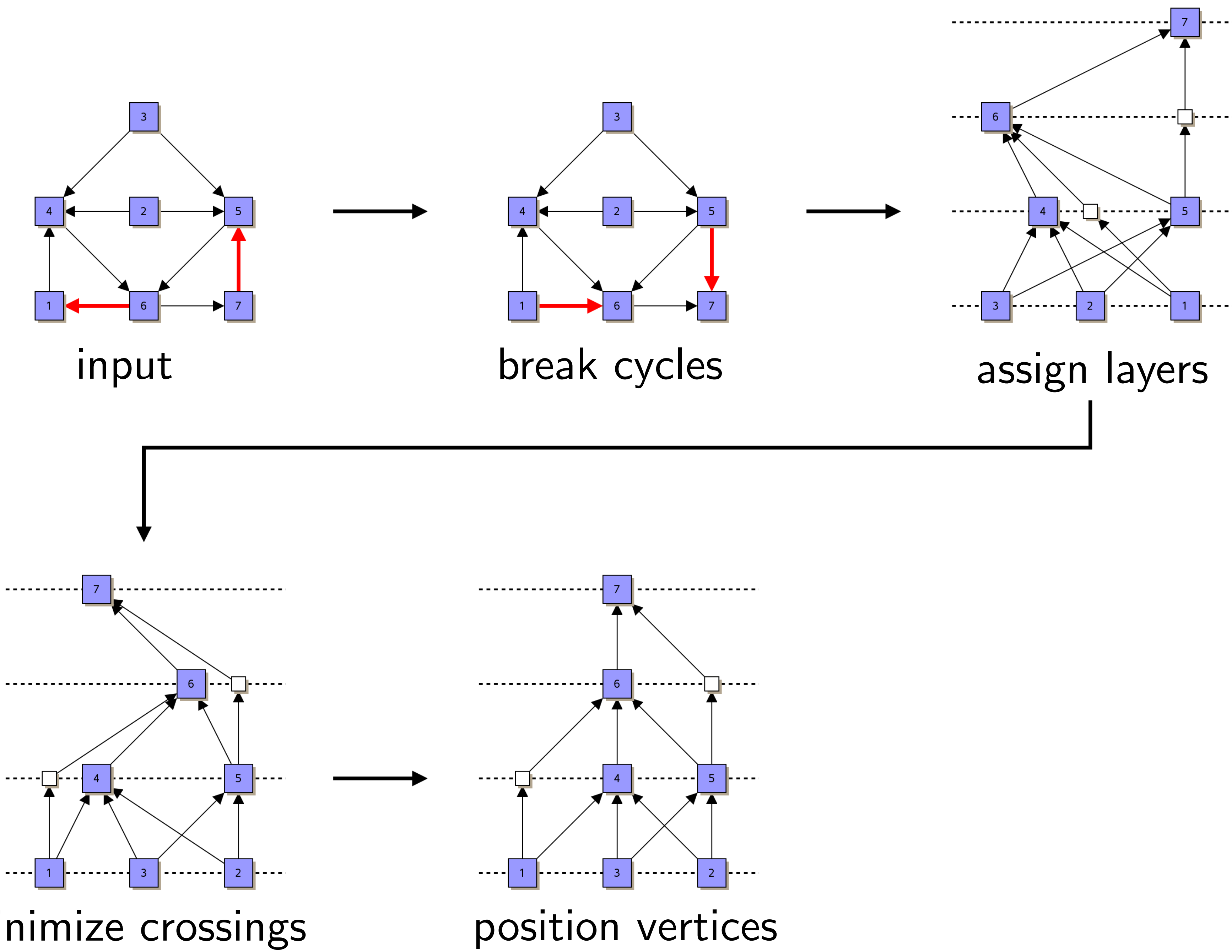
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



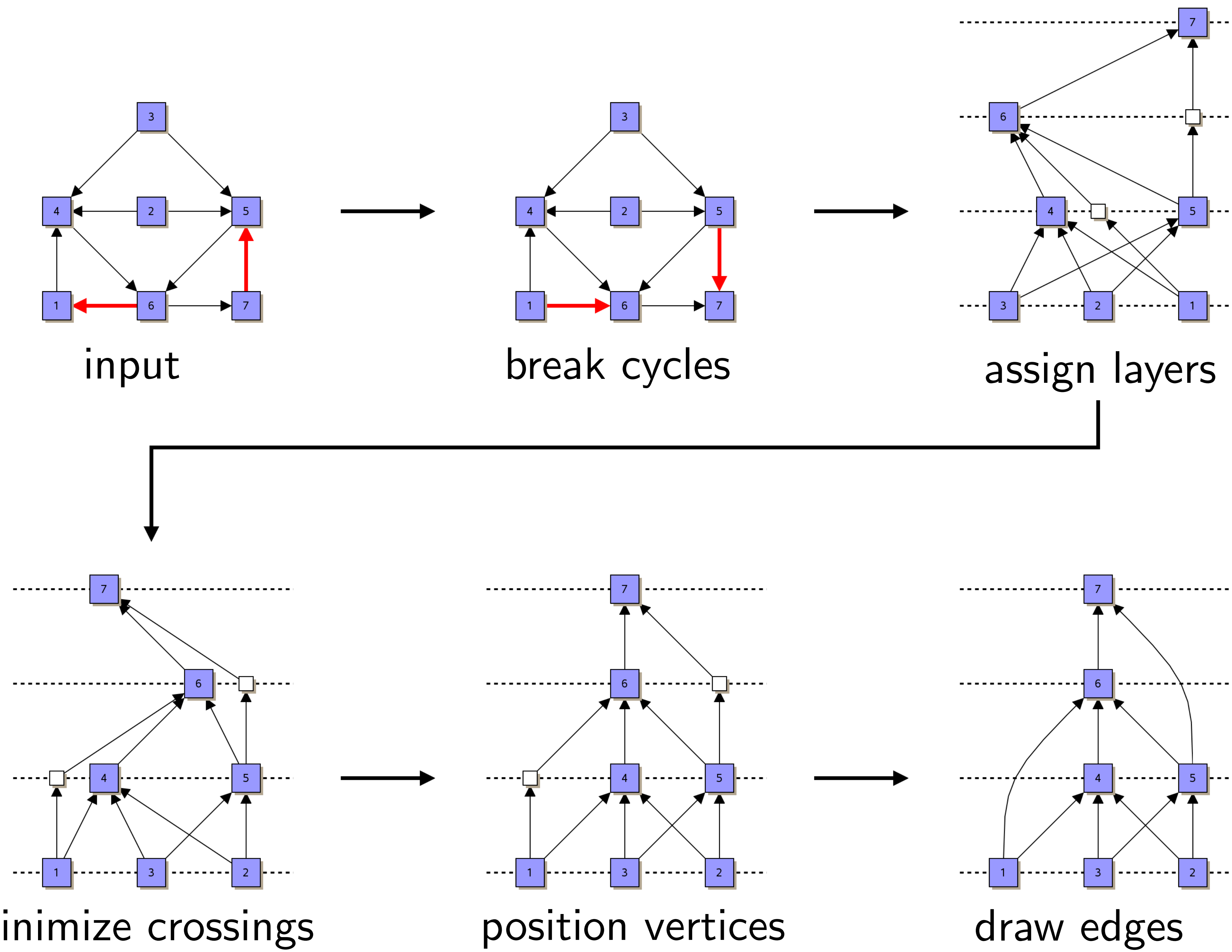
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



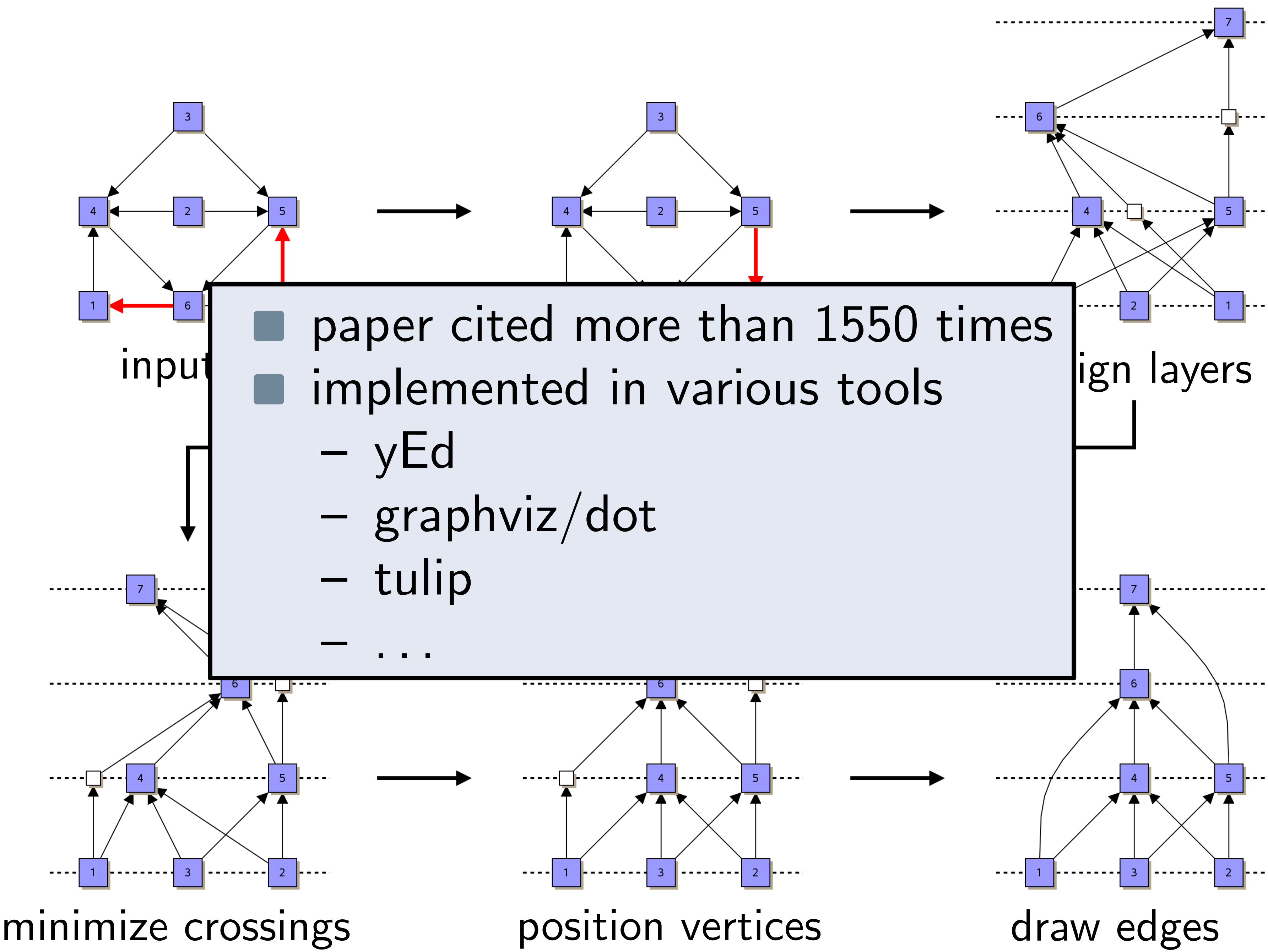
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)

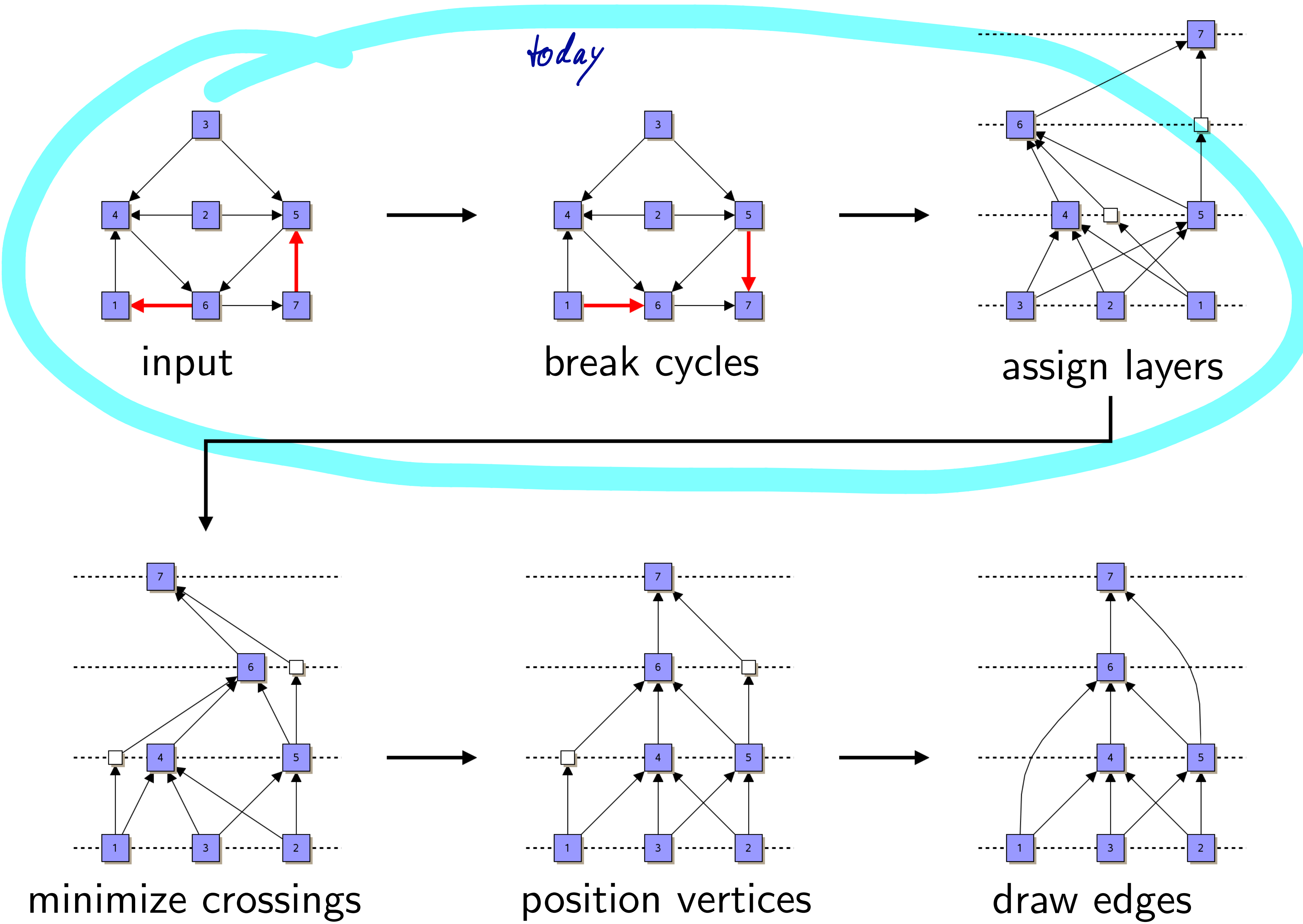


Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)

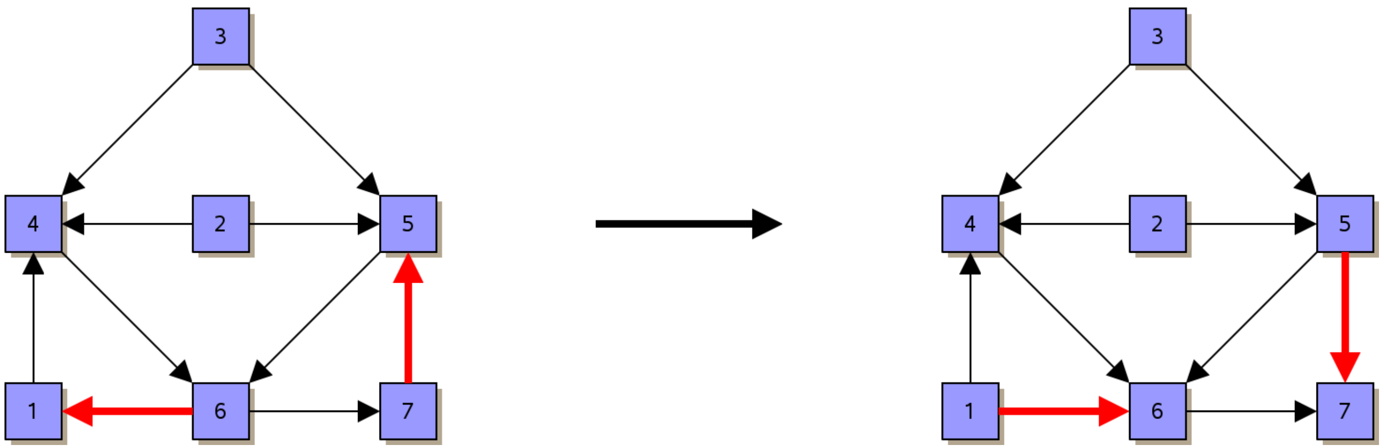


- paper cited more than 1550 times
- implemented in various tools
  - yEd
  - graphviz/dot
  - tulip
  - ...

Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



Step 1: Break Cycles



What would you do?

Feedback Arc Set

Idea:
- find maximum acyclic subgraph
- reverse the directions of the other edges

Feedback Arc Set

Idea:
- find maximum acyclic subgraph
- reverse the directions of the other edges

Maximum Acyclic Subgraph

input: directed graph $D = (V, A)$ $A' \subseteq A$

output: acyclic subgraph $D' = (V, A')$ with $|A'|$ maximum

Feedback Arc Set

Idea:
- find maximum acyclic subgraph
- reverse the directions of the other edges

Maximum Acyclic Subgraph

input: directed graph $D = (V, A)$

output: acyclic subgraph $D' = (V, A')$ with $|A'|$ maximum

Minimum Feedback Arc Set (FAS)

input: directed graph $D = (V, A)$

output: $A_f \subset A$, s.t. $D_f = (V, A \setminus A_f)$ acyclic and $|A_f|$ minimum

Feedback Arc Set

Idea:
- find maximum acyclic subgraph
- reverse the directions of the other edges

Maximum Acyclic Subgraph

input: directed graph $D = (V, A)$

output: acyclic subgraph $D' = (V, A')$ with $|A'|$ maximum

Minimum Feedback Arc Set (FAS)

input: directed graph $D = (V, A)$

output: $A_f \subset A$, s.t. $D_f = (V, A \setminus A_f)$ acyclic and $|A_f|$ minimum

Minimum Feedback Set (FS)

input: directed graph $D = (V, A)$

output: $A_f \subset A$, s.t. $D_f = (V, A \setminus A_f \cup \mathrm{rev}(A_f))$ acyclic and $|A_f|$ minimum

Feedback Arc Set

Idea:
- find maximum acyclic subgraph
- reverse the directions of the other edges

Maximum Acyclic Subgraph

input: directed graph $D = (V, A)$

output: acyclic subgraph $D' = (V, A')$ with $|A'|$ maximum

Minimum Feedback Arc Set (FAS)

input: directed graph $D = (V, A)$

output: $A_f \subset A$, s.t. $D_f = (V, A \setminus A_f)$ acyclic and $|A_f|$ minimum

Minimum Feedback Set (FS)

input: directed graph $D = (V, A)$

output: $A_f \subset A$, s.t. $D_f = (V, A \setminus A_f \cup \mathrm{rev}(A_f))$ acyclic and $|A_f|$ minimum

All three problems are NP-hard!

Heuristic 1 (Berger, Shor 1990)

$A' := \emptyset$;

foreach $v \in V$ **do**

  if $|N^{\rightarrow}(v)| \geq |N^{\leftarrow}(v)|$ **then**

    $A' := A' \cup N^{\rightarrow}(v)$; → v becomes source in D'

  else

    $A' := A' \cup N^{\leftarrow}(v)$; → v becomes sink in D'

  remove v and $N(v)$ from D

return $D' = (V, A')$

$$
\begin{aligned}
N^{\rightarrow}(v) &:= \{(v,u) : (v,u) \in A\} \\
N^{\leftarrow}(v) &:= \{(u,v) : (u,v) \in A\} \\
N(v) &:= N^{\rightarrow}(v) \cup N^{\leftarrow}(v)
\end{aligned}
$$

Heuristic 1 (Berger, Shor 1990)

$A' := \emptyset;$

foreach $v \in V$ **do**

  if $|N^{\rightarrow}(v)| \geq |N^{\leftarrow}(v)|$ **then**

    $A' := A' \cup N^{\rightarrow}(v);$

  else

    $A' := A' \cup N^{\leftarrow}(v);$

  remove v and $N(v)$ from D

return $D' = (V, A')$

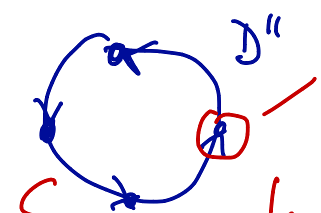
$$\begin{aligned} N^{\rightarrow}(v) &:= \{(v,u) : (v,u) \in A\} \\ N^{\leftarrow}(v) &:= \{(u,v) : (u,v) \in A\} \\ N(v) &:= N^{\rightarrow}(v) \cup N^{\leftarrow}(v) \end{aligned}$$

- D' is a DAG (directed acyclic graph)
- $A \setminus A'$ is a feedback arc set (not necessarily minimum)

- Why are there no cycles in D'?
- Is $D'' = (V, A' \cup \text{rev}(A \setminus A'))$ acyclic?
- What is the running time?
- What do we know about $|A'|$?

→ every vertex is

or

→


first vertex of C seen in the algorithm

↳ cannot have both directions in C


Heuristic 1 (Berger, Shor 1990)

$A' := \emptyset$;

foreach $v \in V$ **do**

  if $|N^{\rightarrow}(v)| \geq |N^{\leftarrow}(v)|$ **then**

    $A' := A' \cup N^{\rightarrow}(v)$;

  else

    $A' := A' \cup N^{\leftarrow}(v)$;

  remove v and $N(v)$ from D

return $D' = (V, A')$

$$\begin{aligned} N^{\rightarrow}(v) &:= \{(v,u) : (v,u) \in A\} \\ N^{\leftarrow}(v) &:= \{(u,v) : (u,v) \in A\} \\ N(v) &:= N^{\rightarrow}(v) \cup N^{\leftarrow}(v) \end{aligned}$$

- D' is a DAG (directed acyclic graph)
- $A \setminus A'$ is a feedback arc set (not necessarily minimum)
- running time $O(|V| + |A|)$
- $|A'| \geq |A|/2$

Heuristic 2 (Eades, Lin, Smyth 1993)

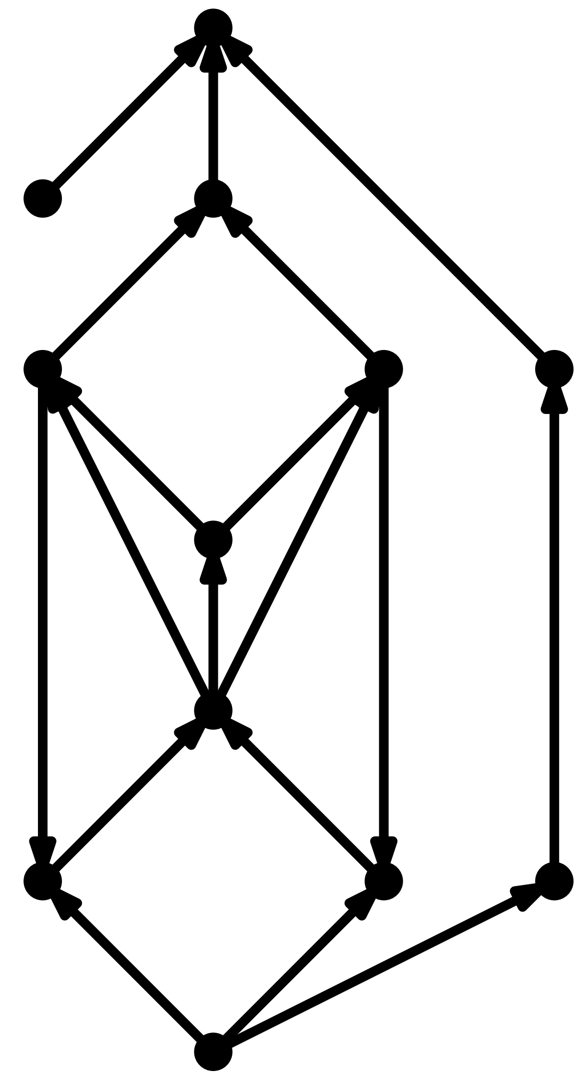
$A' := \emptyset$;

while $V \neq \emptyset$ **do**

  while V contains a sink v **do**

    $A' \leftarrow A' \cup N^{\leftarrow}(v)$

    remove v and $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$



Heuristic 2 (Eades, Lin, Smyth 1993)

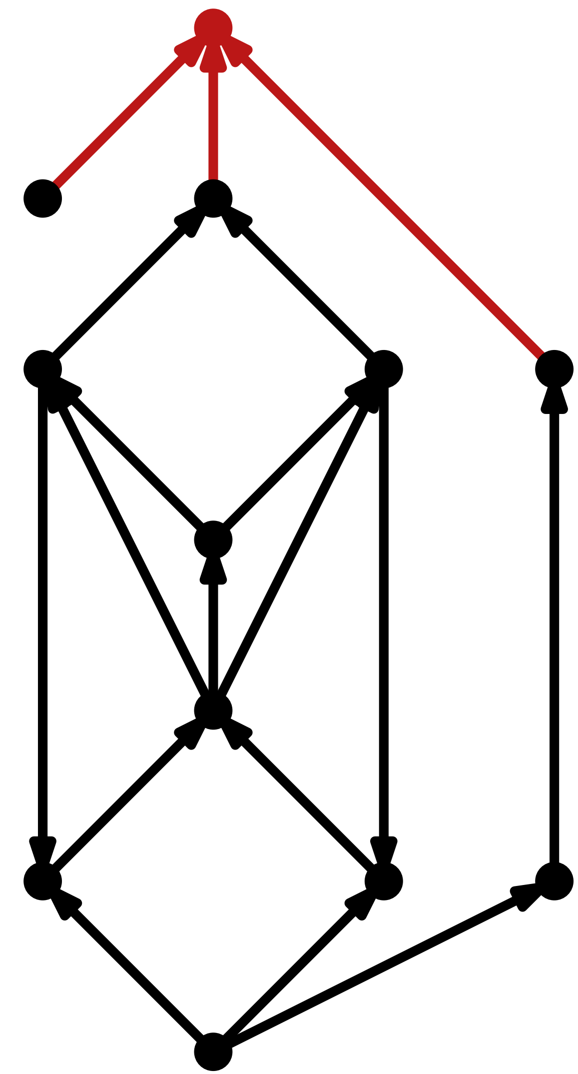
$A' := \emptyset$;

while $V \neq \emptyset$ **do**

  while V contains a sink v **do**

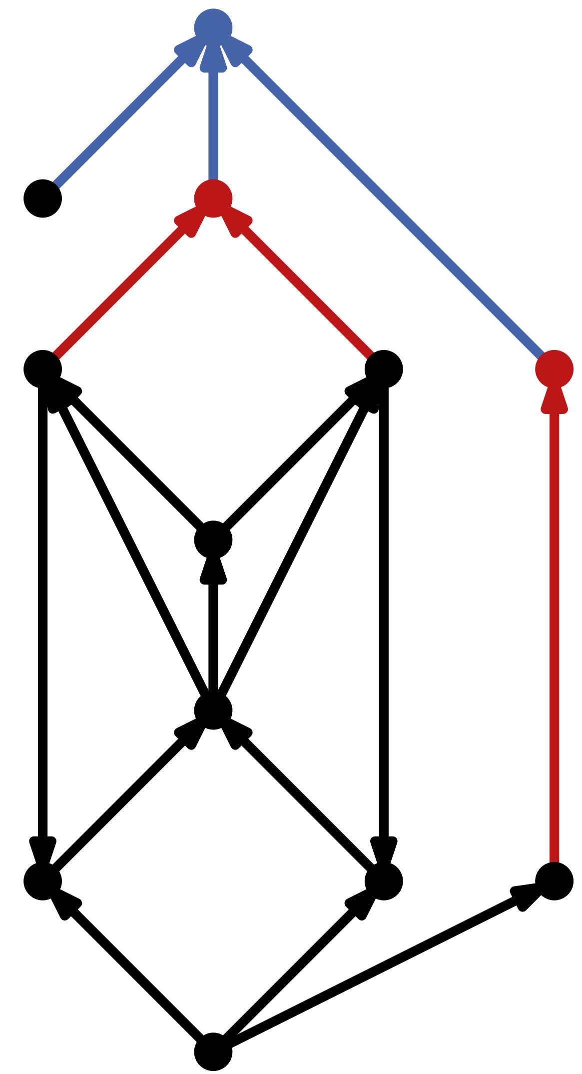
    $A' \leftarrow A' \cup N^{\leftarrow}(v)$

    remove v and $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$



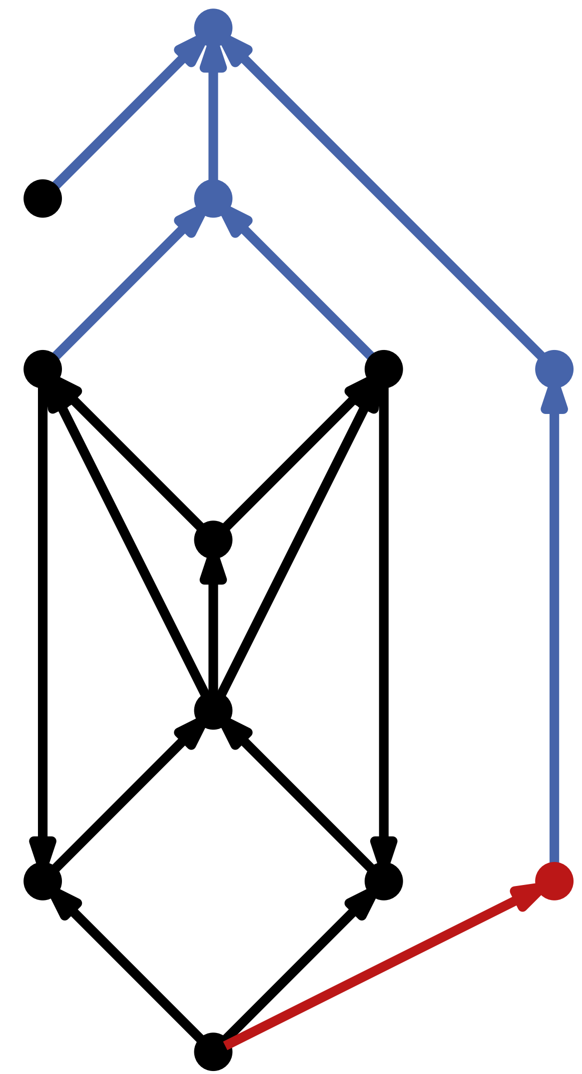
Heuristic 2 (Eades, Lin, Smyth 1993)

$A' := \emptyset$;
while $V \neq \emptyset$ **do**
    while V contains a sink v **do**
        $A' \leftarrow A' \cup N^{\leftarrow}(v)$
        remove v and $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$



Heuristic 2 (Eades, Lin, Smyth 1993)

$A' := \emptyset$;
while $V \neq \emptyset$ **do**
  while V contains a sink v **do**
    $A' \leftarrow A' \cup N^{\leftarrow}(v)$
    remove v and $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$



Heuristic 2 (Eades, Lin, Smyth 1993)

$A' := \emptyset$;

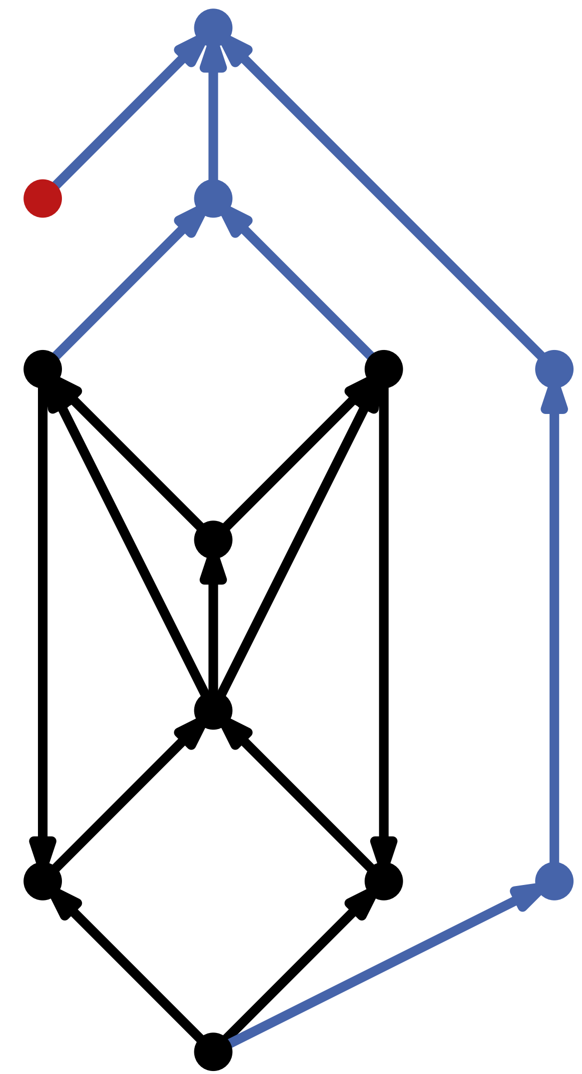
while $V \neq \emptyset$ **do**

  while V contains a sink v **do**

    $A' \leftarrow A' \cup N^{\leftarrow}(v)$

    remove v and $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$

  remove all isolated vertices from V: $\{V, n, m\}_{\text{iso}}$



Heuristic 2 (Eades, Lin, Smyth 1993)

$A' := \emptyset$;

while $V \neq \emptyset$ **do**

  while V contains a sink v **do**

    $A' \leftarrow A' \cup N^{\leftarrow}(v)$

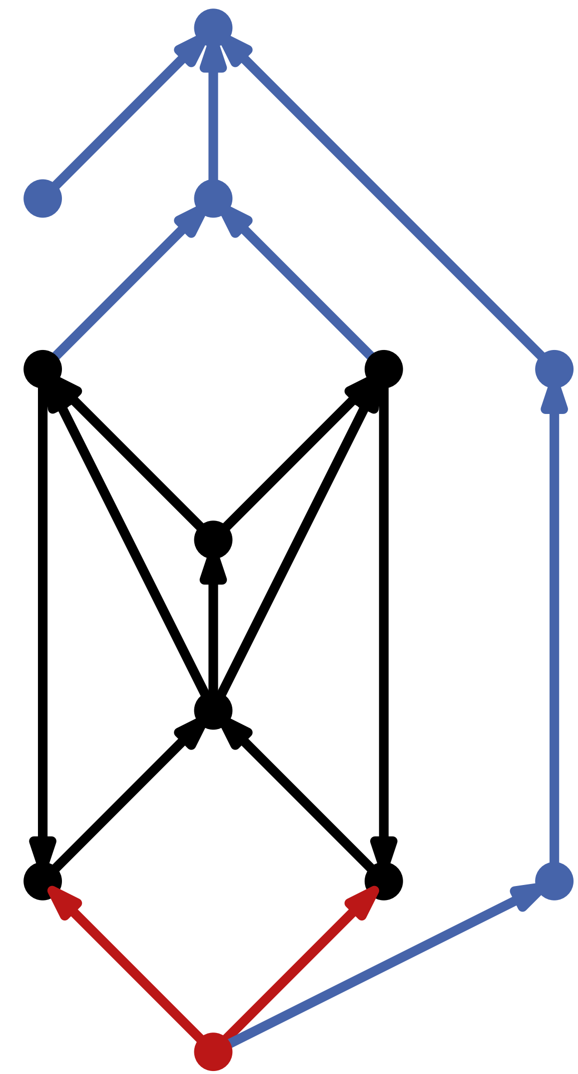
    remove v and $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$

  remove all isolated vertices from V: $\{V, n, m\}_{\text{iso}}$

  while V contains a source v **do**

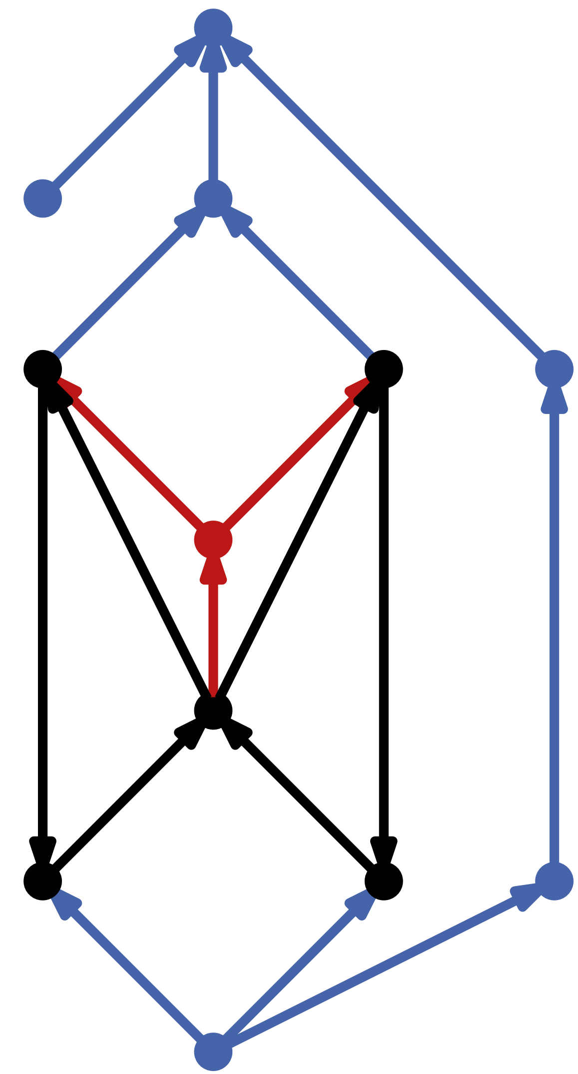
    $A' \leftarrow A' \cup N^{\rightarrow}(v)$

    remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$



Heuristic 2 (Eades, Lin, Smyth 1993)

$A' := \emptyset$;
while $V \neq \emptyset$ **do**
    while V contains a sink v **do**
        $A' \leftarrow A' \cup N^{\leftarrow}(v)$
        remove v and $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$
    remove all isolated vertices from V: $\{V, n, m\}_{\text{iso}}$
    while V contains a source v **do**
        $A' \leftarrow A' \cup N^{\rightarrow}(v)$
        remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$
    if $V \neq \emptyset$ **then**
        let $v \in V$ maximize $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$
        $A' \leftarrow A' \cup N^{\rightarrow}(v)$
        remove v and $N(v)$: $\{V, n, m\}_{\{=,<\}}$



Heuristic 2 (Eades, Lin, Smyth 1993)

$A' := \emptyset$;

while $V \neq \emptyset$ **do**

    while V contains a sink v **do**

        $A' \leftarrow A' \cup N^{\leftarrow}(v)$

        remove v and $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$

    remove all isolated vertices from V: $\{V, n, m\}_{\text{iso}}$

    while V contains a source v **do**

        $A' \leftarrow A' \cup N^{\rightarrow}(v)$

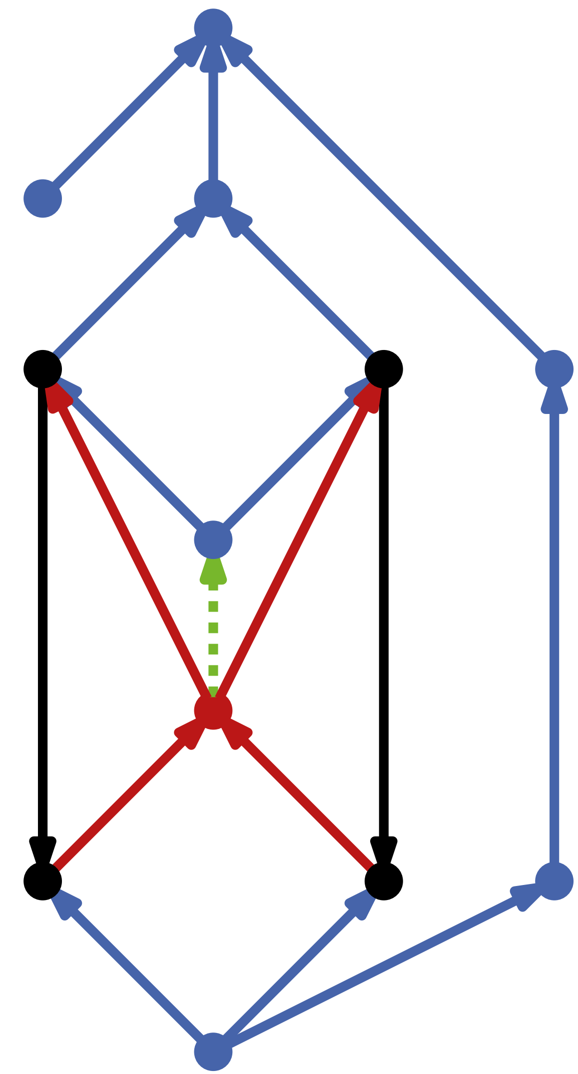
        remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

    if $V \neq \emptyset$ **then**

        let $v \in V$ maximize $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$

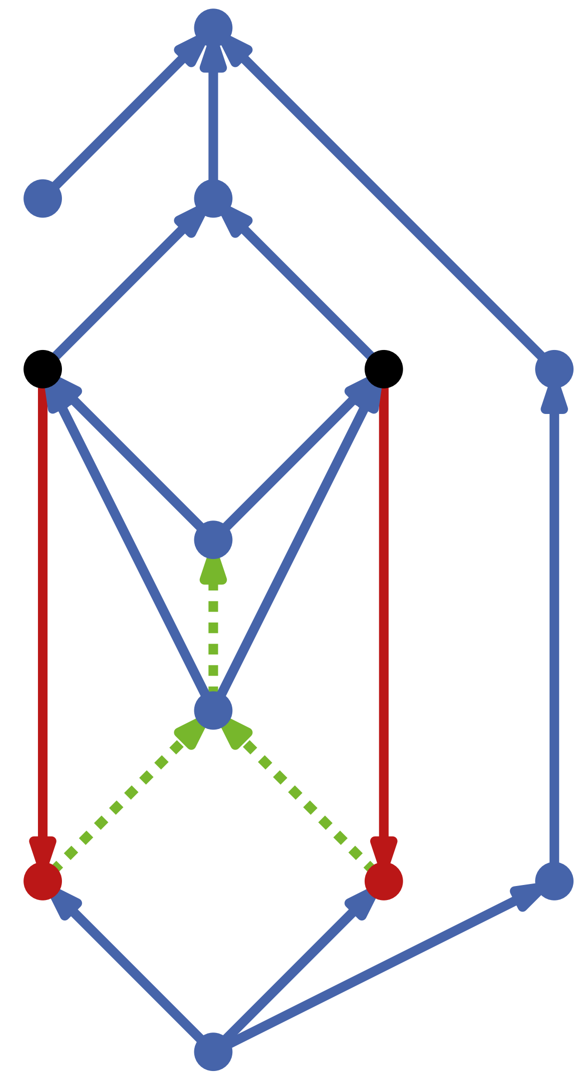
        $A' \leftarrow A' \cup N^{\rightarrow}(v)$

        remove v and $N(v)$: $\{V, n, m\}_{\{=,<\}}$



Heuristic 2 (Eades, Lin, Smyth 1993)

```
A' := ∅;
while V ≠ ∅ do
    while V contains a sink v do
        A' ← A' ∪ N←(v)
        remove v and N←(v): {V, n, m}_sink
    remove all isolated vertices from V: {V, n, m}_iso
    while V contains a source v do
        A' ← A' ∪ N→(v)
        remove v and N→(v): {V, n, m}_source
    if V ≠ ∅ then
        let v ∈ V maximize |N→(v)| − |N←(v)|
        A' ← A' ∪ N→(v)
        remove v and N(v): {V, n, m}_{=,<}
```



Heuristic 2 (Eades, Lin, Smyth 1993)

$A' := \emptyset$;

while $V \neq \emptyset$ **do**

    while V contains a sink v **do**

        $A' \leftarrow A' \cup N^{\leftarrow}(v)$

        remove v and $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$

    remove all isolated vertices from V: $\{V, n, m\}_{\text{iso}}$

    while V contains a source v **do**

        $A' \leftarrow A' \cup N^{\rightarrow}(v)$

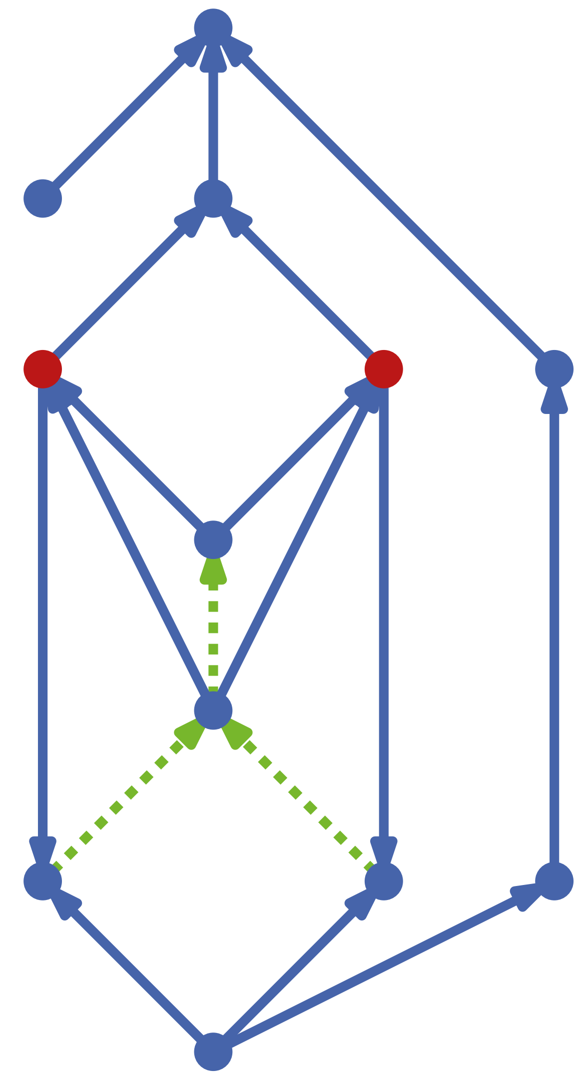
        remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

    if $V \neq \emptyset$ **then**

        let $v \in V$ maximize $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$

        $A' \leftarrow A' \cup N^{\rightarrow}(v)$

        remove v and $N(v)$: $\{V, n, m\}_{\{=,<\}}$



Heuristic 2 (Eades, Lin, Smyth 1993)

$A' := \emptyset$;

while $V \neq \emptyset$ **do**

  while V contains a sink v **do**

    $A' \leftarrow A' \cup N^{\leftarrow}(v)$

    remove v and $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$

  remove all isolated vertices from V: $\{V, n, m\}_{\text{iso}}$

  while V contains a source v **do**

    $A' \leftarrow A' \cup N^{\rightarrow}(v)$

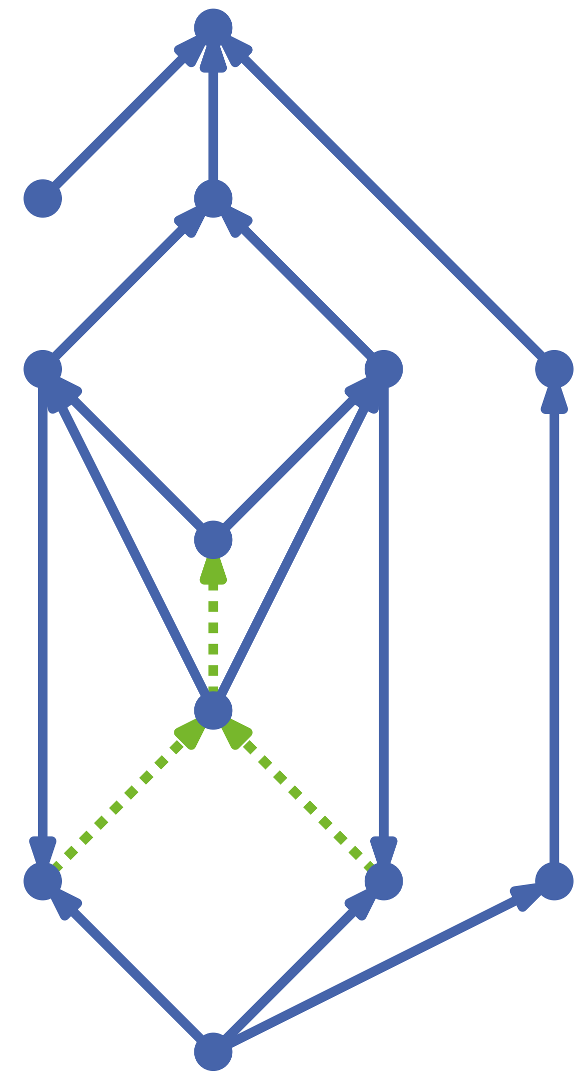
    remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

  if $V \neq \emptyset$ **then**

    let $v \in V$ maximize $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$

    $A' \leftarrow A' \cup N^{\rightarrow}(v)$

    remove v and $N(v)$: $\{V, n, m\}_{\{=,<\}}$

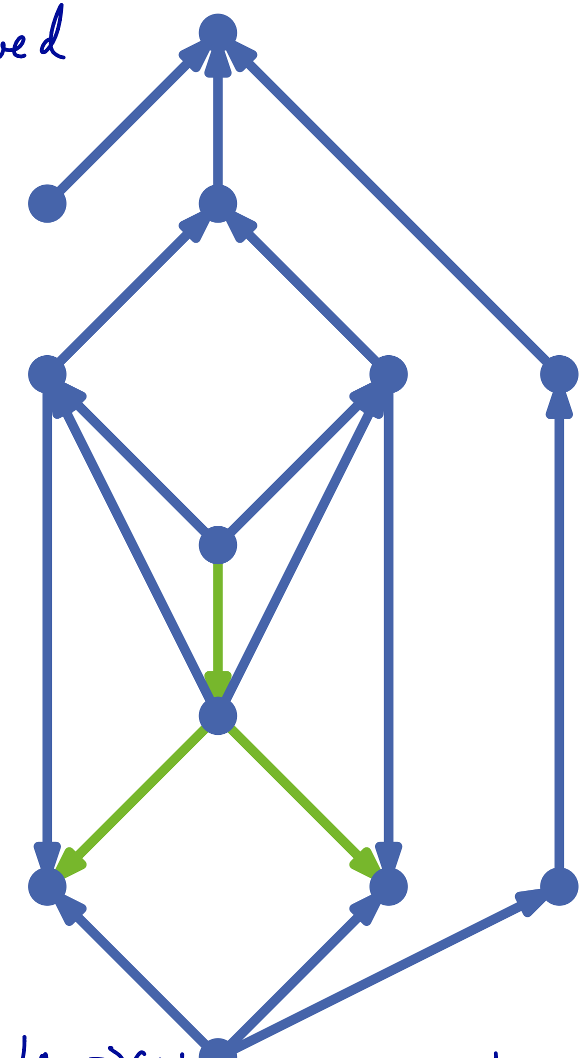


Heuristic 2 (Eades, Lin, Smyth 1993)

$A' := \emptyset$;
while $V \neq \emptyset$ **do**
    while V contains a sink v **do**
        $A' \leftarrow A' \cup N^{\leftarrow}(v)$
        remove v and $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$
    remove all isolated vertices from V: $\{V, n, m\}_{\text{iso}}$
    while V contains a source v **do**
        $A' \leftarrow A' \cup N^{\rightarrow}(v)$
        remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$
    if $V \neq \emptyset$ **then**
        let $v \in V$ maximize $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$
        $A' \leftarrow A' \cup N^{\rightarrow}(v)$
        remove v and $N(v)$: $\{V, n, m\}_{\{=,<\}}$

vertex set
vertices
edges
removed as a sink

$\max_{v \in V} |N^{\rightarrow}(v)| - |N^{\leftarrow}(v)| \geq 0$

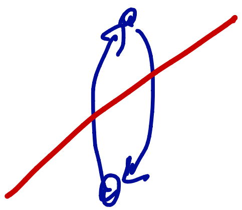


Observation: $n = n_{sink} + n_{source} + n_{iso} + n_{=} + n_{<}$

$m = m_{sink} + m_{source} + m_{=} + m_{<}$

Heuristic 2 – Analysis

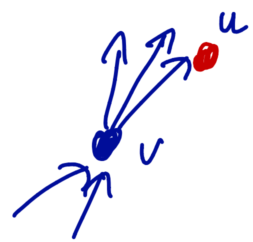
Theorem: Let $D = (V, A)$ be a connected, directed graph without 2-cycles. Heuristic 2 computes an edge set A' with $|A'| \geq |A|/2 + |V|/6$ in $O(|A|)$ time.



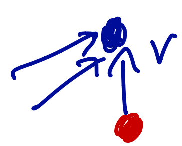
Proof

n_{iso} ?

- none in the beginning
- line 13: no! because u would have been a sink before



- line 9: no! same argument
- line 5: yes, when removing (u,v)



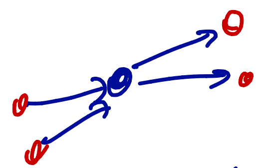
$\Rightarrow n_{iso} \leq n_{sink}$

Heuristic 2 – Analysis

Theorem: Let $D = (V, A)$ be a connected, directed graph without 2-cycles. Heuristic 2 computes an edge set A' with $|A'| \geq |A|/2 + |V|/6$ in $O(|A|)$ time.

$n_=$

- after removing one $v \in V_=$ there is a vertex u with $|N^{\rightarrow}(u)| - |N^{\leftarrow}(u)| > 0$



- next removed vertex is in $V_{sink} \cup V_{source} \cup V_<$

$\Rightarrow n_= \leq n_{sink} + n_{source} + n_<$

$$\Rightarrow n \overset{(*)}{\leq} \underbrace{2n_{sink} + n_{iso}}_{\leq 3m_{sink}} + 2n_{source} + 2n_< \leq 3(m_{sink} + n_{source} + n_<)$$

Heuristic 2 – Analysis

Theorem: Let $D = (V, A)$ be a connected, directed graph without 2-cycles. Heuristic 2 computes an edge set A' with $|A'| \geq |A|/2 + |V|/6$ in $O(|A|)$ time.

how many edges are in A'?

- for $u \in V_=$: $\frac{\deg(u)}{2}$
- for $u \in V_<$: $\geq \frac{\deg(u)+1}{2}$

$$\sum_{u \in V_=} \frac{\deg(u)}{2} = \frac{m_=}{2}$$

$$\sum_{u \in V_<} \frac{\deg(u)+1}{2} = \frac{m_<}{2} + \frac{n_<}{2}$$

$$|A'| \geq m_{source} + m_{sink} + \frac{m_=}{2} + \frac{m_< + n_<}{2}$$

$$= \frac{m}{2} + \frac{m_{source} + m_{sink} + n_<}{2} \geq \frac{m}{2} + \frac{m_{sink} + n_{source} + n_<}{2}$$

$$\overset{(*)}{\geq} \frac{m}{2} + \frac{n}{6}$$

Heuristic 2 – Analysis

Theorem: Let $D = (V, A)$ be a connected, directed graph without 2-cycles. Heuristic 2 computes an edge set A' with $|A'| \geq |A|/2 + |V|/6$ in $O(|A|)$ time.

running time

define buckets $B_{-n+3}, \ldots, B_0, \ldots, B_{n-3}$ → as doubly linked list

- put vertices with $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)| = i$ into B_i
- sources, sinks, isolated into their own sets

- initialize all sets in $O(|A|)$ time
- each step takes $O(\deg(v))$ [includes update of neighbors and shift to next or previous bucket]

Heuristic 2 – Analysis

Theorem: Let $D = (V, A)$ be a connected, directed graph without 2-cycles. Heuristic 2 computes an edge set A' with $|A'| \geq |A|/2 + |V|/6$ in $O(|A|)$ time.

Further techniques:

- $|A'| \geq |A| \left(1/2 + \Omega\left(\frac{1}{\sqrt{\deg_{\max}(D)}}\right)\right)$ (Berger, Shor 1990)
- exact solution using integer linear programming and branch-and-cut (Grötschel et al. 1985)

Heuristic 2 – Analysis

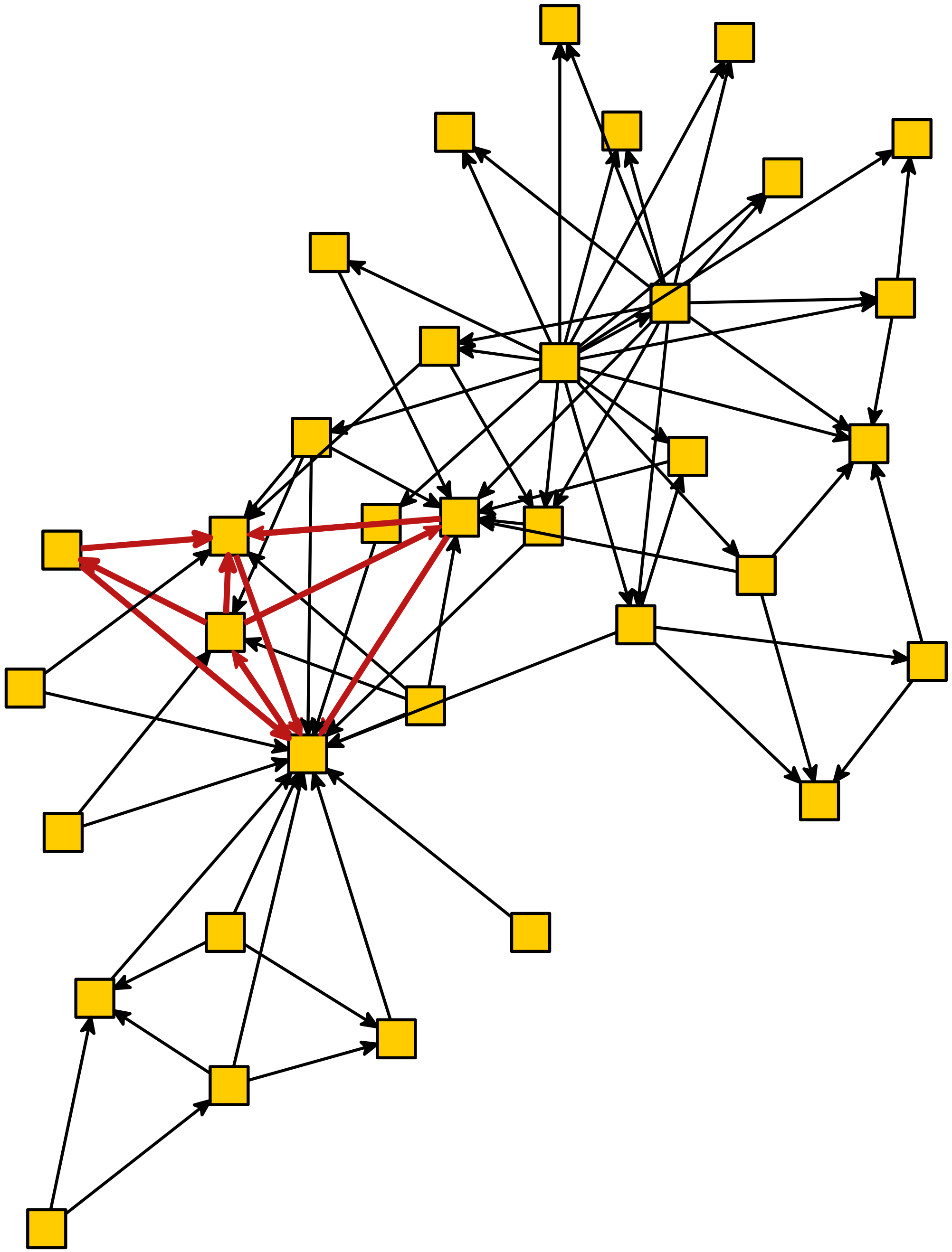
Theorem: Let $D = (V, A)$ be a connected, directed graph without 2-cycles. Heuristic 2 computes an edge set A' with $|A'| \geq |A|/2 + |V|/6$ in $O(|A|)$ time.

Further techniques:

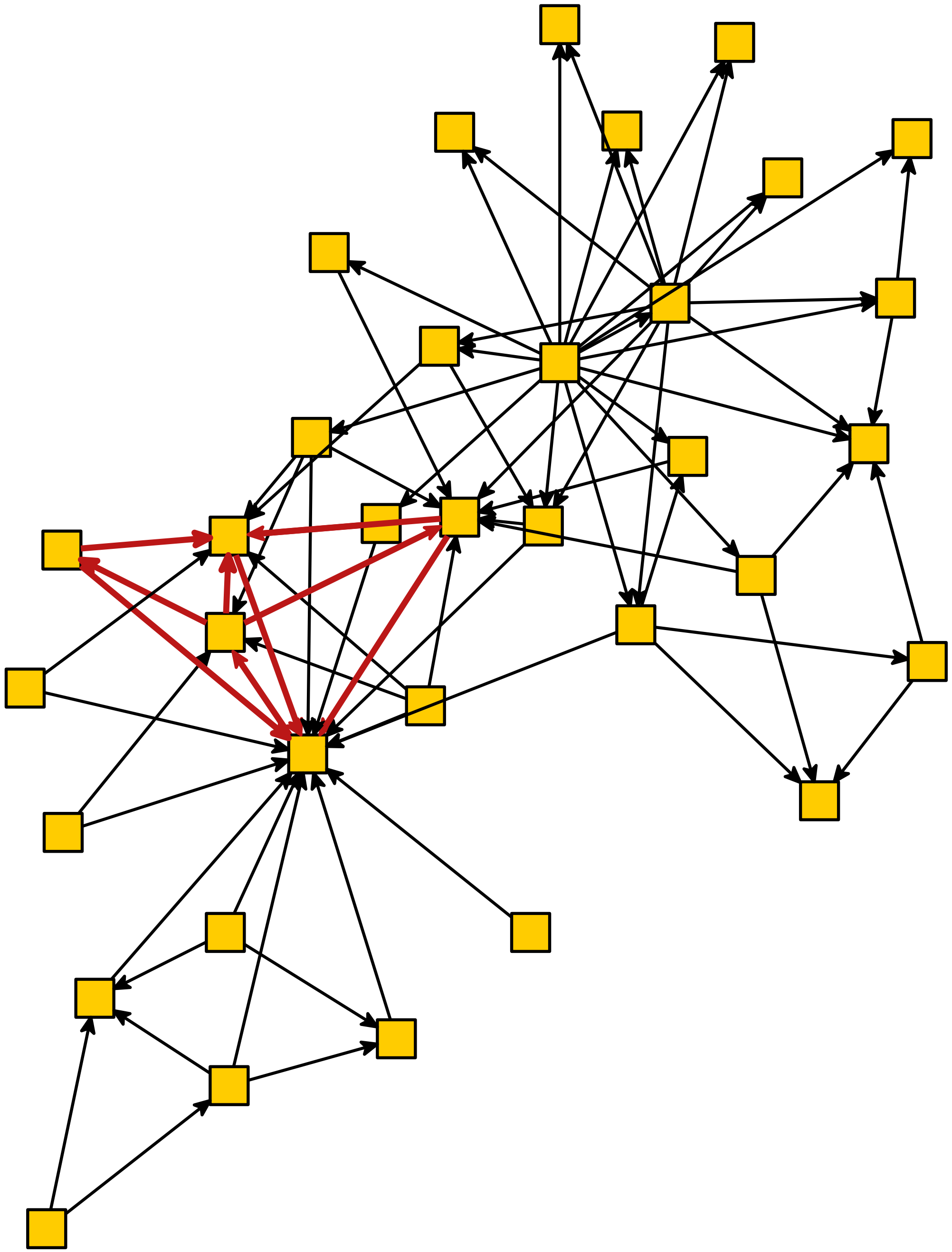
- $|A'| \geq |A| \left(1/2 + \Omega \left(\frac{1}{\sqrt{\deg_{\max}(D)}} \right) \right)$ (Berger, Shor 1990)
- exact solution using integer linear programming and branch-and-cut (Grötschel et al. 1985)

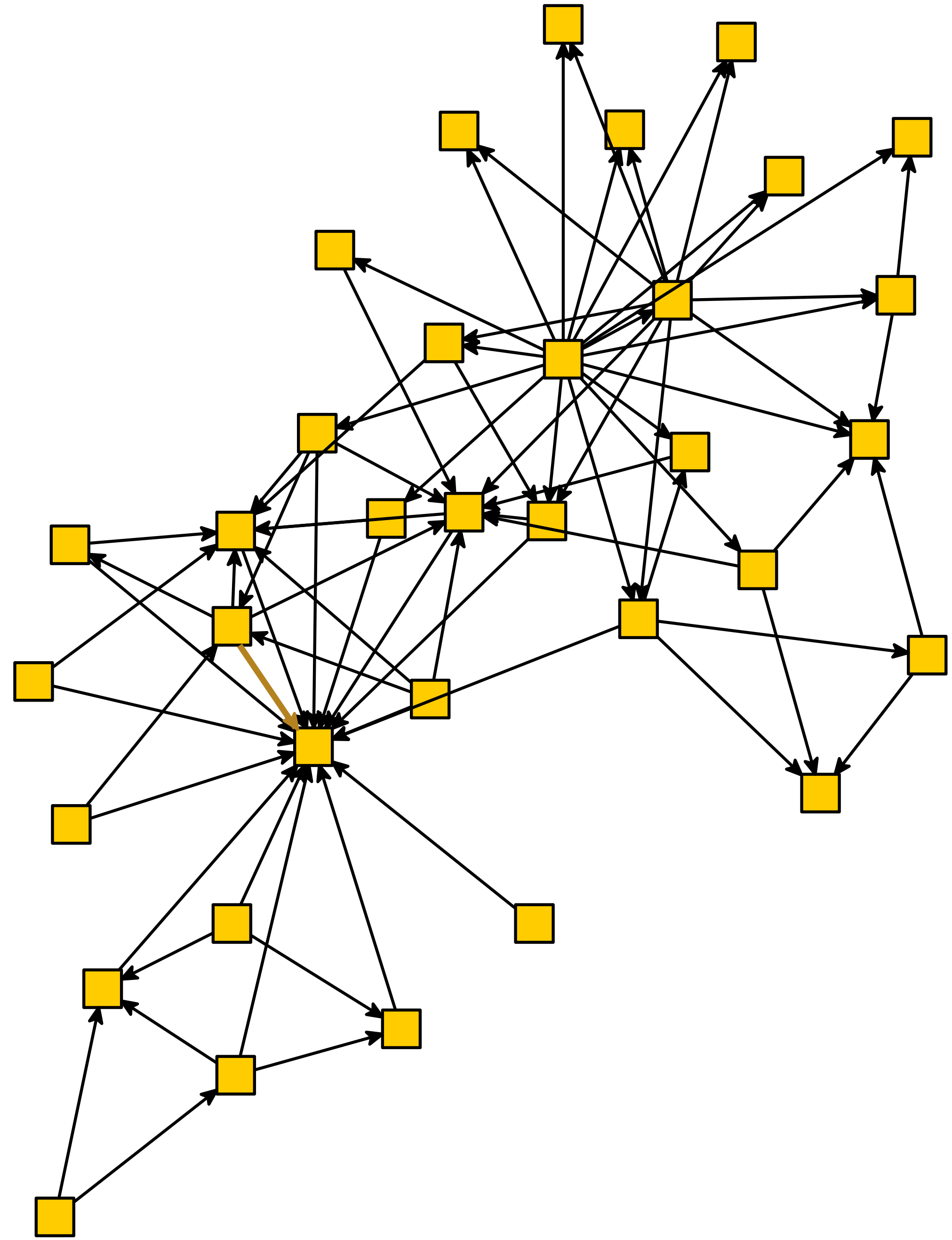
For $|A| \in O(|V|)$ Heuristic 2 is at least as good.

Example

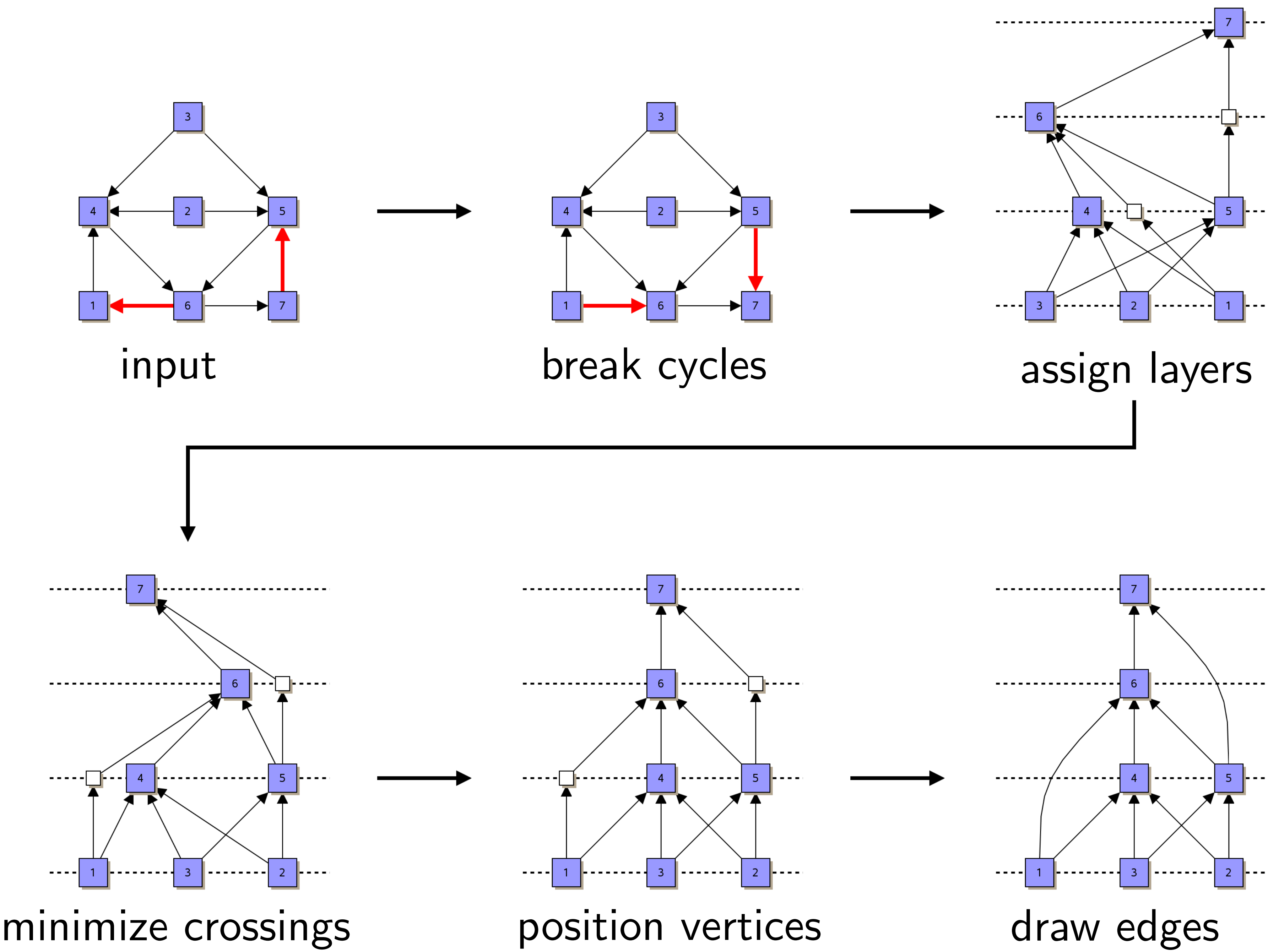


Example

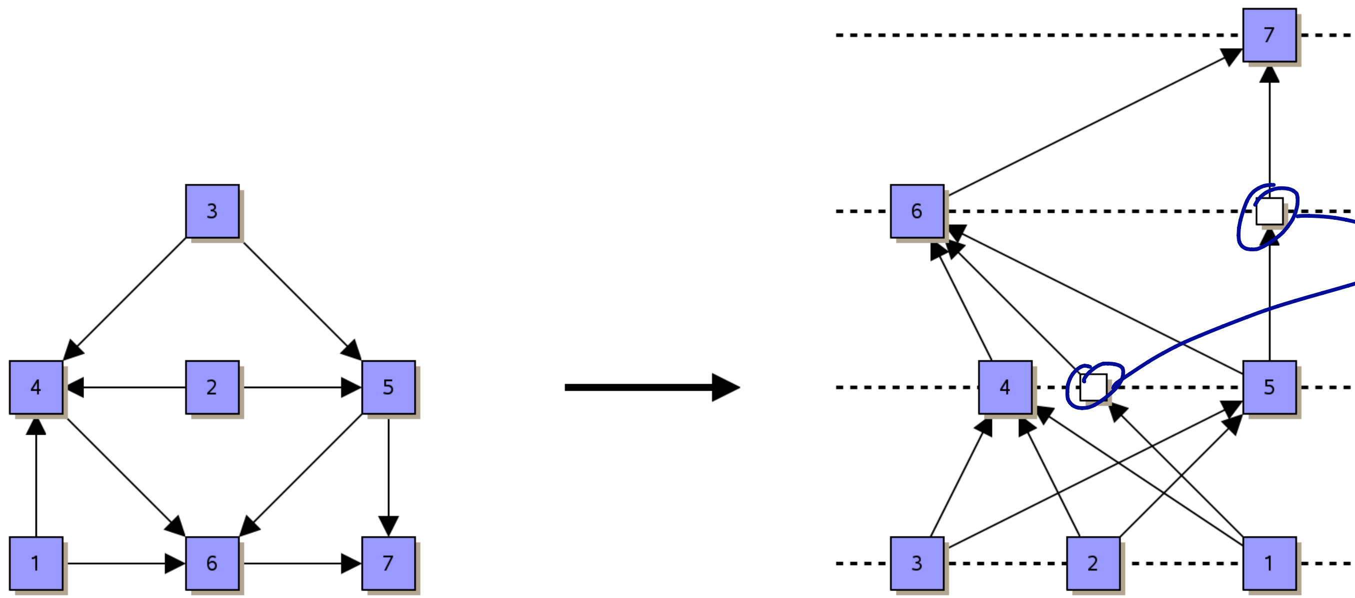




Overview


input
break cycles
assign layers
minimize crossings
position vertices
draw edges

Step 2: Assign Layers



dummy nodes for "long" edges

What would you do?

→ topological sorting

Layer Assignment

Input: directed acyclic graph $D = (V, A)$

Output: partition of V into disjoint subsets (**layers**) $L_1, \dots, L_h$
s.t. $(u, v) \in A$, $u \in L_i$, $v \in L_j \Rightarrow i < j$

Define: y-Coordinate $y(u) = i \Leftrightarrow u \in L_i$

Layer Assignment

Input: directed acyclic graph $D = (V, A)$

Output: partition of V into disjoint subsets (**layers**) $L_1, \dots, L_h$
s.t. $(u, v) \in A$, $u \in L_i$, $v \in L_j \Rightarrow i < j$

Define: y-Coordinate $y(u) = i \Leftrightarrow u \in L_i$

Some optimization criteria:

- minimize the number h of layers ($=$ height of the layouts)
- minimize the width, i.e., $\max\{|L_i| \mid 1 \leq i \leq h\}$
- minimize the longest edge, i.e.,
  $\max\{j - i \mid (u, v) \in A,\, u \in L_i,\, v \in L_j\}$
- minimize the total edge length ($\approx$ number of dummy vertices)

Height Minimization

Idea: assign each vertex v to the layer L_i for which i is the length of the longest simple path from a source to v

- all predecessors are below v
- the total height h is minimal

Height Minimization

Idea: assign each vertex v to the layer L_i for which i is the length of the longest simple path from a source to v

- all predecessors are below v
- the total height h is minimal

Algorithm

- $L_1 \leftarrow$ set of all sources of D
- set $y(u) = 1$ for all $u \in L_1$
- while $D \neq \emptyset$
  - remove L_1 from D and assign $L_1 \leftarrow$ new set of sources
  - set $y(u) \leftarrow \max_{v \in N^{\leftarrow}(u)}\{y(v)\} + 1$ for all $u \in L_1$

Height Minimization

Idea: assign each vertex v to the layer L_i for which i is the length of the longest simple path from a source to v

- all predecessors are below v
- the total height h is minimal

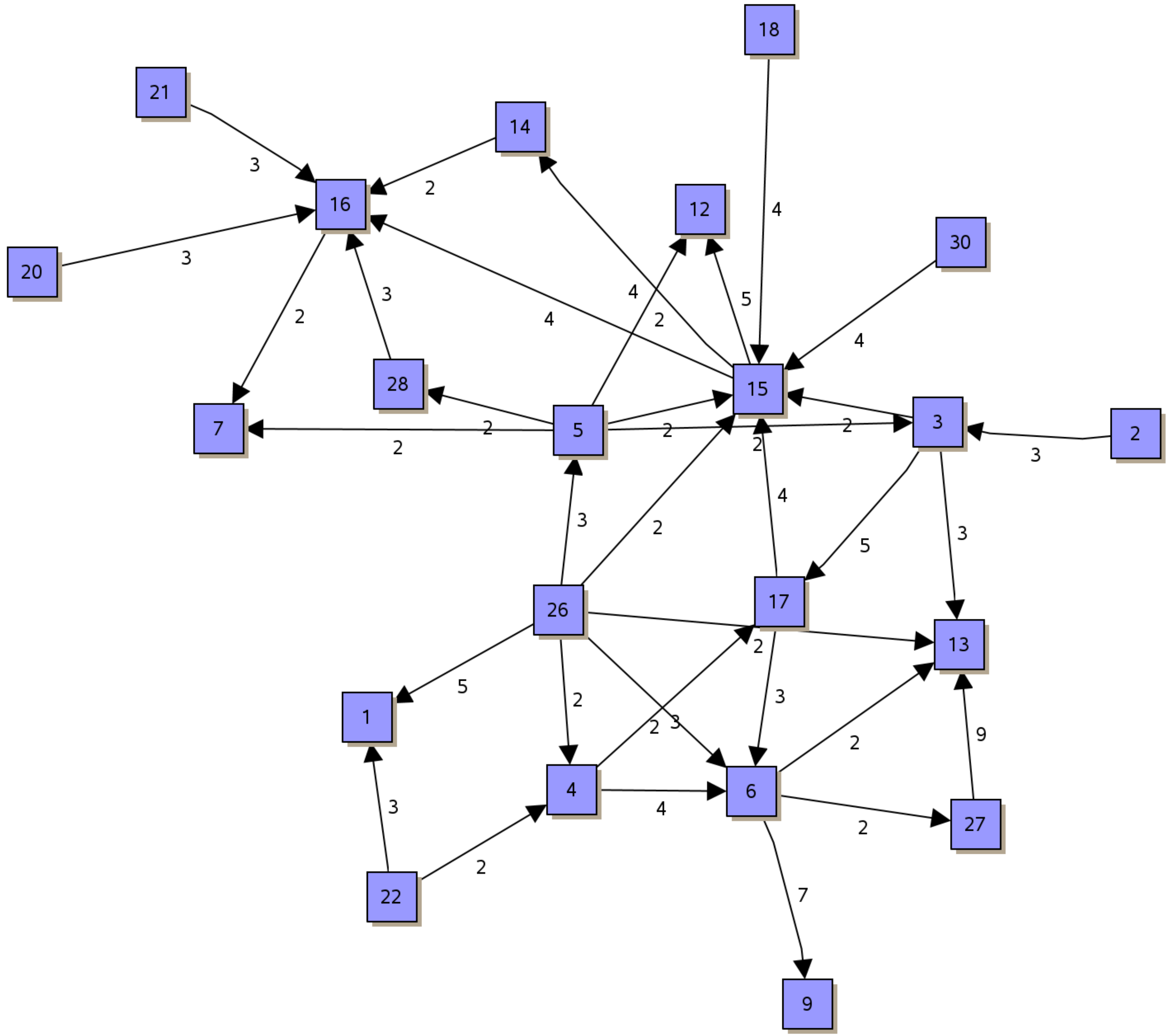
Algorithm

- $L_1 \leftarrow$ set of all sources of D
- set $y(u) = 1$ for all $u \in L_1$
- while $D \neq \emptyset$
  - remove L_1 from D and assign $L_1 \leftarrow$ new set of sources
  - set $y(u) \leftarrow \max_{v \in N^{\leftarrow}(u)}\{y(v)\} + 1$ for all $u \in L_1$

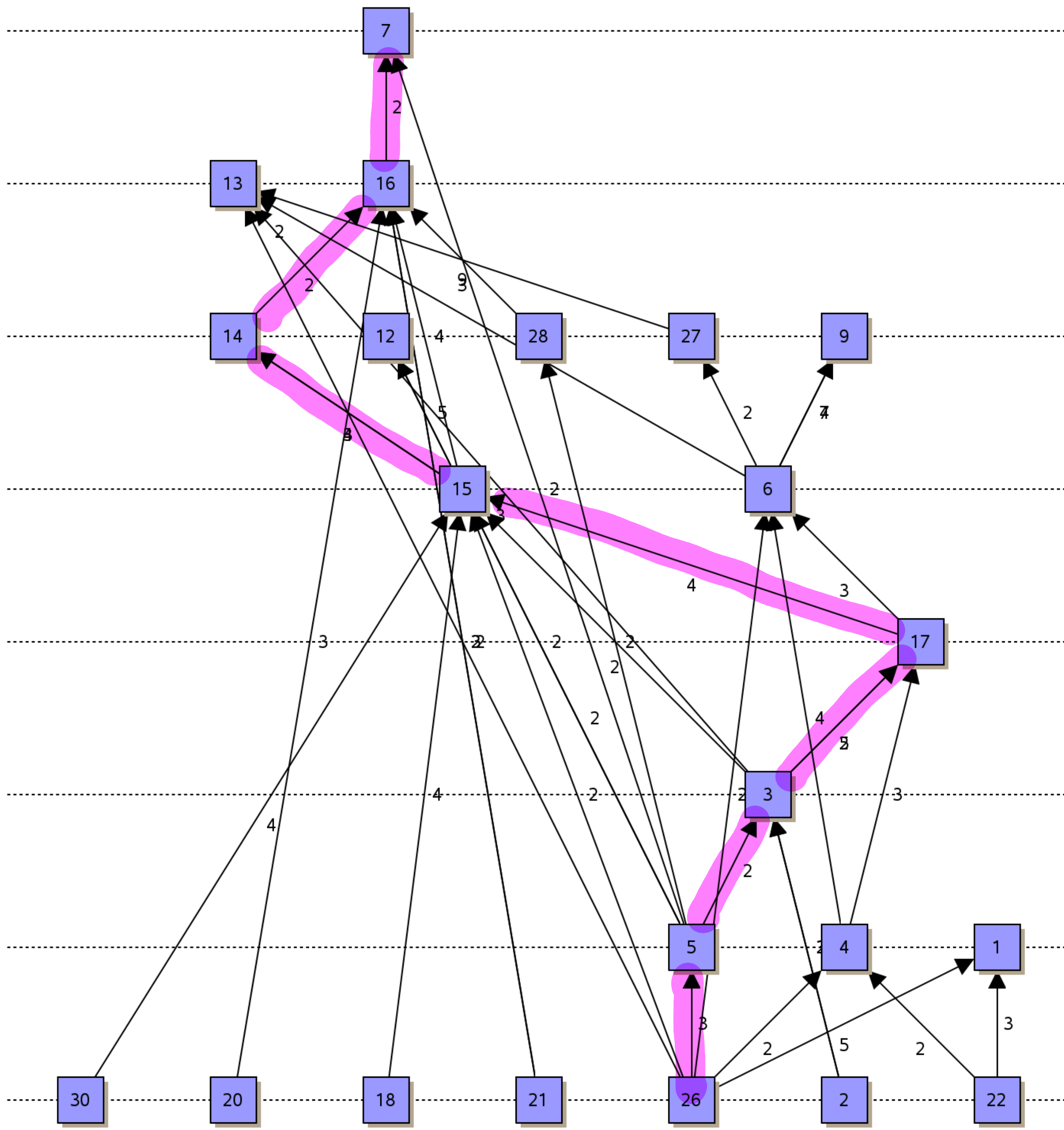
Can we implement this idea in linear time $O(|V| + |A|)$?

Yes! Topological sorting of vertex set

Example



Example



Minimizing Total Edge Length

Can be formulated as an integer linear program:

$$
\begin{array}{rll}
\text{min} & \sum_{(u,v)\in A}(y(v)-y(u)) & \\
\text{subject to} & y(v)-y(u)\geq 1 & \forall (u,v)\in A \\
 & y(v)\geq 1 & \forall v\in V \\
 & y(v)\in \mathbb{Z} & \forall v\in V
\end{array}
$$

Minimizing Total Edge Length

Can be formulated as an integer linear program:

$$\begin{array}{rll} \text{min} & \sum_{(u,v)\in A}(y(v)-y(u)) & \\ \text{subject to} & y(v)-y(u)\geq 1 & \forall (u,v)\in A \\ & y(v)\geq 1 & \forall v\in V \\ & y(v)\in\mathbb{Z} & \forall v\in V \end{array}$$

One can show that:

- constraint matrix is **totally unimodular**
- $\Rightarrow$ solution of the relaxed linear program is integral
- total edge length can be minimized in polynomial time

Announcements

- The class on June 12 is shifted to Monday, June 11 in the same time slot 9:00–11:00 in seminar room 186.
- Student presentations from exercise groups will take place on July 3, 10:00–12:00 and 13:00–15:00.
- We are running an experimental online study on human vs. machine drawings of small graphs.
  You are all invited to participate:

  http://tinyurl.com/turingGD

- oral exams : early July or late September