Karlsruhe Institute of Technology

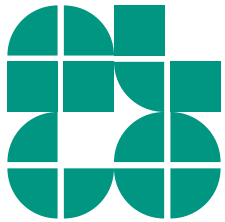
Algorithms for Graph Visualization

Layered Layout

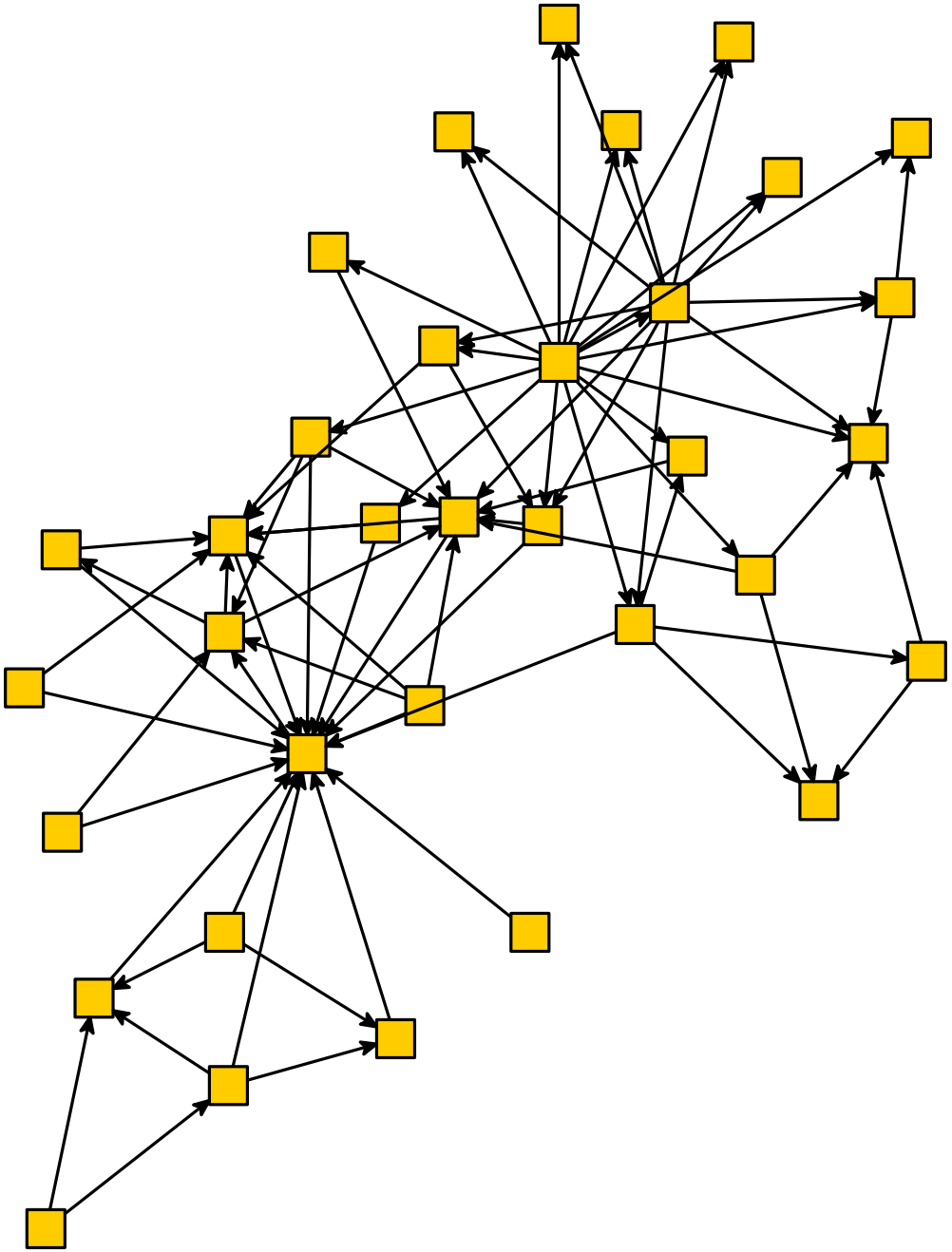
INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Tamara Mchedlidze

13.12.2017



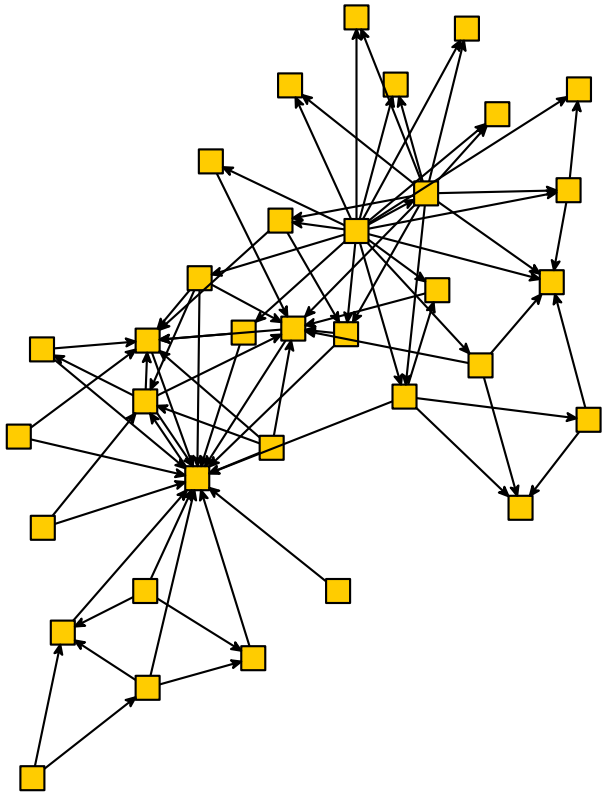
Example

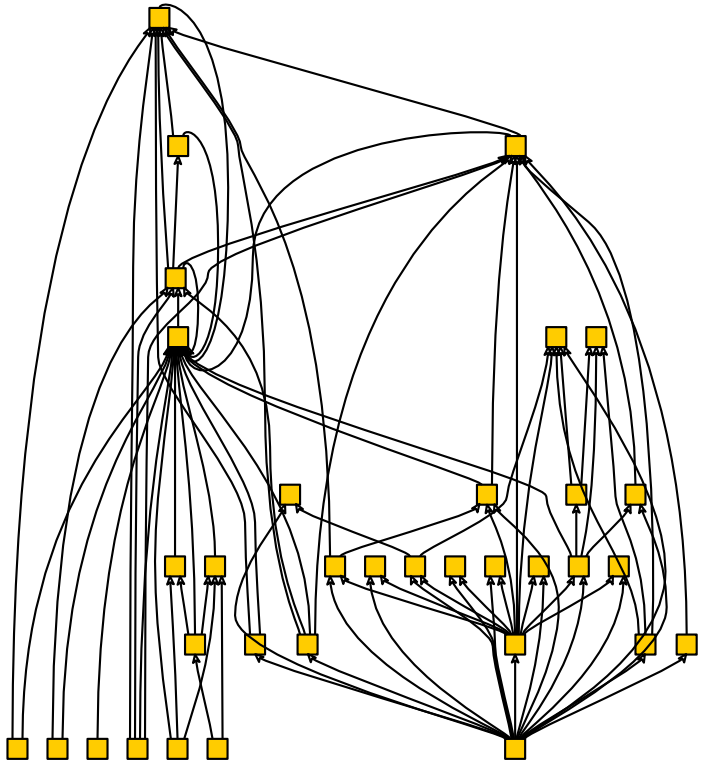


Layered Layout

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

Find: drawing of D that emphasized the hierarchy

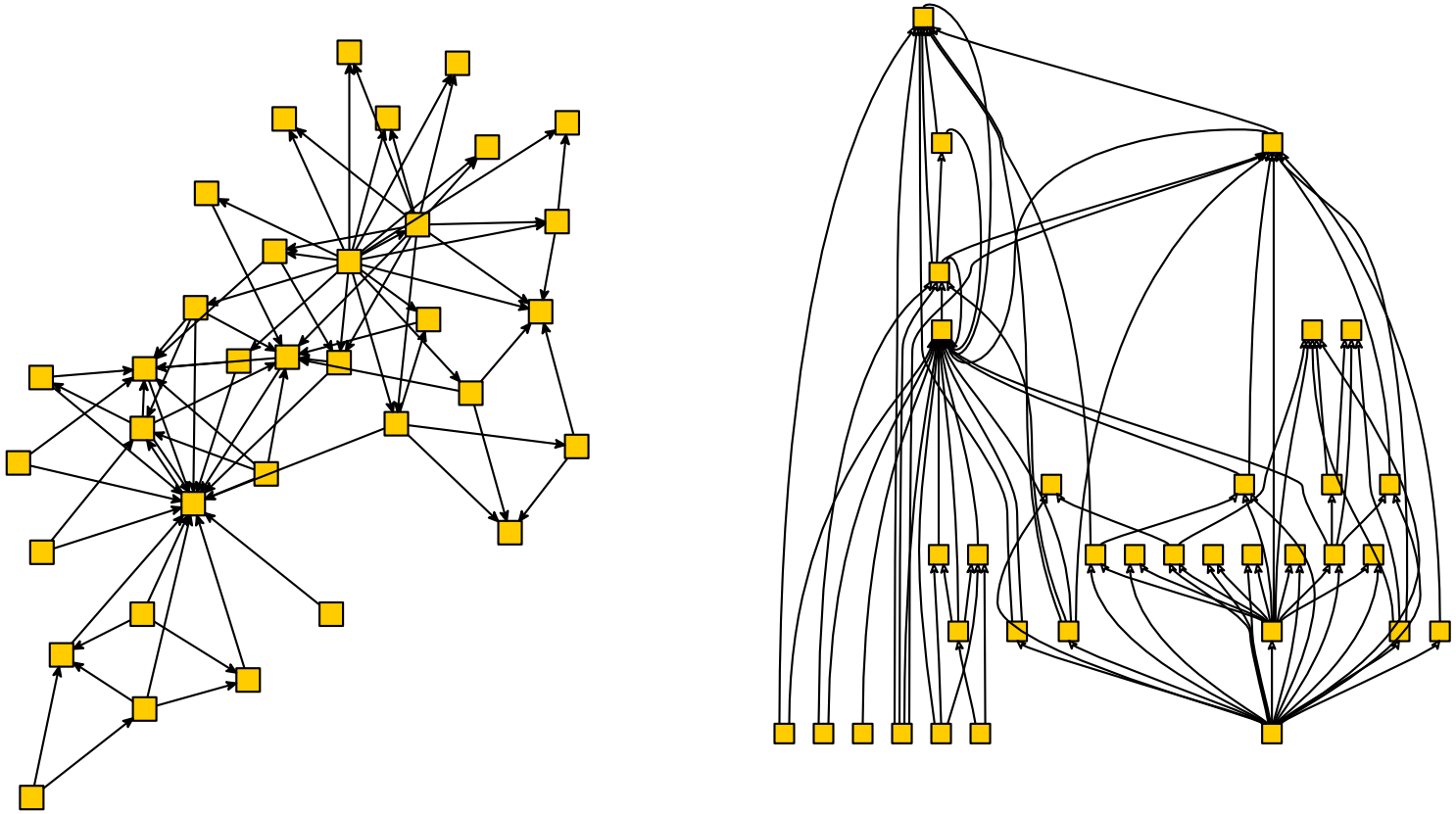




Layered Layout

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

Find: drawing of D that emphasized the hierarchy

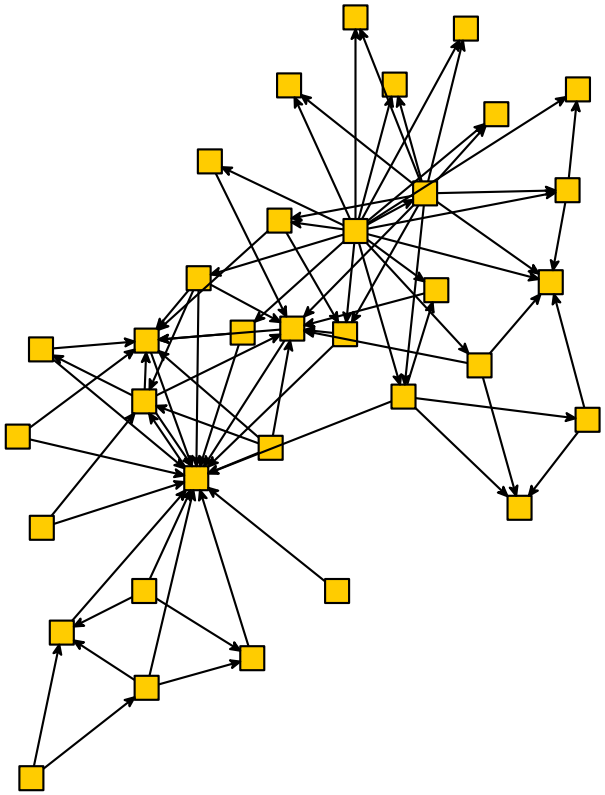


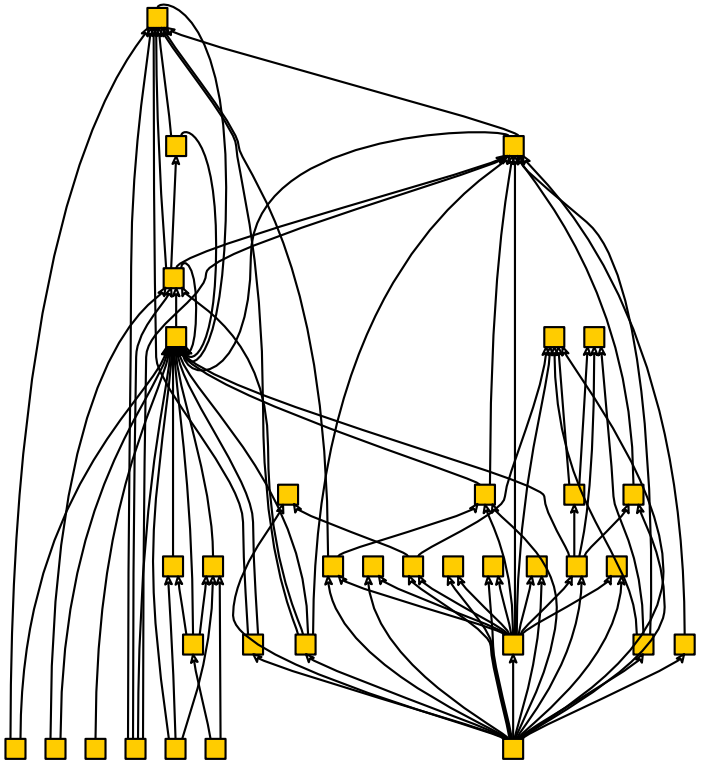
- many edges pointing to the same direction
- nodes lie on (few) horizontal lines

Layered Layout

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

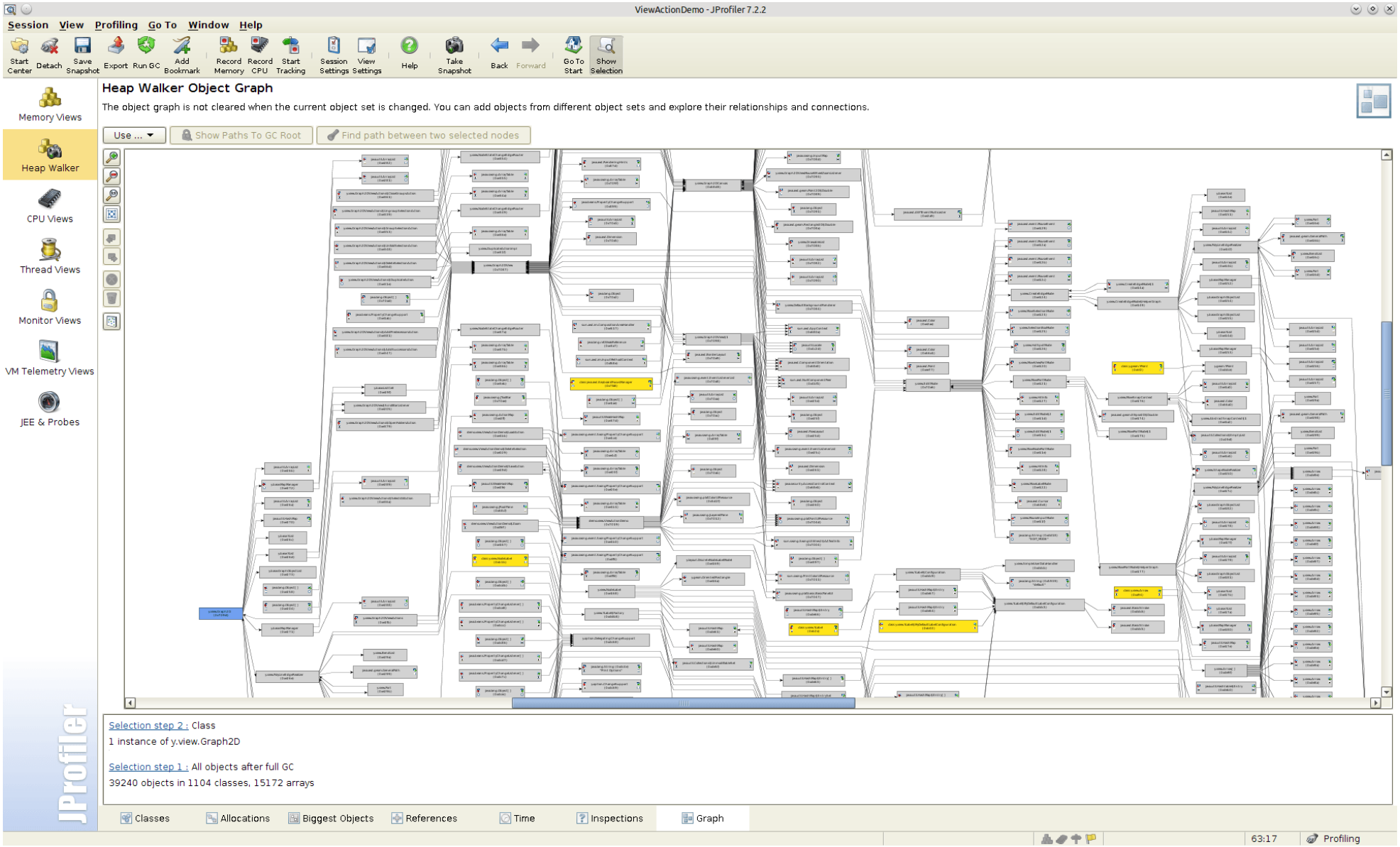
Find: drawing of D that emphasized the hierarchy





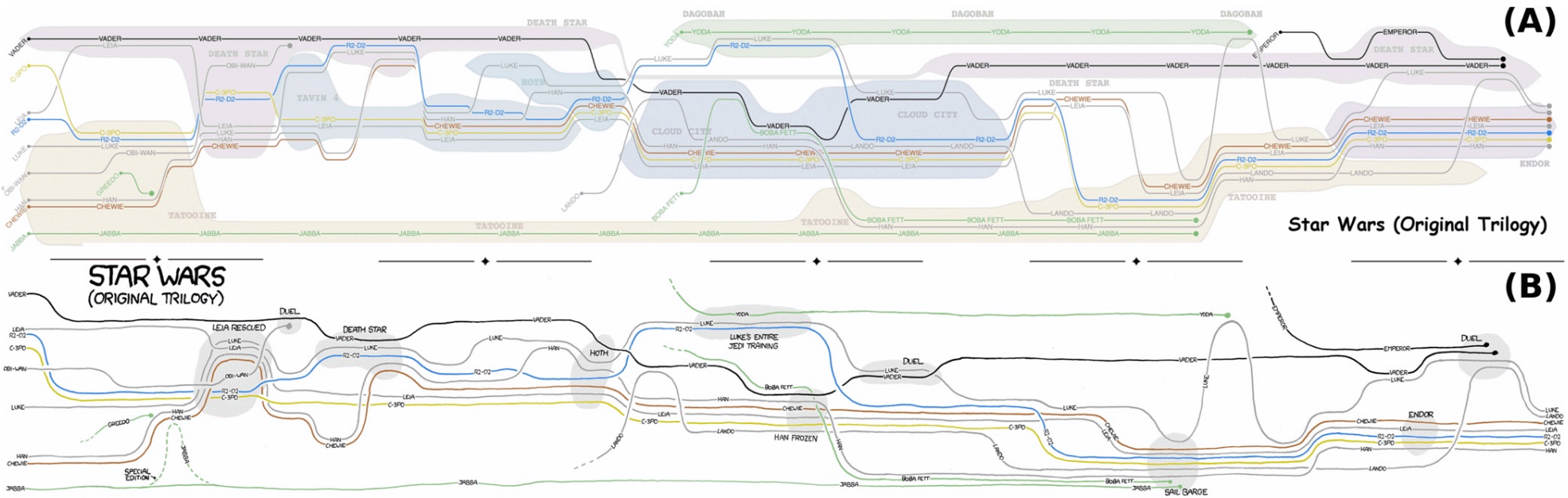
- edges as straight as possible and short
- few edge crossings
- nodes distributed evenly

Application: Java Profiler



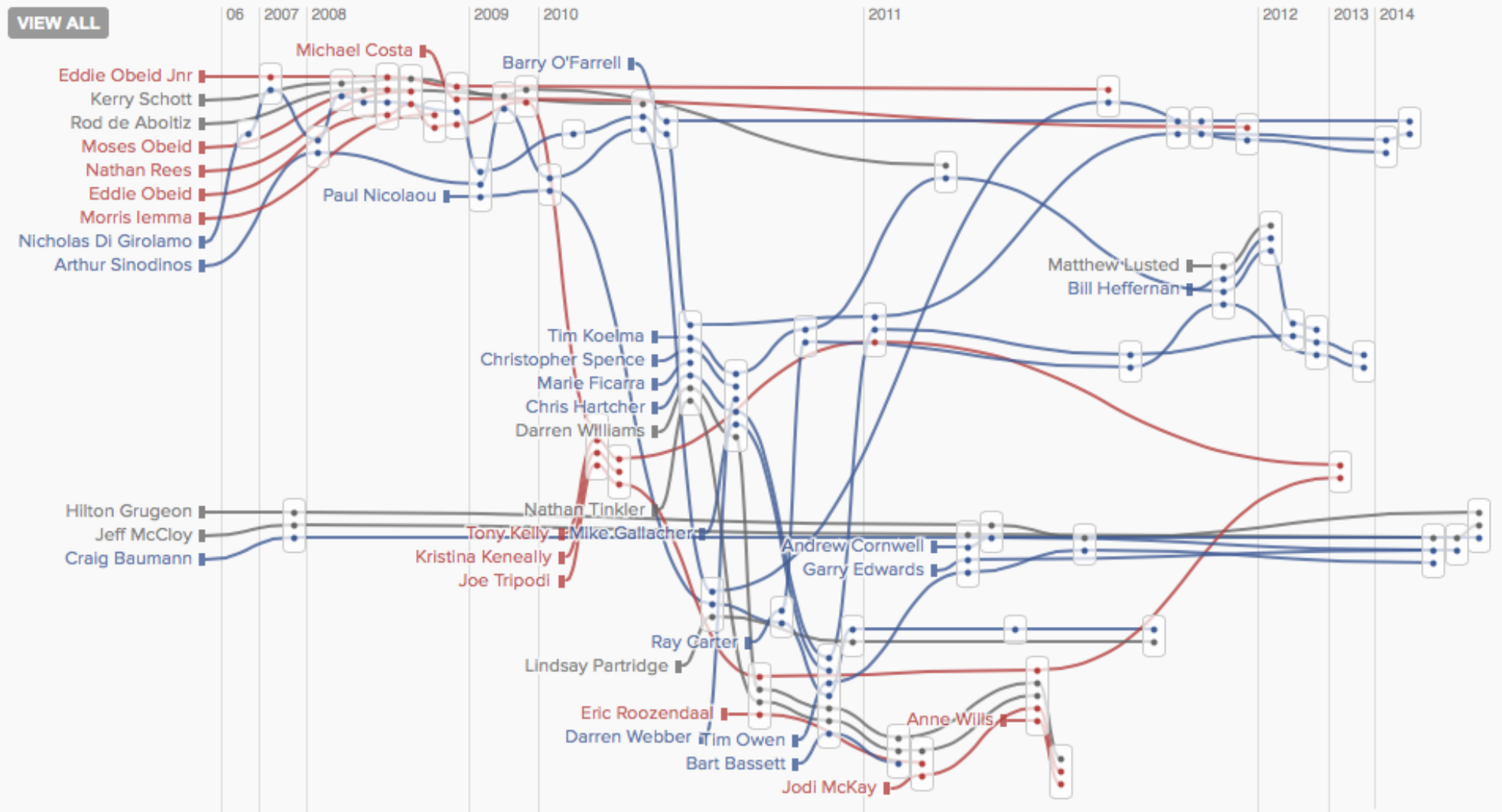
yEd Gallery: Java profiler JProfiler using yFiles

Application: Storylines



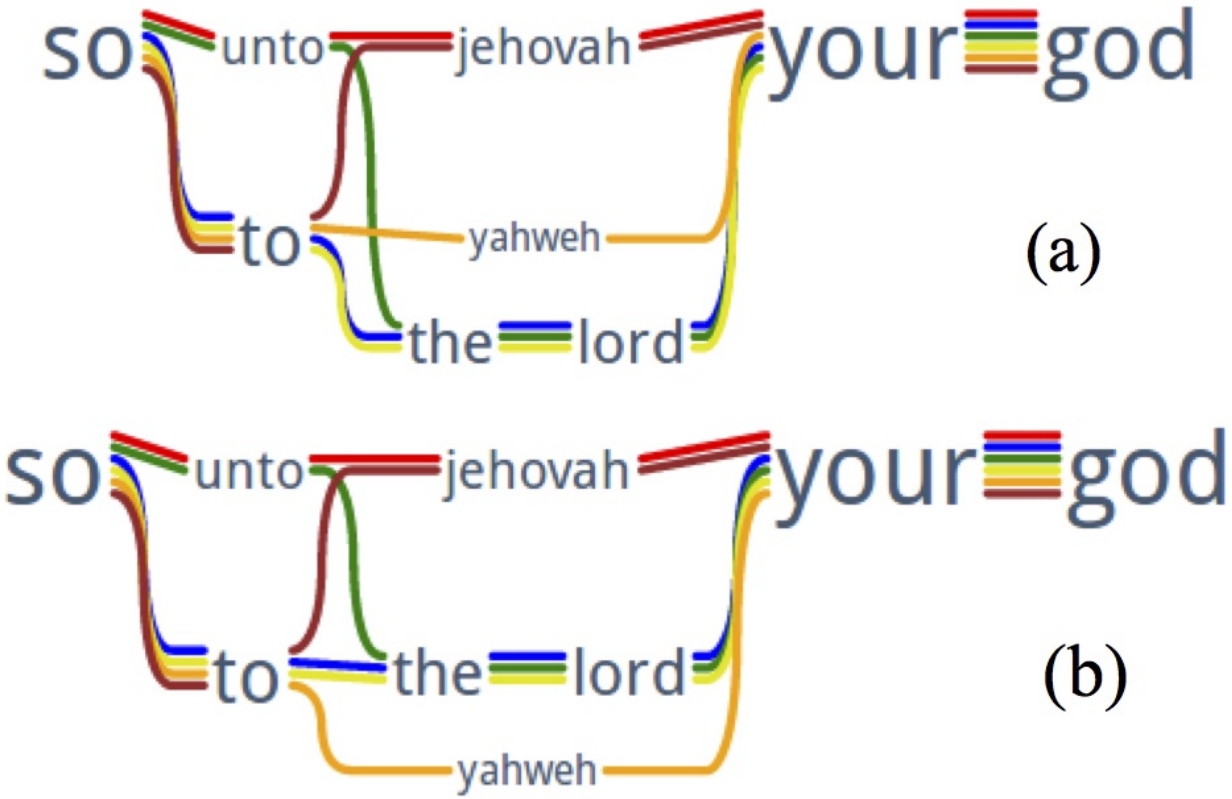
Source: "Design Considerations for Optimizing Storyline Visualizations" Tanahashi et al.

Application: Storylines



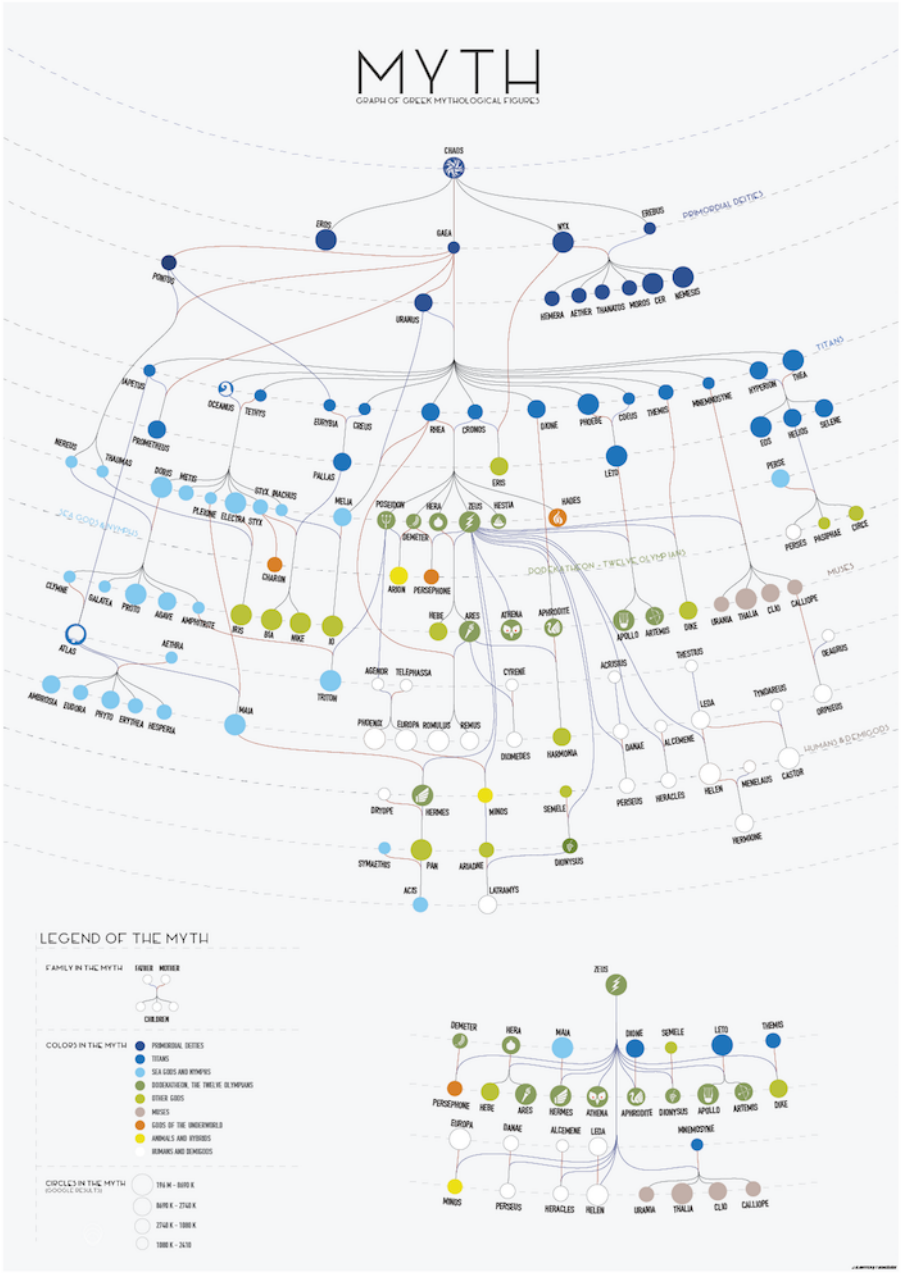
Source: ABC news, Australia

Application: Text-Variant graphs



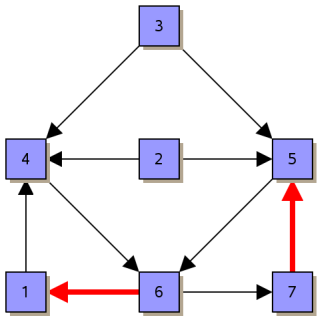
Source: Improving the Layout for Text Variant Graphs Jänicke et al.

Application: Mythological Creatures and Gods



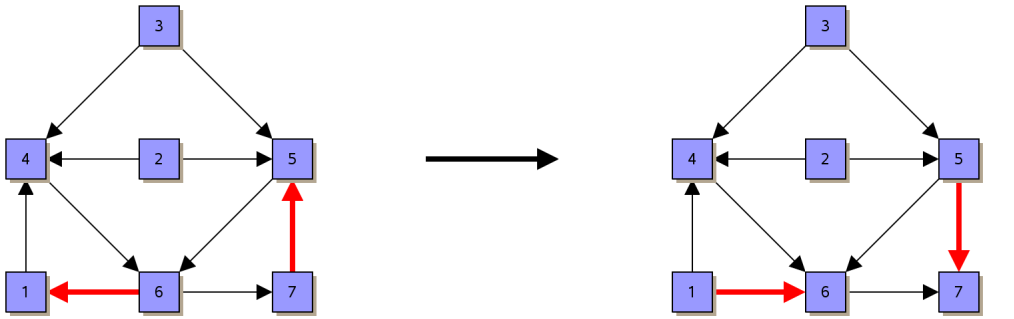
Source: Visualization that won the Graph Drawing contest 2016. Klawitter&Mchedlidze

Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



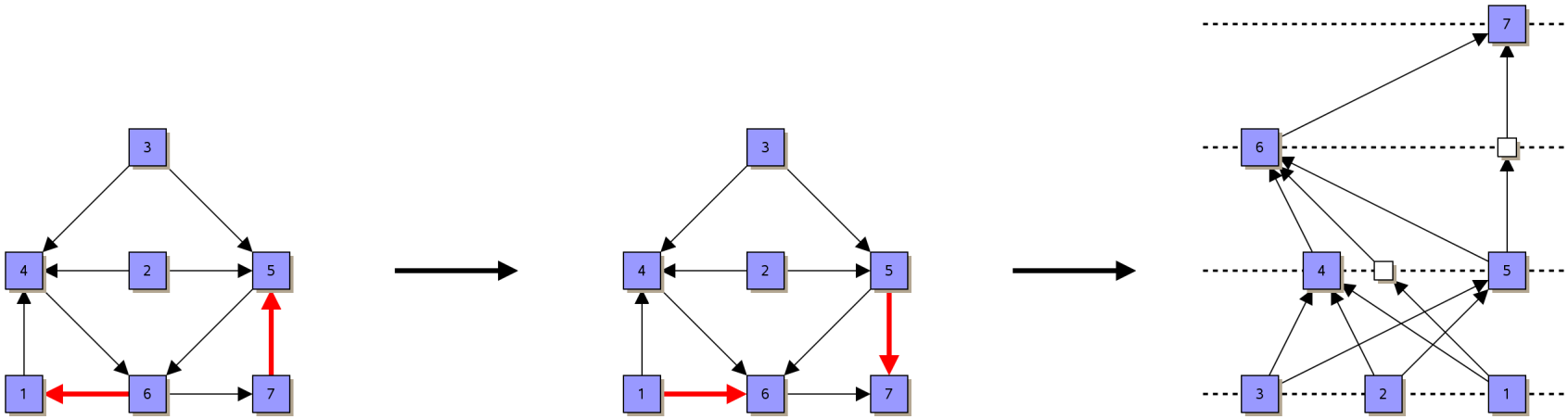
given

Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)

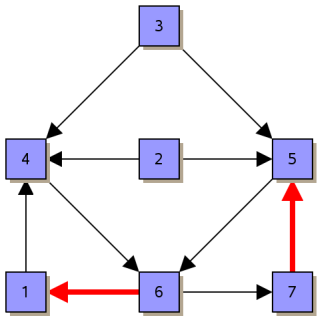


given resolve cycles

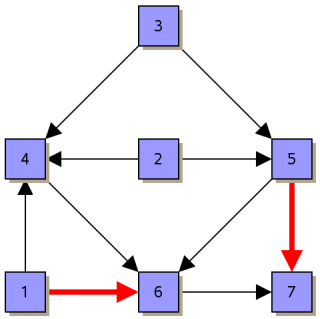
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



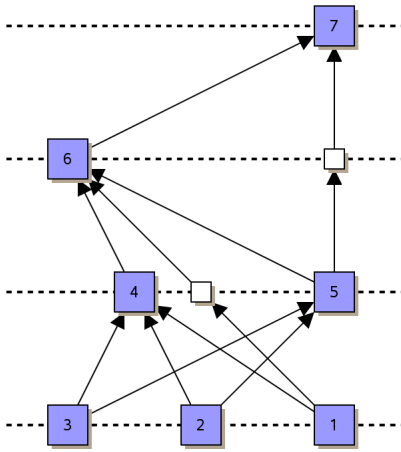
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



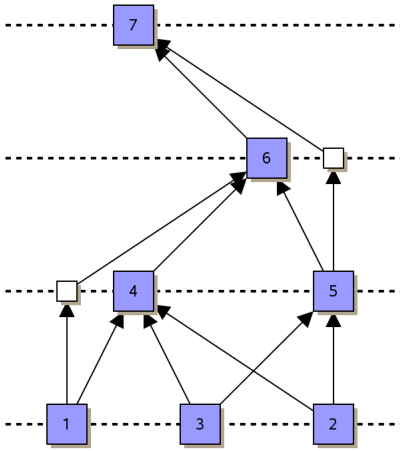
given



resolve cycles

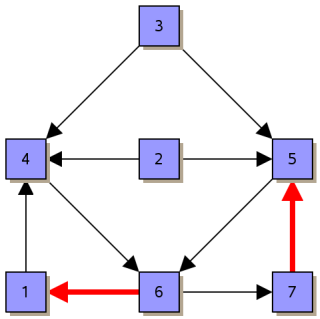


layer assignement

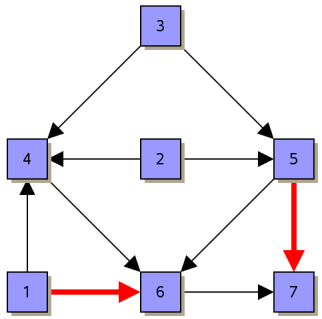


crossing minimization

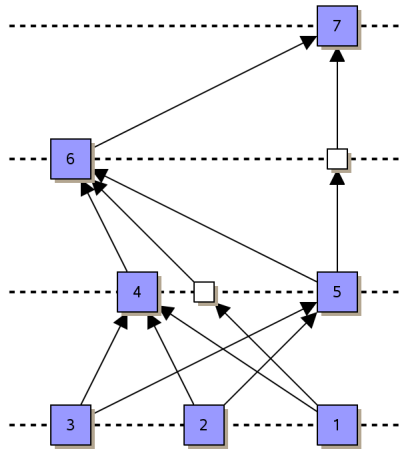
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



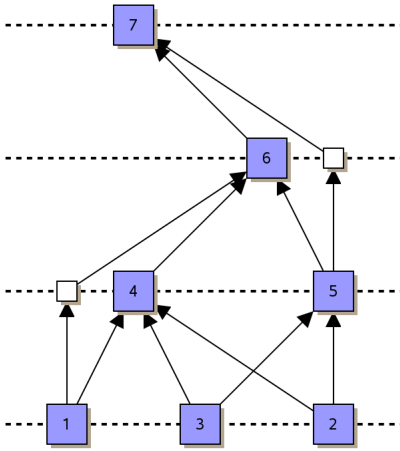
given



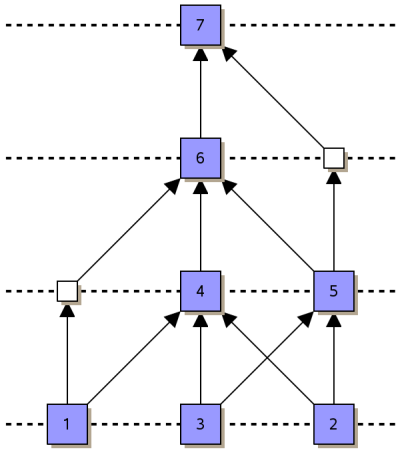
resolve cycles



layer assignement

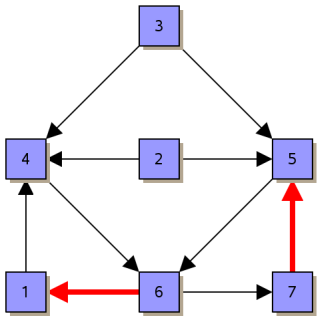


crossing minimization

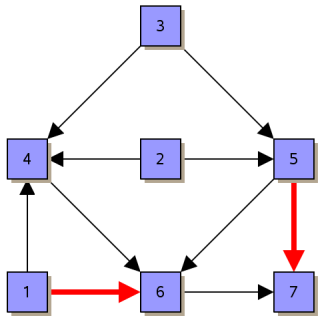


node positioning

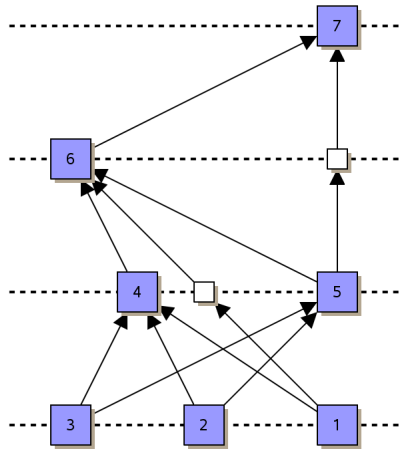
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



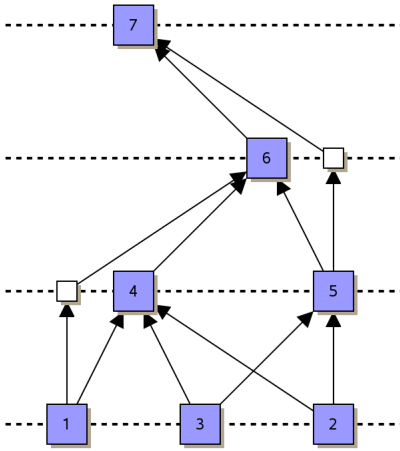
given



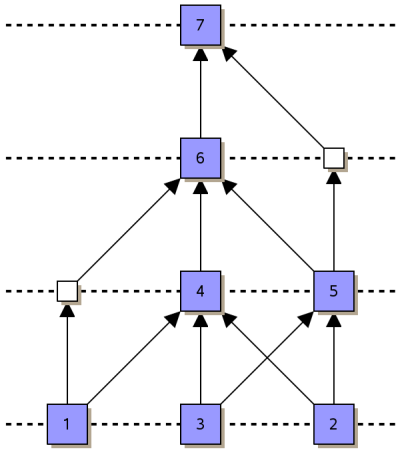
resolve cycles



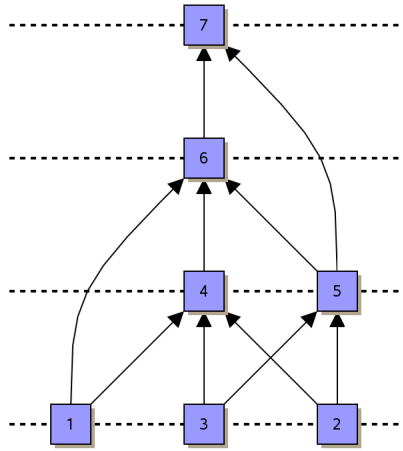
layer assignement



crossing minimization

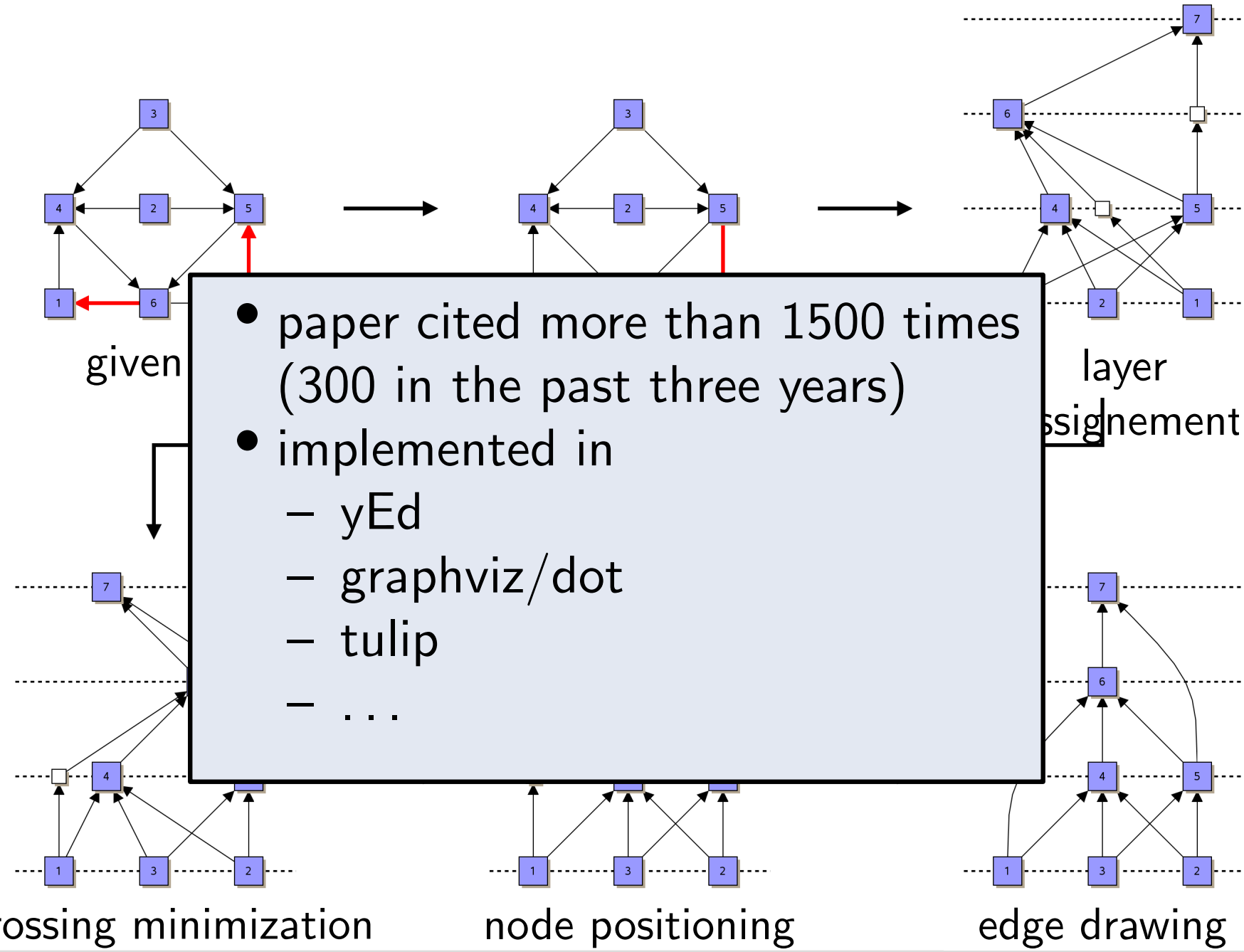


node positioning



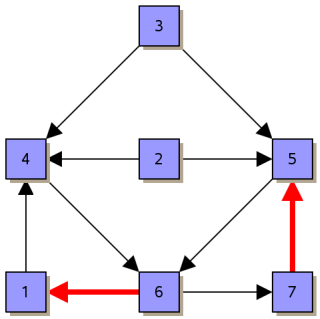
edge drawing

Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)

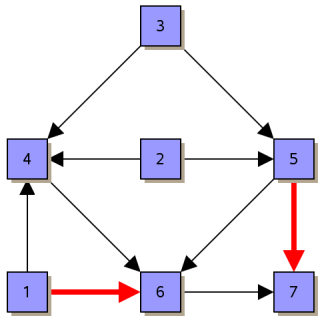


- paper cited more than 1500 times (300 in the past three years)
- implemented in
  - yEd
  - graphviz/dot
  - tulip
  - ...

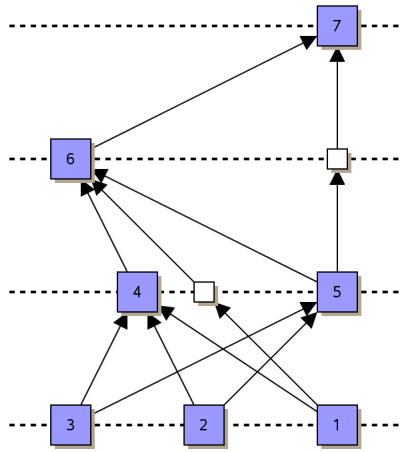
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



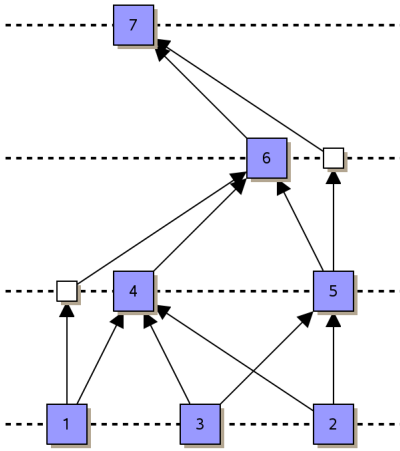
given



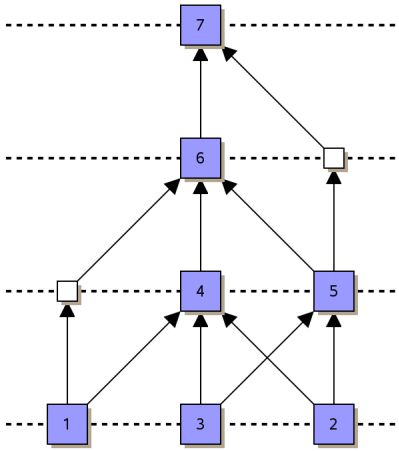
resolve cycles



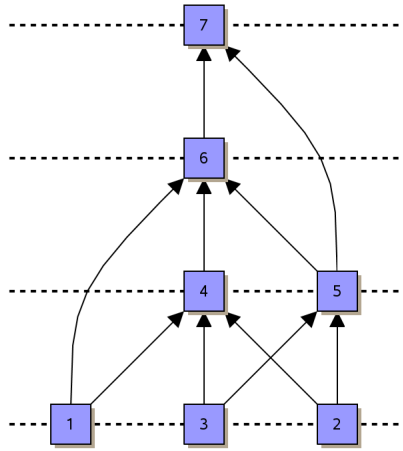
layer assignement



crossing minimization

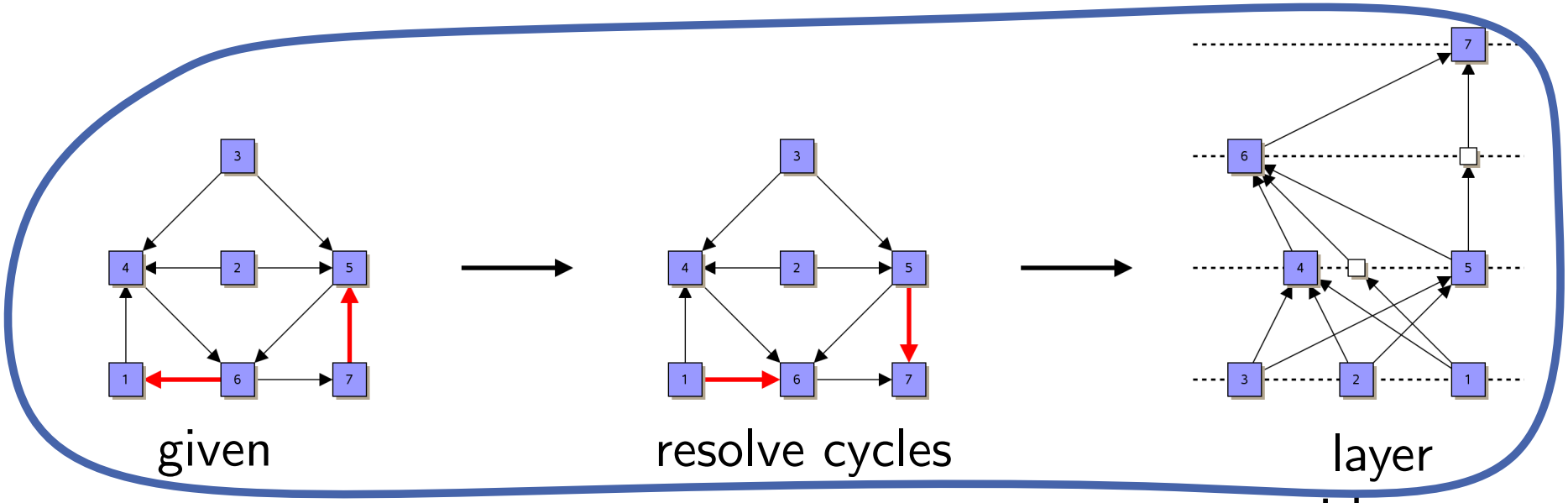


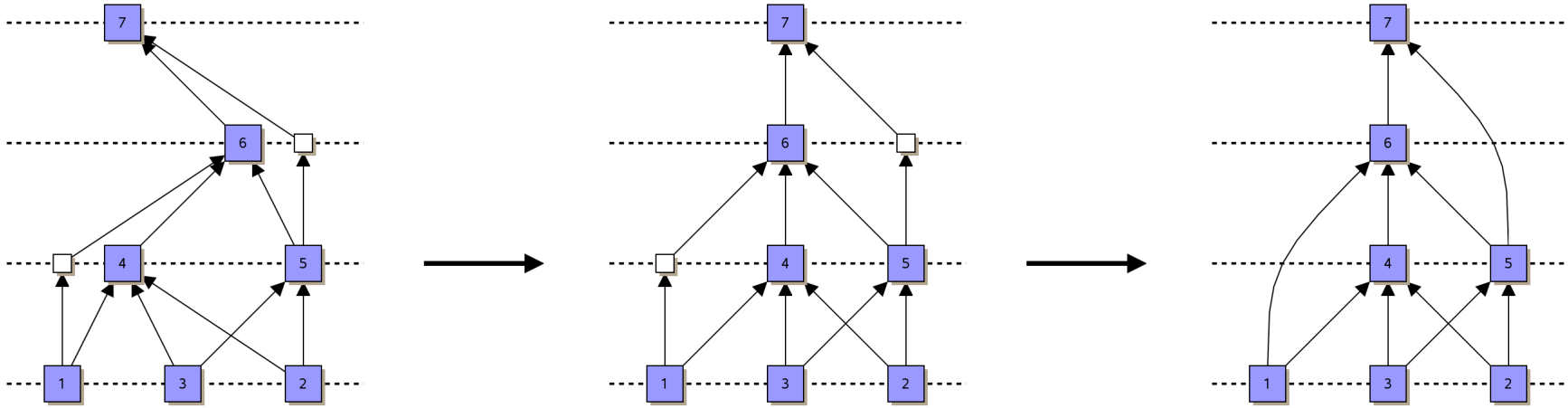
node positioning



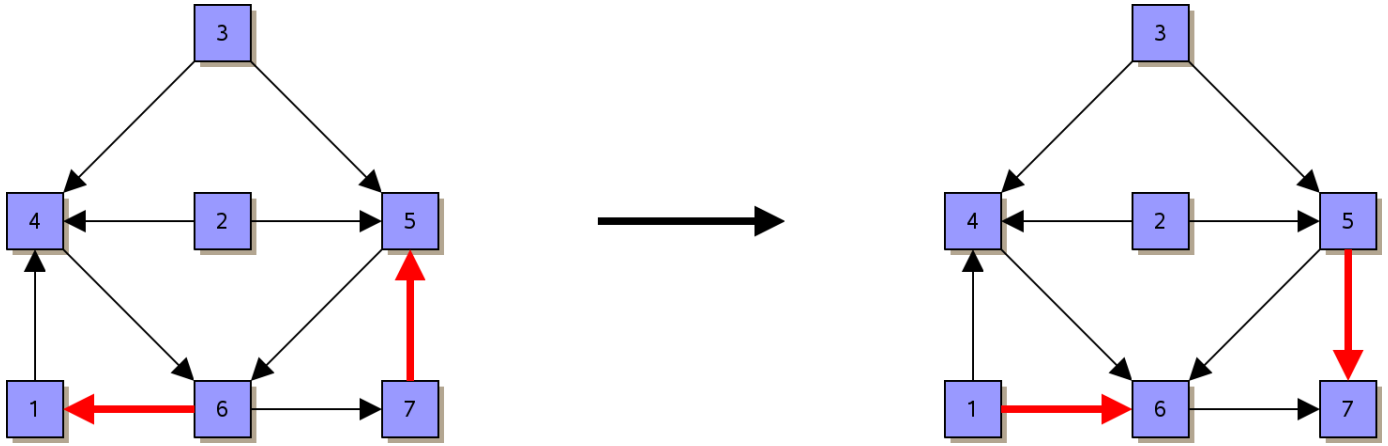
edge drawing

Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)





Step 1: Resolve Cycles



Feedback Arc Set

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

Feedback Arc Set

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

Maximum Acyclic Subgraph

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

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

Feedback Arc Set

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

Maximum Acyclic Subgraph

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

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

Minimum Feedback Arc Set (FAS)

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

Find: $A_f \subset A$, with $D_f = (V, A \setminus A_f)$ acyclic with minimum $|A_f|$

Feedback Arc Set

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

Maximum Acyclic Subgraph

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

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

Minimum Feedback Arc Set (FAS)

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

Find: $A_f \subset A$, with $D_f = (V, A \setminus A_f)$ acyclic with minimum $|A_f|$

Minimum Feedback Set (FS)

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

Find: $A_f \subset A$, with $D_f = (V, A \setminus A_f \cup \text{rev}(A_f))$ acyclic with minimum $|A_f|$

Feedback Arc Set

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

Maximum Acyclic Subgraph

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

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

Minimum Feedback Arc Set (FAS)

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

Find: $A_f \subset A$, with $D_f = (V, A \setminus A_f)$ acyclic with minimum $|A_f|$

Minimum Feedback Set (FS)

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

Find: $A_f \subset A$, with $D_f = (V, A \setminus A_f \cup \text{rev}(A_f))$ acyclic with minimum $|A_f|$

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**

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

  else

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

  remove v and $N(v)$ from D.

return (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 (V, A')

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

- $D' = (V, A')$ is a DAG
- $A \setminus A'$ is a feedback arc set

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 (V, A')

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

- $D' = (V, A')$ is a DAG
- $A \setminus A'$ is a feedback arc set

Work with your neighbour(s) and then share

Why D' does not contain cycles?

What one can say about $|A'|$ in terms of $|V|$?

7 min

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 (V, A')

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

Task togo:

- Is $D'' = (V, A' \cup \text{rev}(A \setminus A'))$ acyclic?
- What is the running time?

- $D' = (V, A')$ is a DAG
- $A \setminus A'$ is a feedback arc set

Work with your neighbour(s) and then share

Why D' does not contain cycles?
What one can say about $|A'|$ in terms of $|V|$?

7 min

Heuristic 2 (Eades, Lin, Smyth 1993)

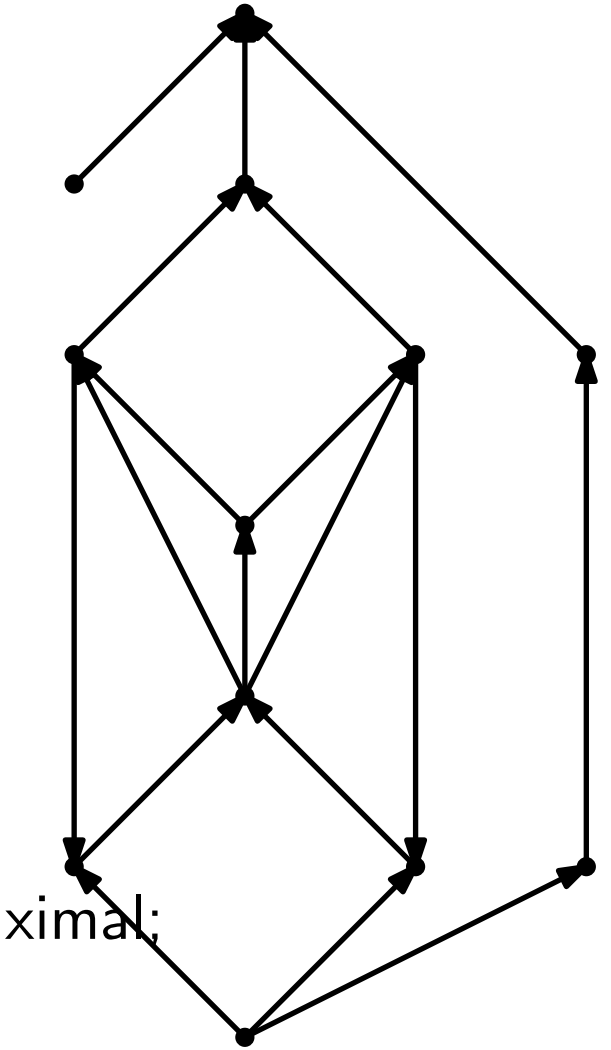
1 $A' := \emptyset$;

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

3 | **while** in V exists a sink v **do**

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

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



ximal;

Heuristic 2 (Eades, Lin, Smyth 1993)

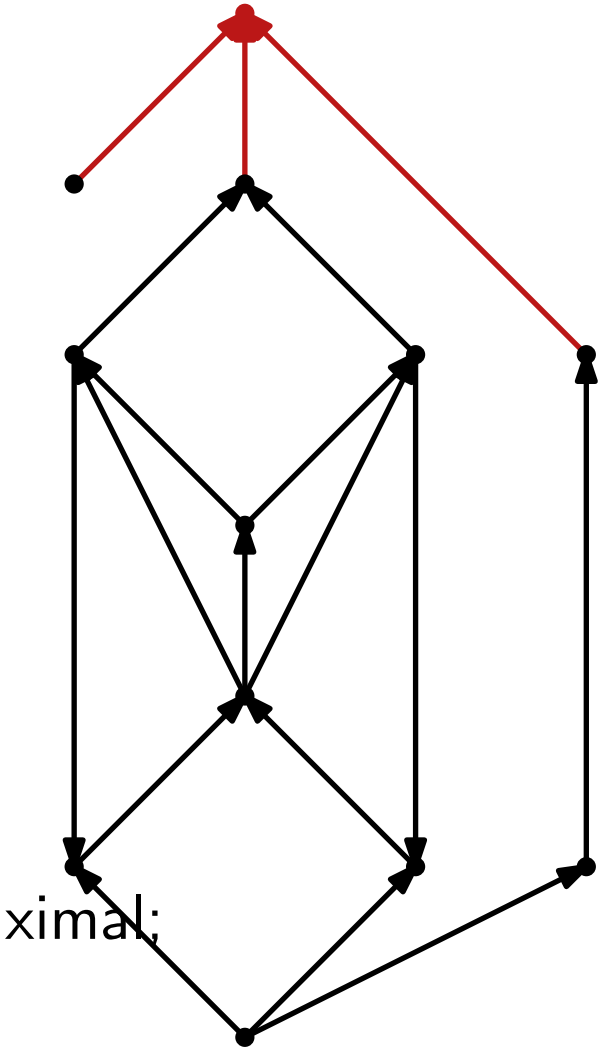
1 $A' := \emptyset$;

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

3 | **while** in V exists a sink v **do**

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

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



ximal;

Heuristic 2 (Eades, Lin, Smyth 1993)

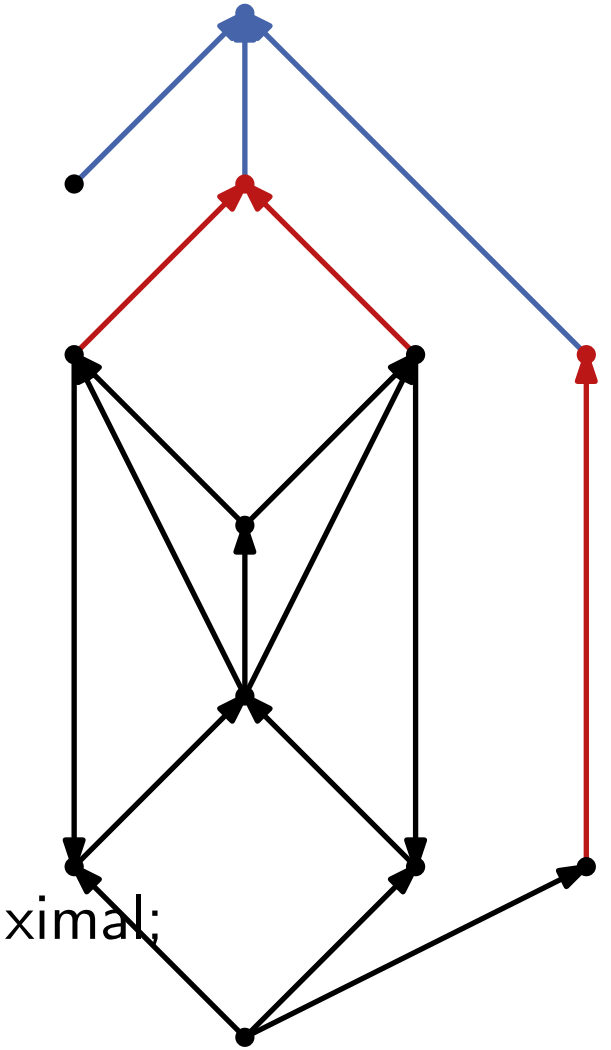
1 $A' := \emptyset$;

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

3 | **while** in V exists a sink v **do**

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

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



ximal;

Heuristic 2 (Eades, Lin, Smyth 1993)

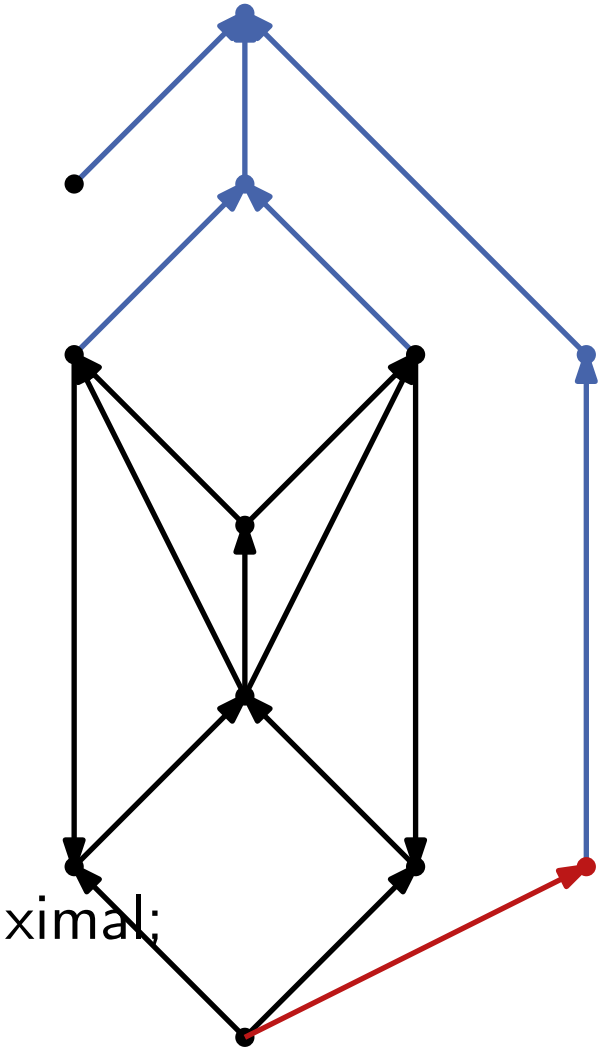
1 $A' := \emptyset$;

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

3 | **while** in V exists a sink v **do**

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

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



ximal;

Heuristic 2 (Eades, Lin, Smyth 1993)

1 $A' := \emptyset$;

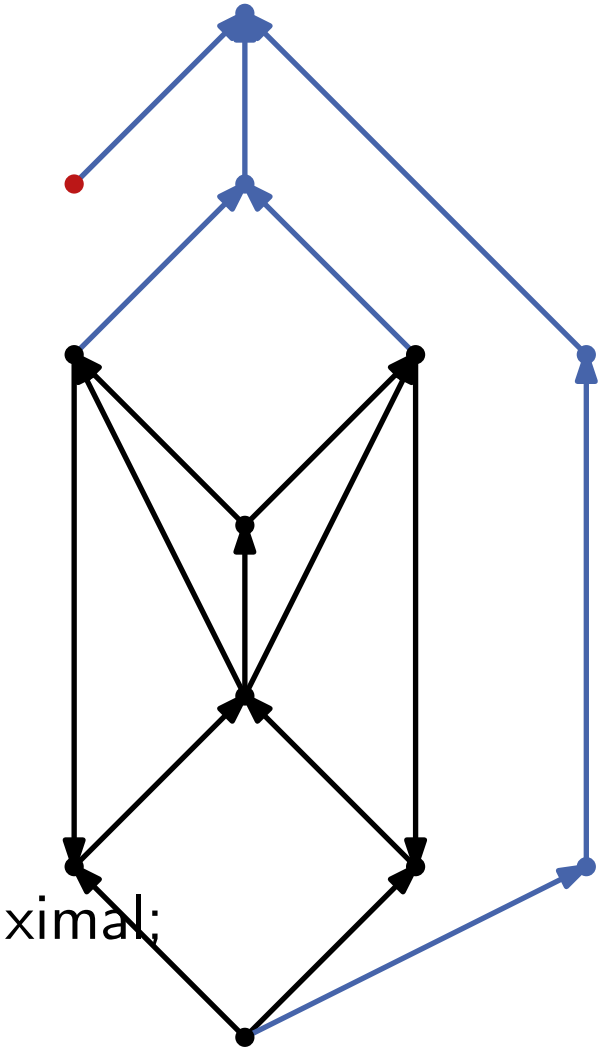
2 **while** $V \neq \emptyset$ **do**

3 | **while** in V exists a sink v **do**

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

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

6 | Remove all isolated node from V: $\{V, n, m\}_{\text{iso}}$



ximal;

Heuristic 2 (Eades, Lin, Smyth 1993)

1 $A' := \emptyset$;

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

3 | **while** in V exists a sink v **do**

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

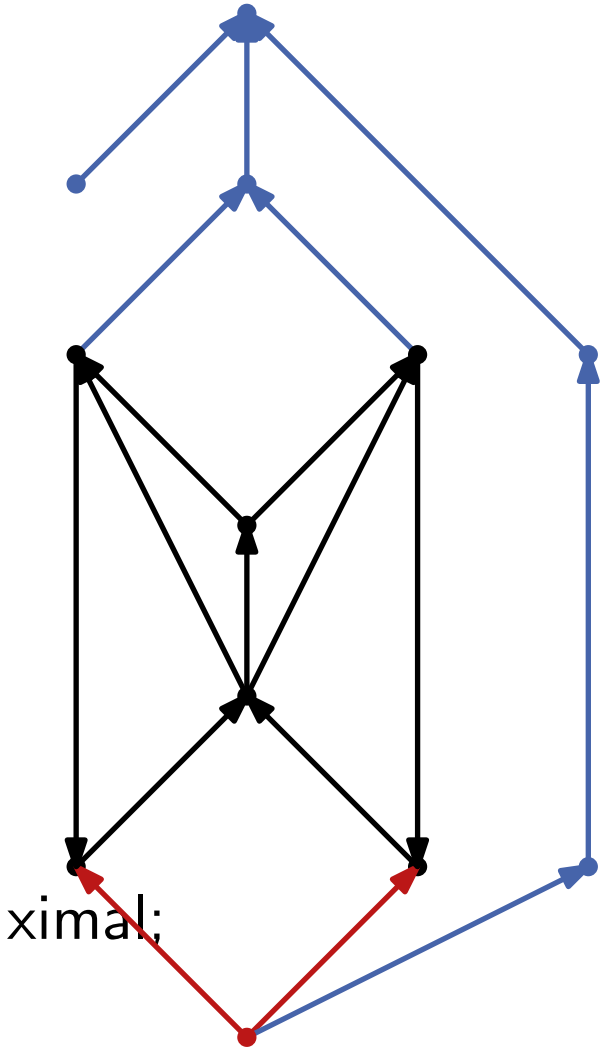
5 | | remove v and $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$

6 | Remove all isolated node from V: $\{V, n, m\}_{\text{iso}}$

7 | **while** in V exists a source v **do**

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

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



ximal;

Heuristic 2 (Eades, Lin, Smyth 1993)

1 $A' := \emptyset$;

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

3 **while** in V exists a sink v **do**

4 $A' \leftarrow A' \cup N^{\leftarrow}(v)$

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

6 Remove all isolated node from V: $\{V, n, m\}_{\text{iso}}$

7 **while** in V exists a source v **do**

8 $A' \leftarrow A' \cup N^{\rightarrow}(v)$

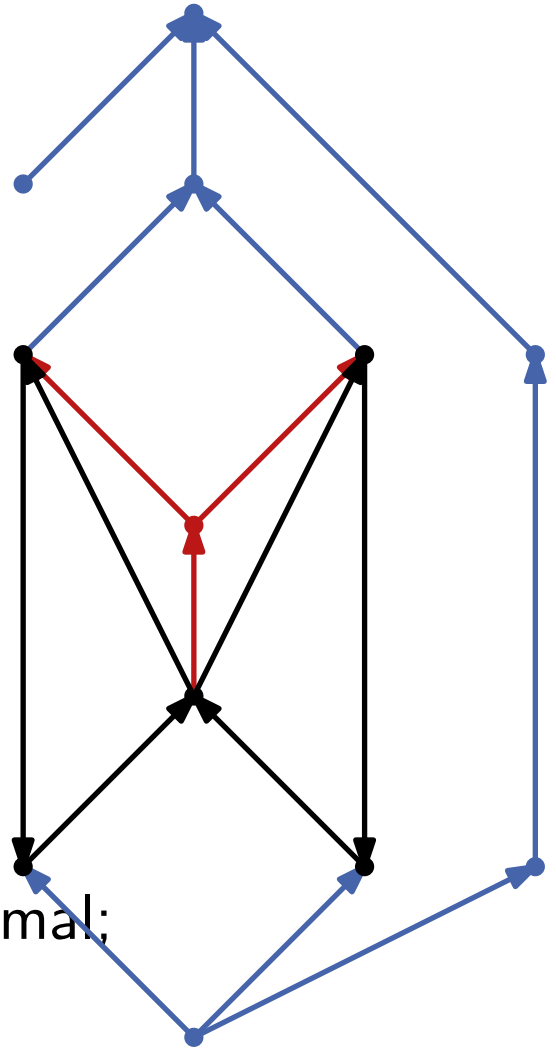
9 remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

11 let $v \in V$ such that $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$ maximal;

12 $A' \leftarrow A' \cup N^{\rightarrow}(v)$

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



Heuristic 2 (Eades, Lin, Smyth 1993)

1 $A' := \emptyset$;

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

3 **while** in V exists a sink v **do**

4 $A' \leftarrow A' \cup N^{\leftarrow}(v)$

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

6 Remove all isolated node from V: $\{V, n, m\}_{\text{iso}}$

7 **while** in V exists a source v **do**

8 $A' \leftarrow A' \cup N^{\rightarrow}(v)$

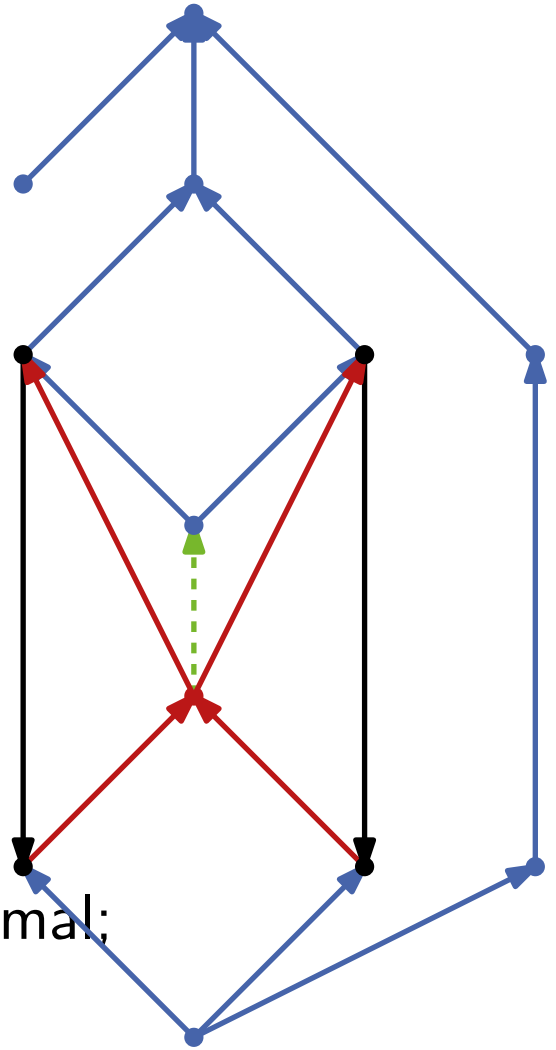
9 remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

11 let $v \in V$ such that $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$ maximal;

12 $A' \leftarrow A' \cup N^{\rightarrow}(v)$

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



Heuristic 2 (Eades, Lin, Smyth 1993)

1 $A' := \emptyset$;

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

3 **while** in V exists a sink v **do**

4 $A' \leftarrow A' \cup N^{\leftarrow}(v)$

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

6 Remove all isolated node from V: $\{V, n, m\}_{\text{iso}}$

7 **while** in V exists a source v **do**

8 $A' \leftarrow A' \cup N^{\rightarrow}(v)$

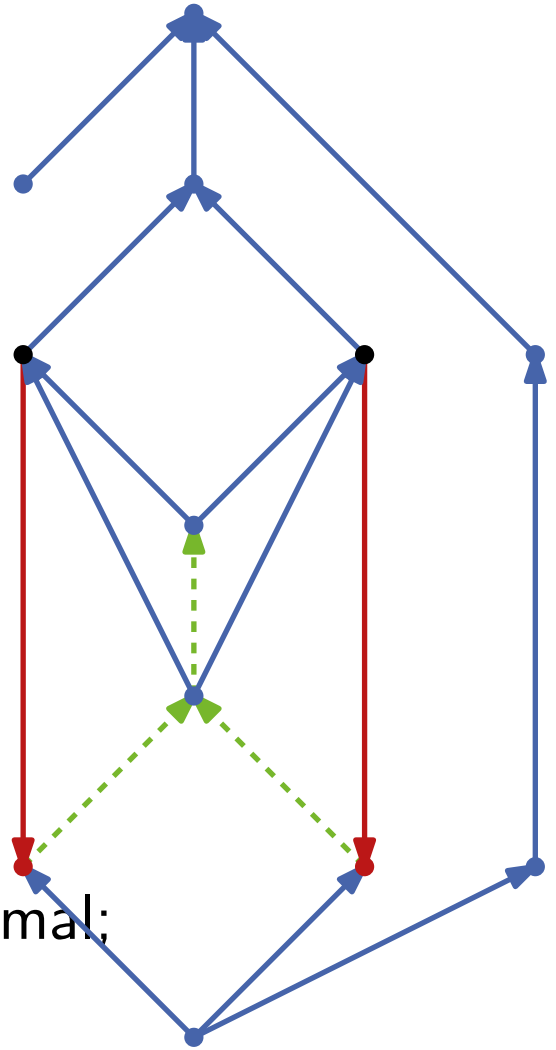
9 remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

11 let $v \in V$ such that $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$ maximal;

12 $A' \leftarrow A' \cup N^{\rightarrow}(v)$

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



Heuristic 2 (Eades, Lin, Smyth 1993)

1 $A' := \emptyset$;

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

3 **while** in V exists a sink v **do**

4 $A' \leftarrow A' \cup N^{\leftarrow}(v)$

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

6 Remove all isolated node from V: $\{V, n, m\}_{\text{iso}}$

7 **while** in V exists a source v **do**

8 $A' \leftarrow A' \cup N^{\rightarrow}(v)$

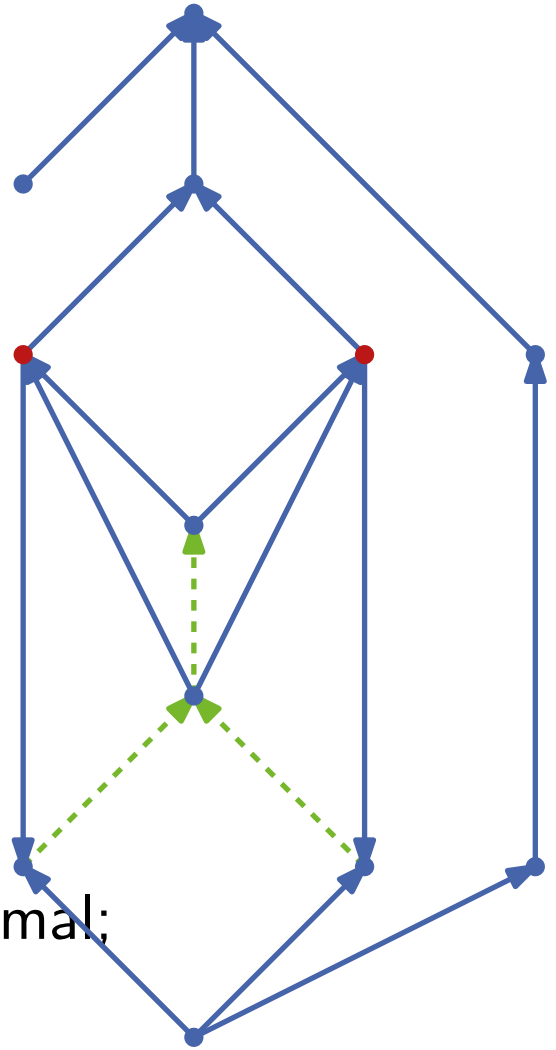
9 remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

11 let $v \in V$ such that $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$ maximal;

12 $A' \leftarrow A' \cup N^{\rightarrow}(v)$

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



Heuristic 2 (Eades, Lin, Smyth 1993)

1 $A' := \emptyset$;

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

3 **while** in V exists a sink v **do**

4 $A' \leftarrow A' \cup N^{\leftarrow}(v)$

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

6 Remove all isolated node from V: $\{V, n, m\}_{\text{iso}}$

7 **while** in V exists a source v **do**

8 $A' \leftarrow A' \cup N^{\rightarrow}(v)$

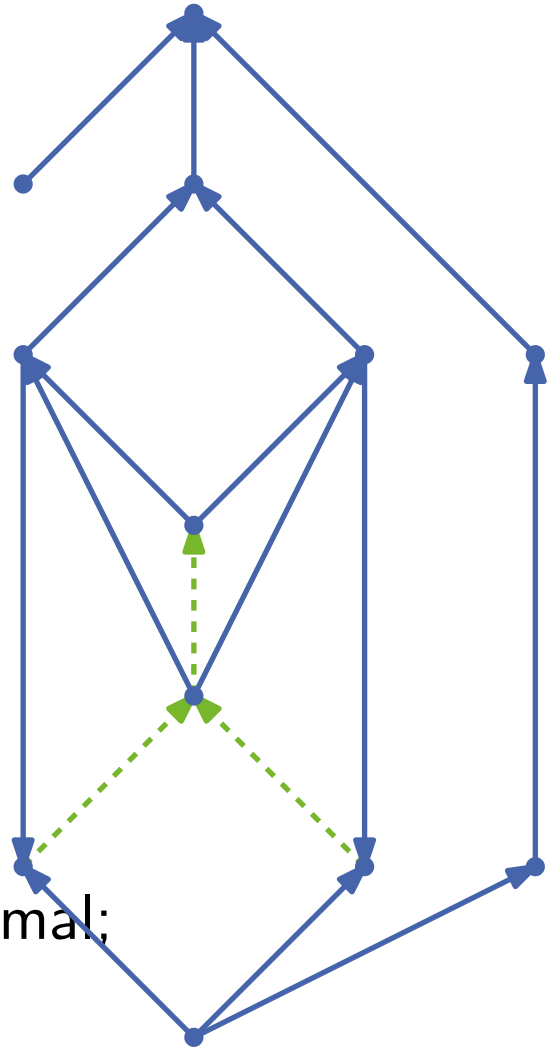
9 remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

11 let $v \in V$ such that $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$ maximal;

12 $A' \leftarrow A' \cup N^{\rightarrow}(v)$

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



Heuristic 2 (Eades, Lin, Smyth 1993)

1 $A' := \emptyset$;

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

3 **while** in V exists a sink v **do**

4 $A' \leftarrow A' \cup N^{\leftarrow}(v)$

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

6 Remove all isolated node from V: $\{V, n, m\}_{\text{iso}}$

7 **while** in V exists a source v **do**

8 $A' \leftarrow A' \cup N^{\rightarrow}(v)$

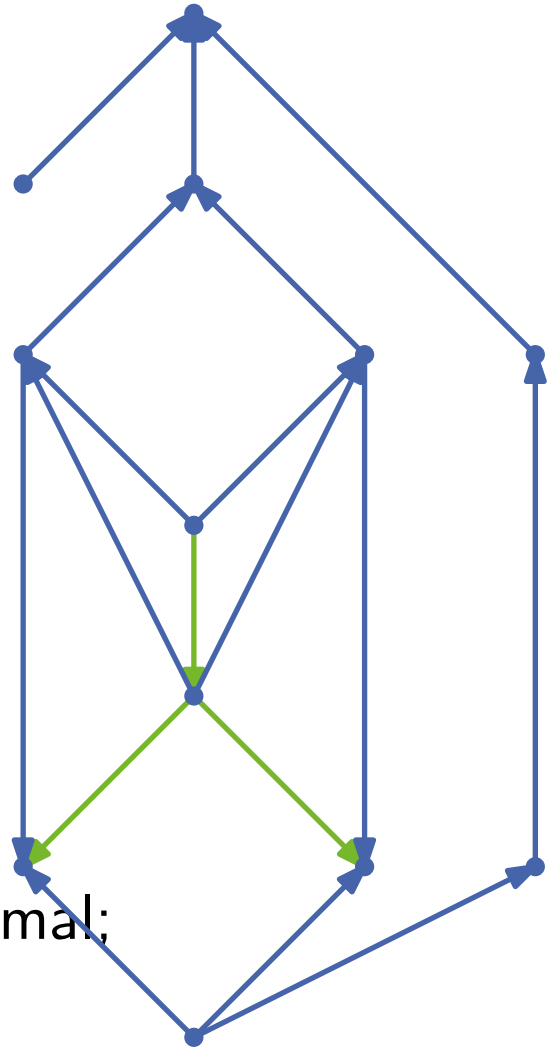
9 remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

11 let $v \in V$ such that $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$ maximal;

12 $A' \leftarrow A' \cup N^{\rightarrow}(v)$

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



Heuristic 2 – Analysis

Theorem 1: Let $D = (V, A)$ be a connected, directed graph without 2-cycles. Heuristic 2 computes a set of edges A' with $|A'| \geq |A|/2 + |V|/6$.
The running time is $O(|A|)$.

Heuristic 2 – Analysis

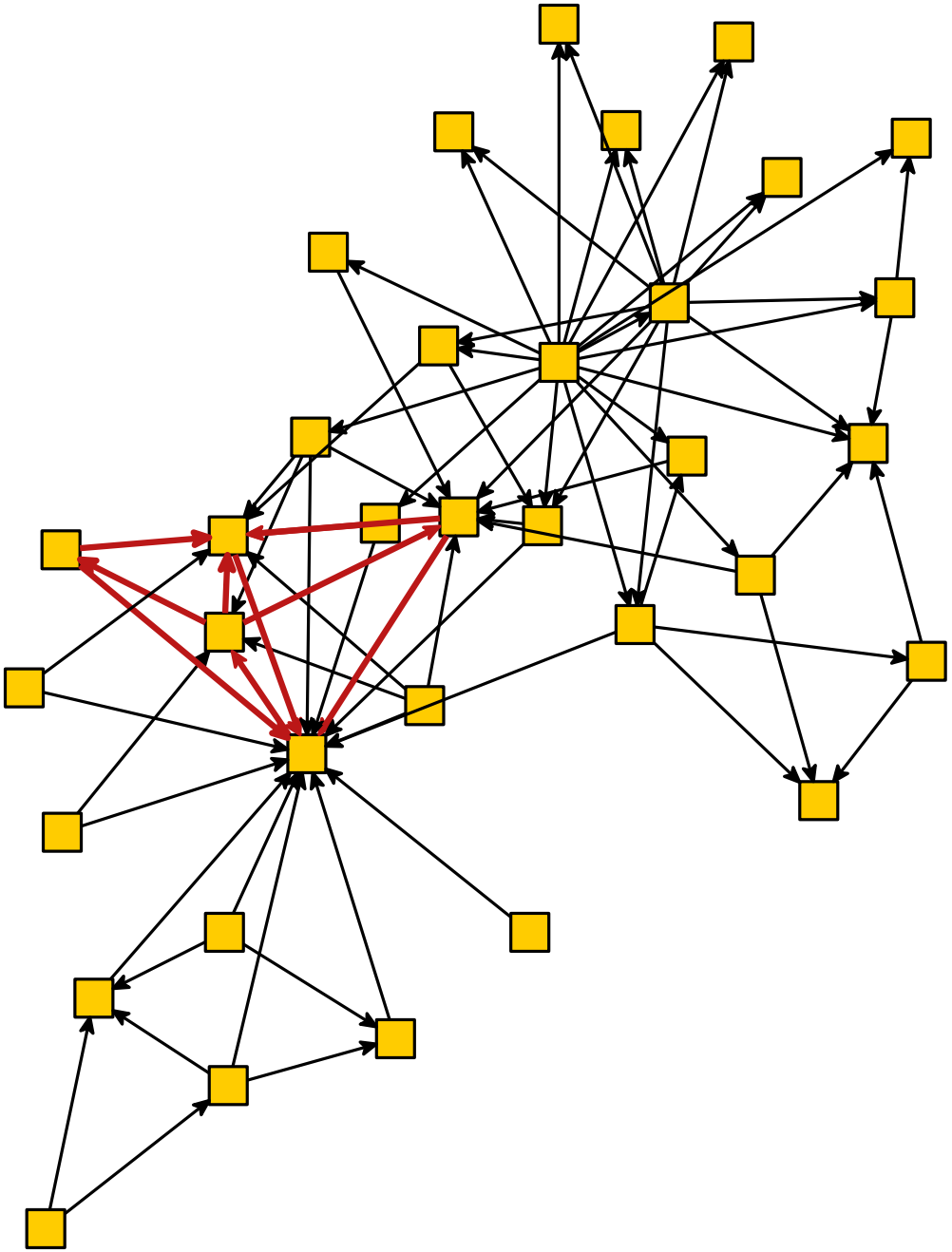
Theorem 1: Let $D = (V, A)$ be a connected, directed graph without 2-cycles. Heuristic 2 computes a set of edges A' with $|A'| \geq |A|/2 + |V|/6$.
The running time is $O(|A|)$.

Exact Solution:

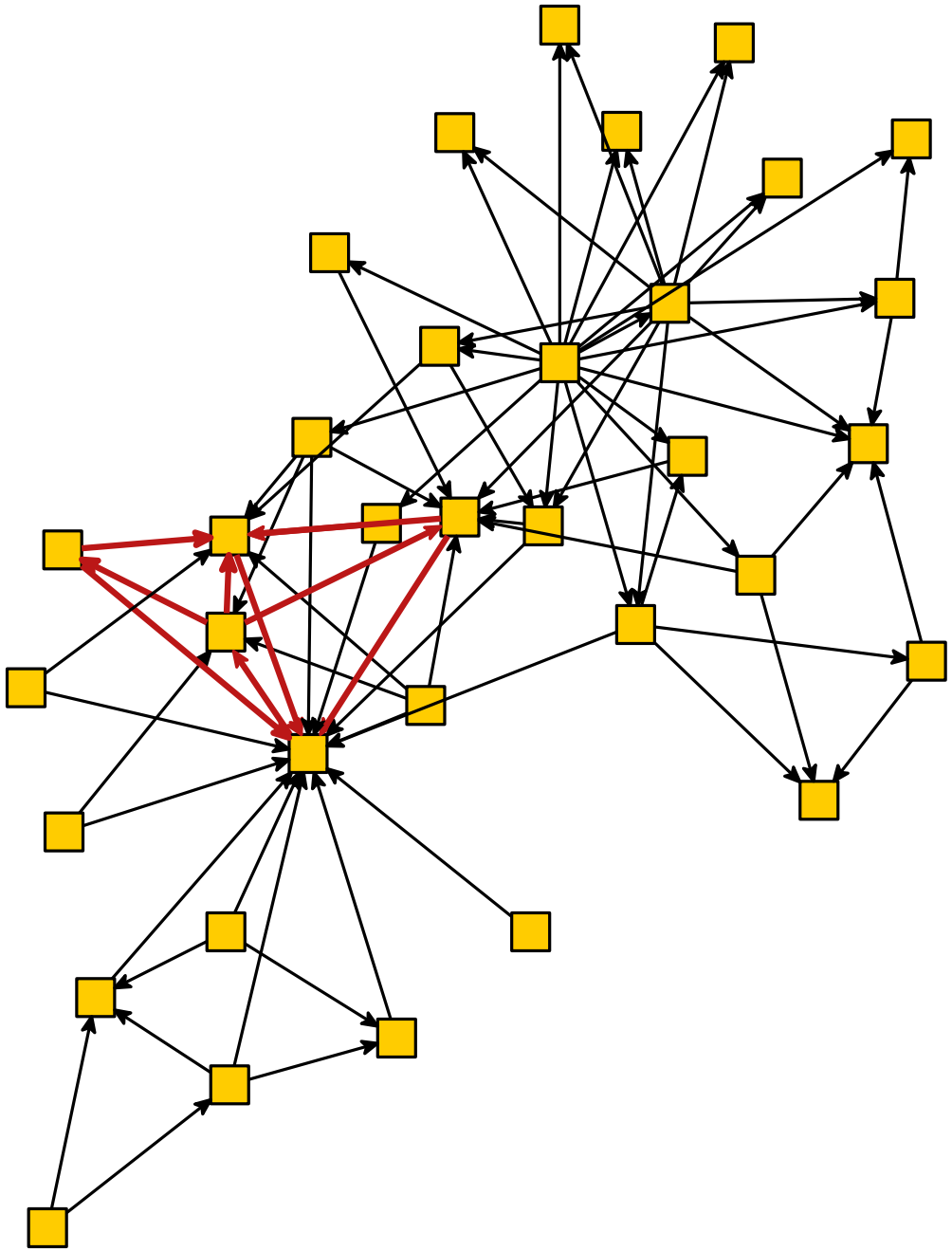
- integer linear programming, using branch-and-cut technique

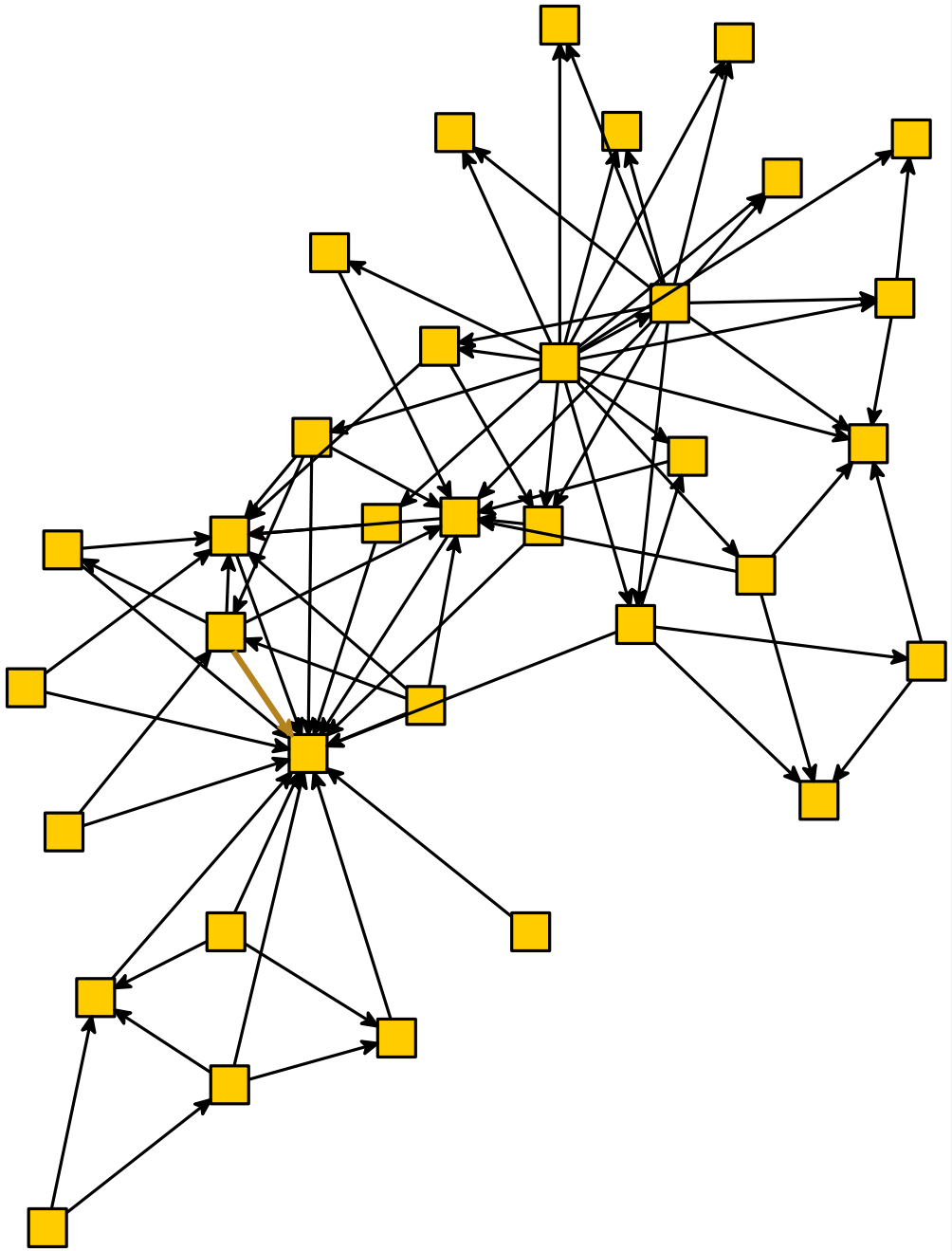
  (Grötschel et al. 1985)

Example

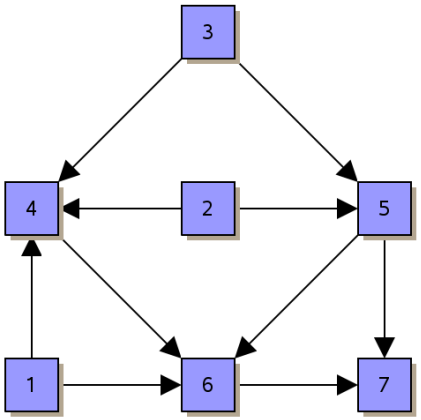


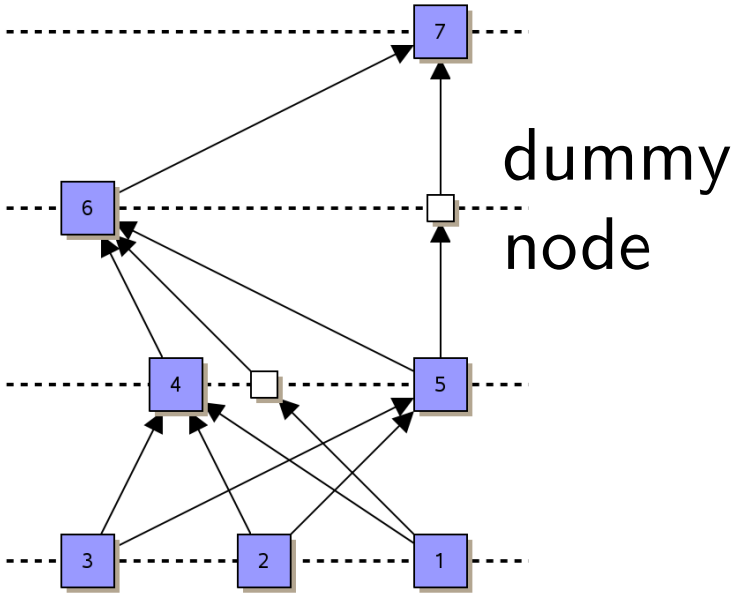
Example



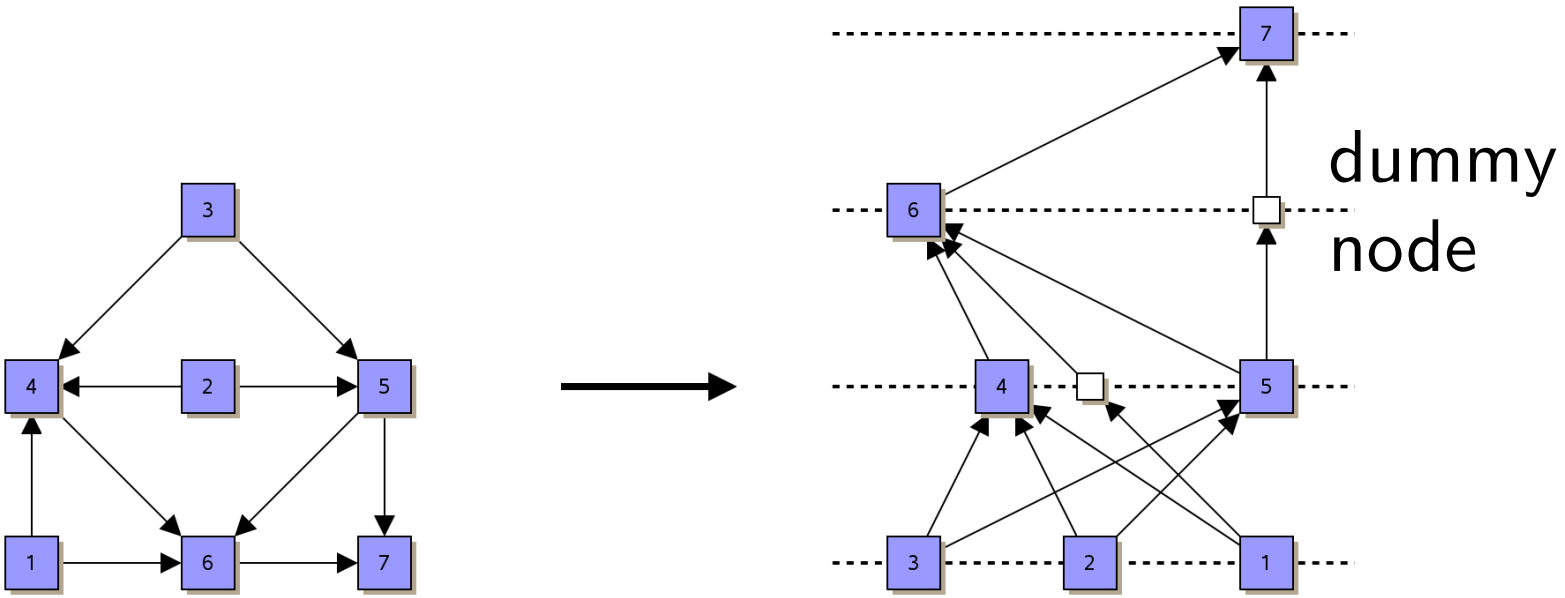


Step 2: Layer Assignement





Step 2: Layer Assignement



Think for a minute and then share

What could we optimize when doing the layer assignment?

1 min

Step 2: Layer Assignement

Given.: directed acyclic graph (DAG) $D = (V, A)$

Find: Partition the vertex set V into disjoint subsets (**layers**) $L_1, \ldots, L_h$ s.t. $(u, v) \in A$, $u \in L_i$, $v \in L_j \Rightarrow i < j$

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

Step 2: Layer Assignement

Given.: directed acyclic graph (DAG) $D = (V, A)$

Find: Partition the vertex set V into disjoint subsets (**layers**) $L_1, \ldots, L_h$ s.t. $(u, v) \in A$, $u \in L_i$, $v \in L_j \Rightarrow i < j$

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

Criteria that we discuss

- minimize the number of layers h ($=$ height of the layouts)
- minimize the total length of edges ($\approx$ number of dummy nodes)
- minimize width, e.g. $\max\{|L_i| \mid 1 \leq i \leq h\}$

Height Optimization

Idea: assign each node v to the layer L_i, where i is the length of the longest simple path from a source to v

- all incomming neighbours lie below v
- the resulting height h is minimized

Height Optimization

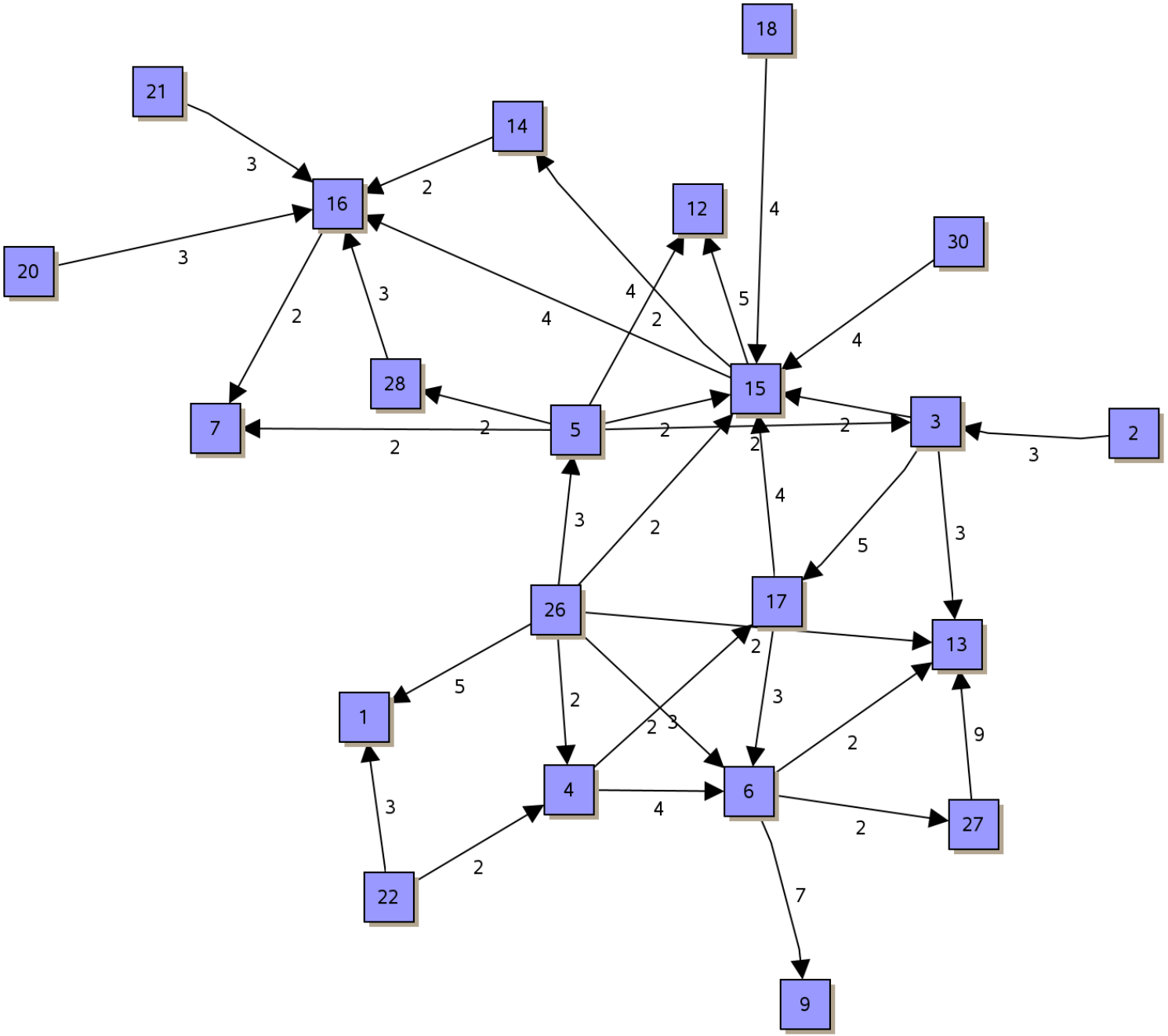
Idea: assign each node v to the layer L_i, where i is the length of the longest simple path from a source to v

- all incomming neighbours lie below v
- the resulting height h is minimized

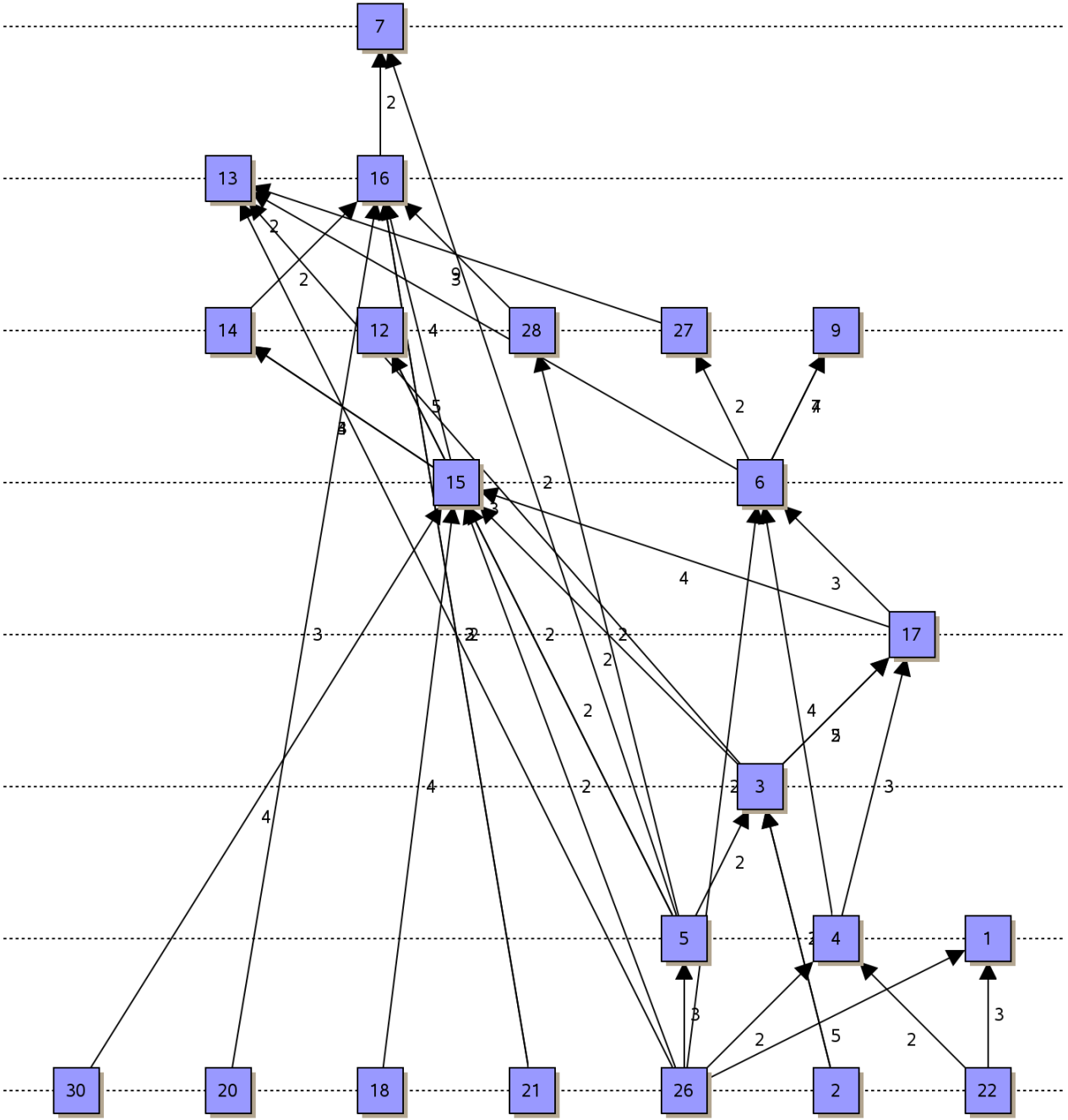
Algorithm

- $L_1 \leftarrow$ the set of sources in D
- set $y(u) \leftarrow \max_{v \in N^{\leftarrow}(u)}\{y(v)\} + 1$

Example


18
21
14
3
2
16
12
4
30
20
3
4
5
2
3
2
4
4
28
15
7
5
3
2
2
2
2
2
3
4
3
2
5
3
17
26
13
2
5
1
2
3
2
3
9
2
4
6
27
4
2
22
2
7
9

Example



Total Edge Length

Karlsruhe Institute of Technology

Can be formulated as an integer linear program:

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

Total Edge Length

Karlsruhe Institute of Technology

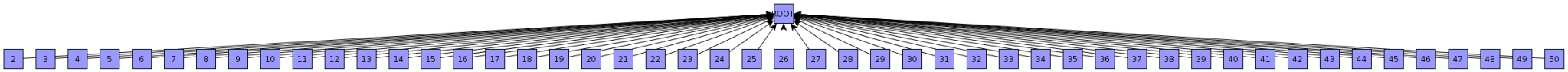
Can be formulated as an integer linear program:

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

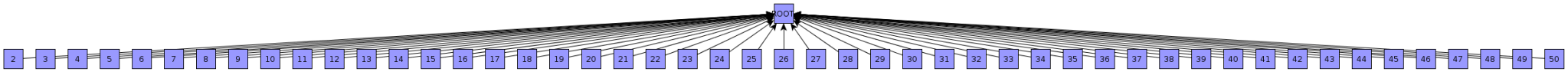
One can show that:

- Constraint-Matrix is **totally unimodular**
- $\Rightarrow$ Solution of the relaxed linear program is integer
- The total edge length can be minimized in a polynomial time

Width of the Layout



Width of the Layout



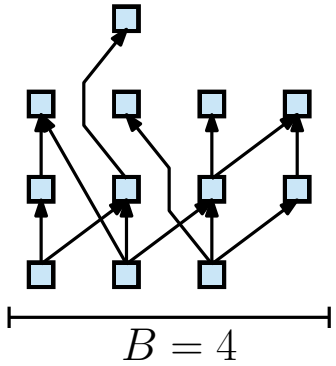
$\rightarrow$ bound the width!

Layer Assignement with Fixed Width

Fixed-Width Layer Assignment

Given: directed acyclic graph $D = (V, A)$, width B

Find: layer assignement $\mathcal{L}$ of minumum height with at most B nodes per layer

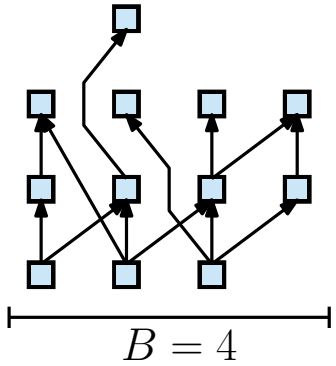


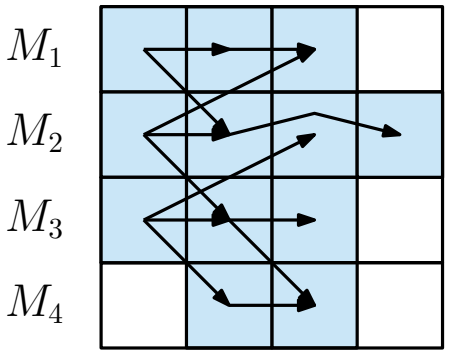
Layer Assignement with Fixed Width

Fixed-Width Layer Assignment

Given: directed acyclic graph $D = (V, A)$, width B

Find: layer assignement $\mathcal{L}$ of minumum height with at most B nodes per layer





$\rightarrow$ equivalent to the following scheduling problem:

Minimum Precedence Constrained Scheduling (MPCS)

Given: n Jobs $J_1, \ldots, J_n$ with identical unit processing time, precedence constraints $J_i < J_k$, and B identical machines

Find: Schedule of minimum length, that satisfies all the precendence constraints

Fixed Width: Complexity

Theorem 2: It is NP-hard to decide, whether for n jobs $J_1, \ldots, J_n$ of identical length, given partial ordering constraints, and number of machinces B, there exists a schedule of height at most T, even if $T = 3$.

Fixed Width: Complexity

Theorem 2: It is NP-hard to decide, whether for n jobs $J_1, \ldots, J_n$ of identical length, given partial ordering constraints, and number of machinces B, there exists a schedule of height at most T, even if $T = 3$.

Corollary: If $\mathcal{P} \neq \mathcal{NP}$, there is no polynomial algorithm for MPCS with approximation factor $< 4/3$.

Fixed Width: Complexity

Theorem 2: It is NP-hard to decide, whether for n jobs $J_1, \ldots, J_n$ of identical length, given partial ordering constraints, and number of machinces B, there exists a schedule of height at most T, even if $T = 3$.

Corollary: If $\mathcal{P} \neq \mathcal{NP}$, there is no polynomial algorithm for MPCS with approximation factor $< 4/3$.

Work with your neighbour(s) and then share

Why the corollary holds?

5 min

Fixed Width: Complexity

Theorem 2: It is NP-hard to decide, whether for n jobs $J_1, \ldots, J_n$ of identical length, given partial ordering constraints, and number of machinces B, there exists a schedule of height at most T, even if $T = 3$.

Corollary: If $\mathcal{P} \neq \mathcal{NP}$, there is no polynomial algorithm for MPCS with approximation factor $< 4/3$.

Work with your neighbour(s) and then share

Why the corollary holds?

5 min

Fixed Width: Algorithm

Theorem 3: There exist an approximation algorithm for MPCS with factor $\leq 2 - \frac{1}{B}$.

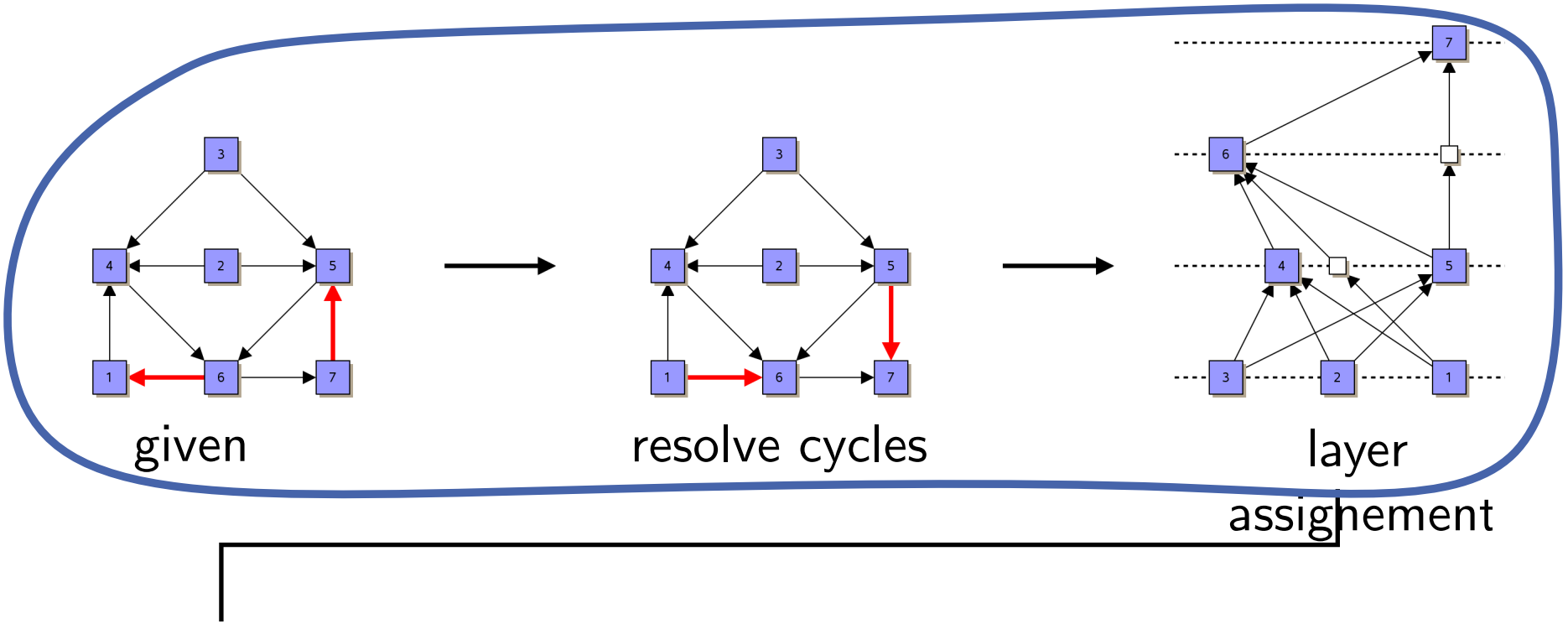
Fixed Width: Algorithm

Theorem 3: There exist an approximation algorithm for MPCS with factor $\leq 2 - \frac{1}{B}$.

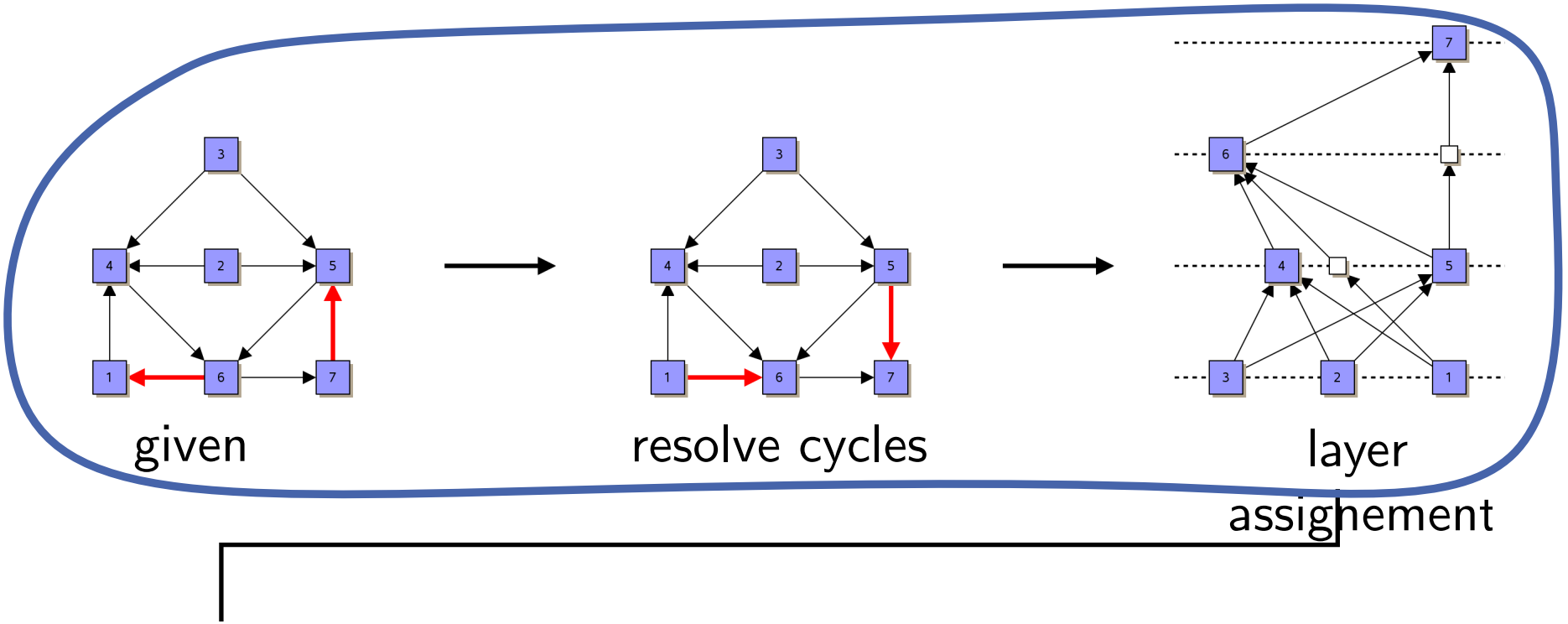
List-Scheduling-Algorithm:

- order jobs arbitrarily as a list $\mathcal{L}$
- when a machine is free, select an allowed job from $\mathcal{L}$; Machine is idle of there is no such job

Summary



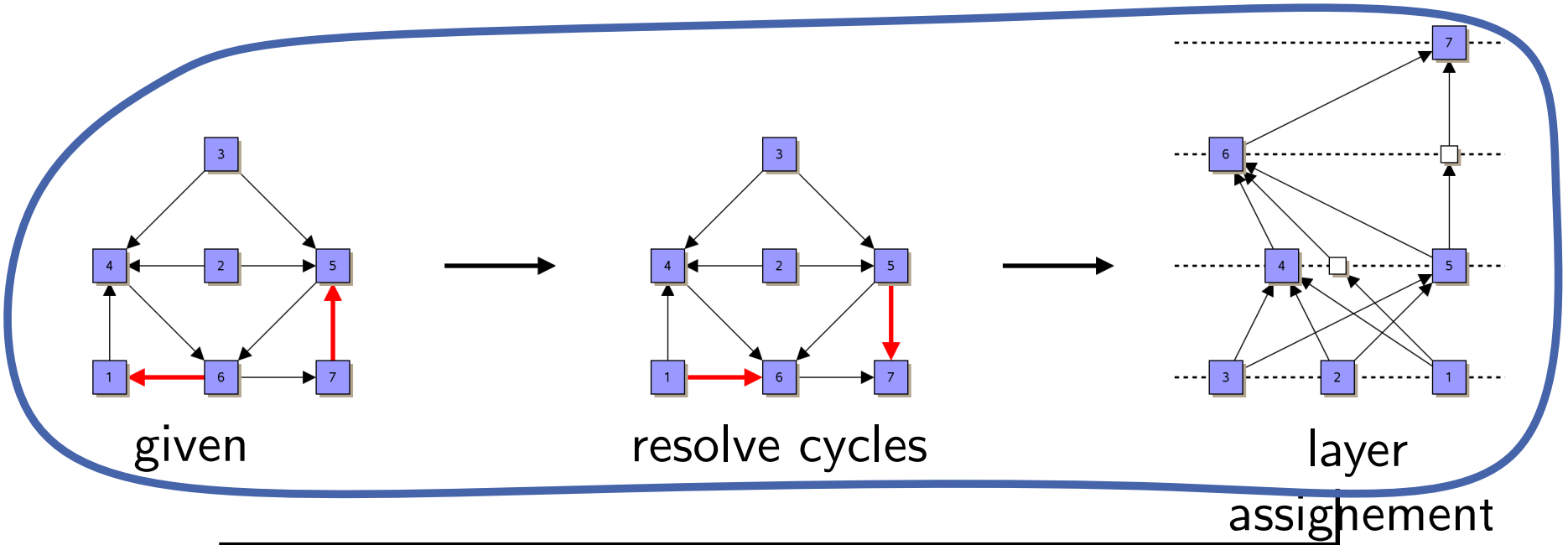
Summary



Resolve cycles

- equivalent to **minimum feedback set** problem, and is NP-hard
- Heuristic with $|A'| \geq |A|/2$
- Heuristic with $|A'| \geq |A|/2 + |V|/6$

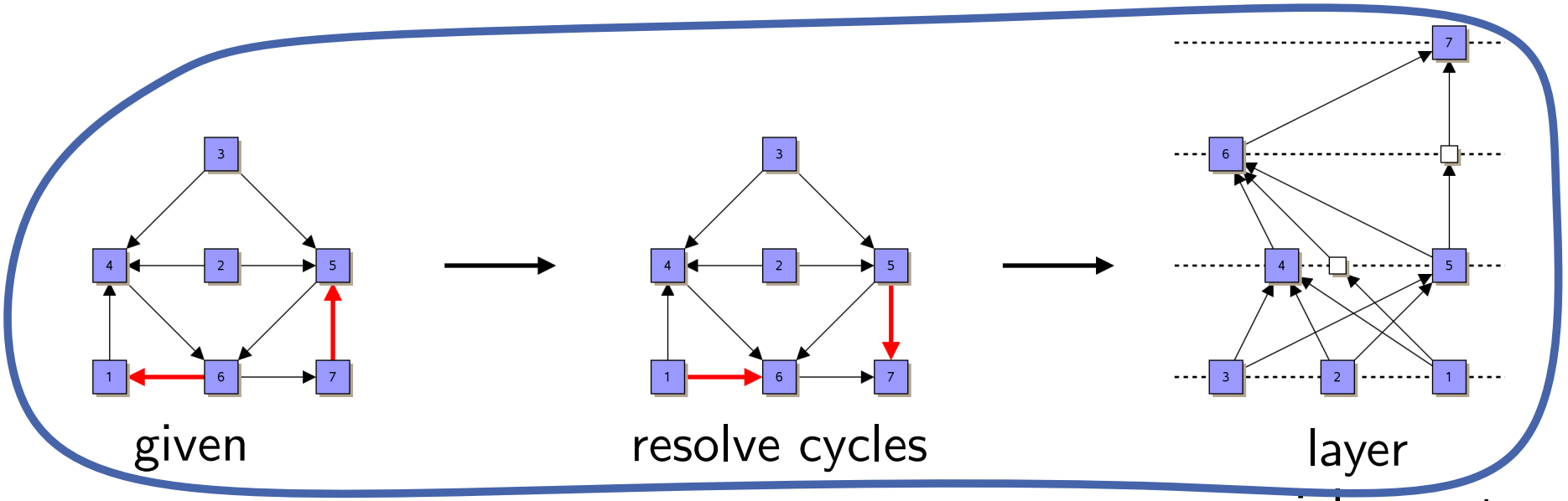
Summary

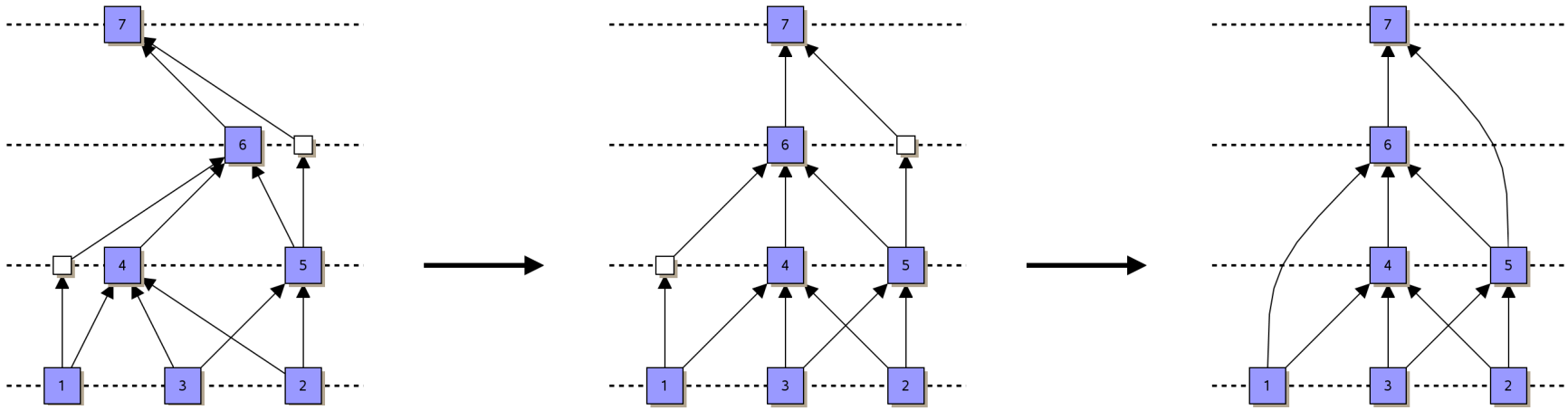


Layer assignement

- Height optimization: topological numbering
- Total edge length: polynomial alg. through integer linear program
- Height optimization with fixed width: equivalent to **MPCS**. NP-hard for 3 levels. Approximation algorithm with factor $\leq 2 - \frac{1}{B}$.

Summary


given
resolve cycles
layer assignement


crossing minimization
node positioning
edge drawing