Algorithms for Graph Visualization

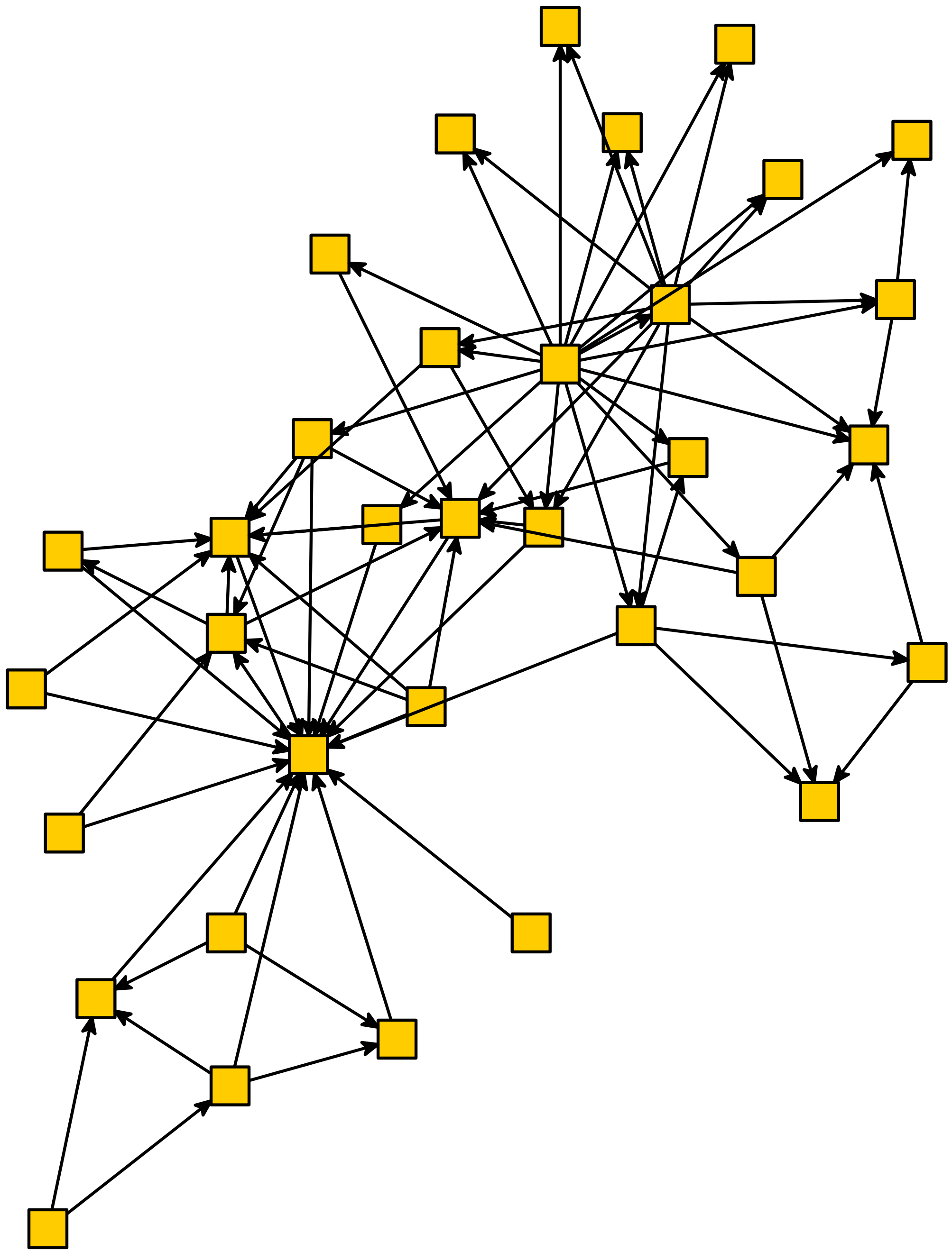
Layered Layout

INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Tamara Mchedlidze

5.12.2016

Example

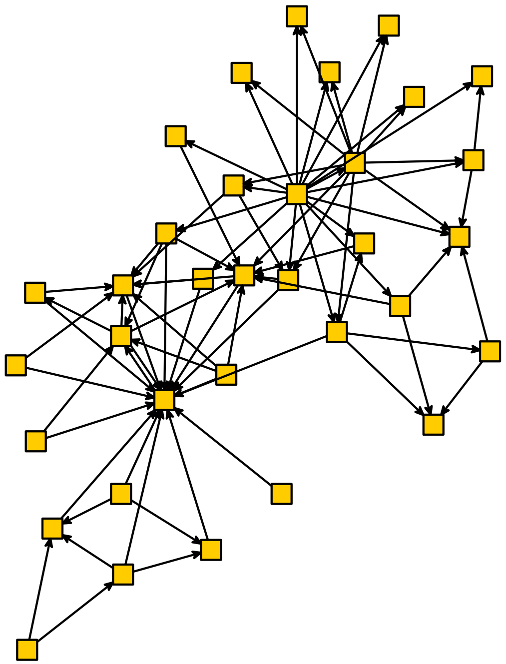


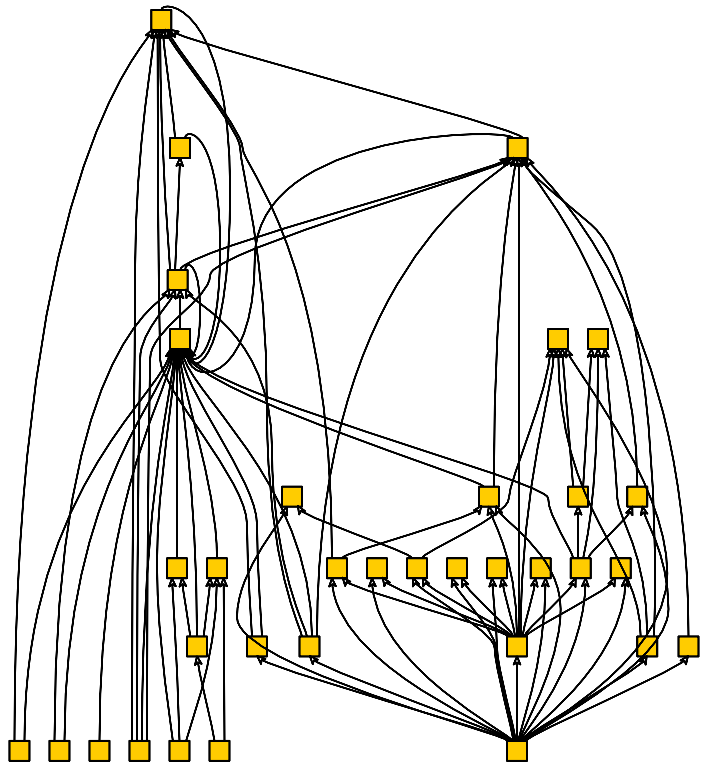
- Which are the properties?
- Which aesthetic ctireria are usefull?

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

Criteria:

- many edges pointing to the same direction
- edges preferably straght and short
- position nodes on (few) horizontal lines
- preferably few edge crossings
- nodes distributed evenly

Layered Layout

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

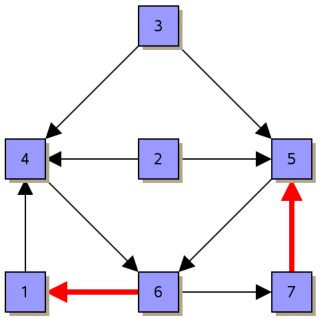
Find: drawing of D that emphasized the hierarchy

Criteria:

- many edges pointing to the same direction
- edges preferably straght and short
- position nodes on (few) horizontal lines
- preferably few edge crossings
- nodes distributed evenly

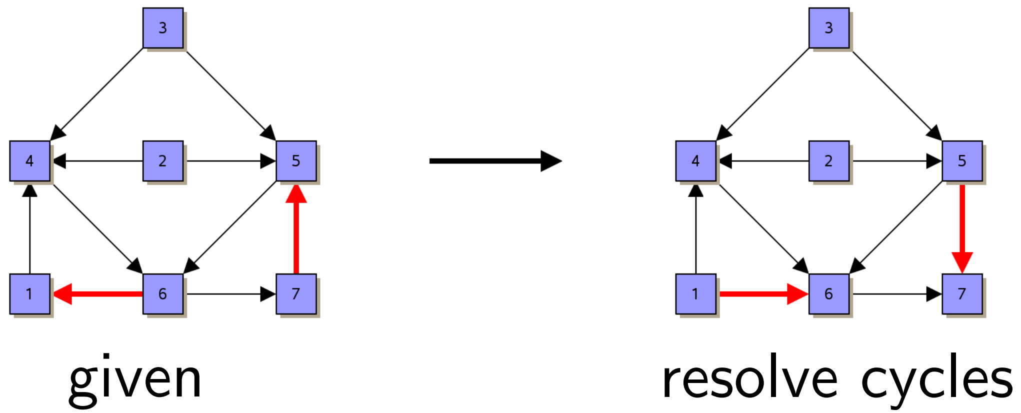
⚠ Optimization criteria partially overlap

Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)

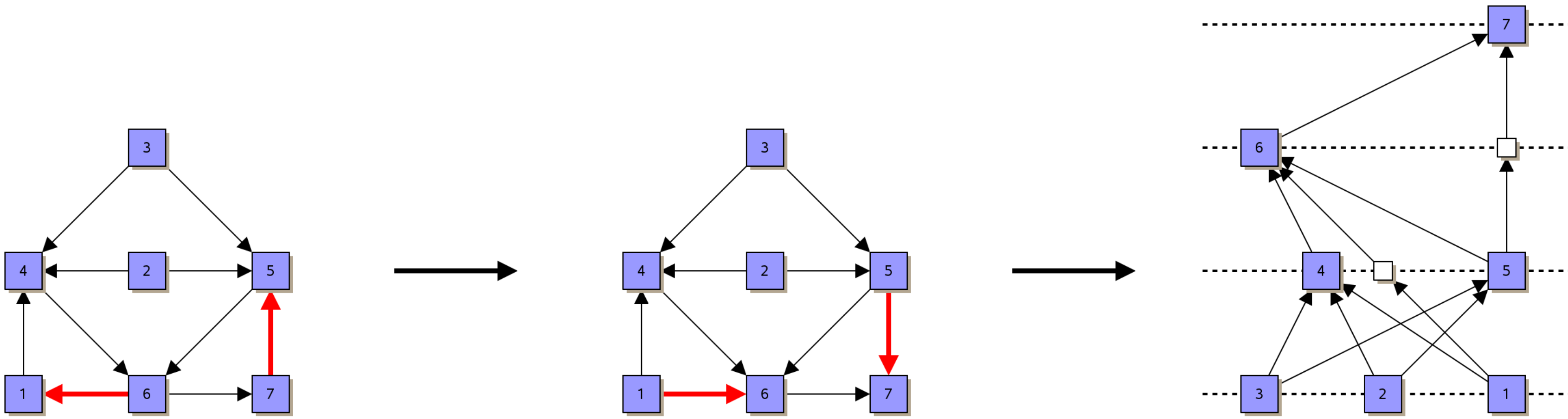


given

Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)

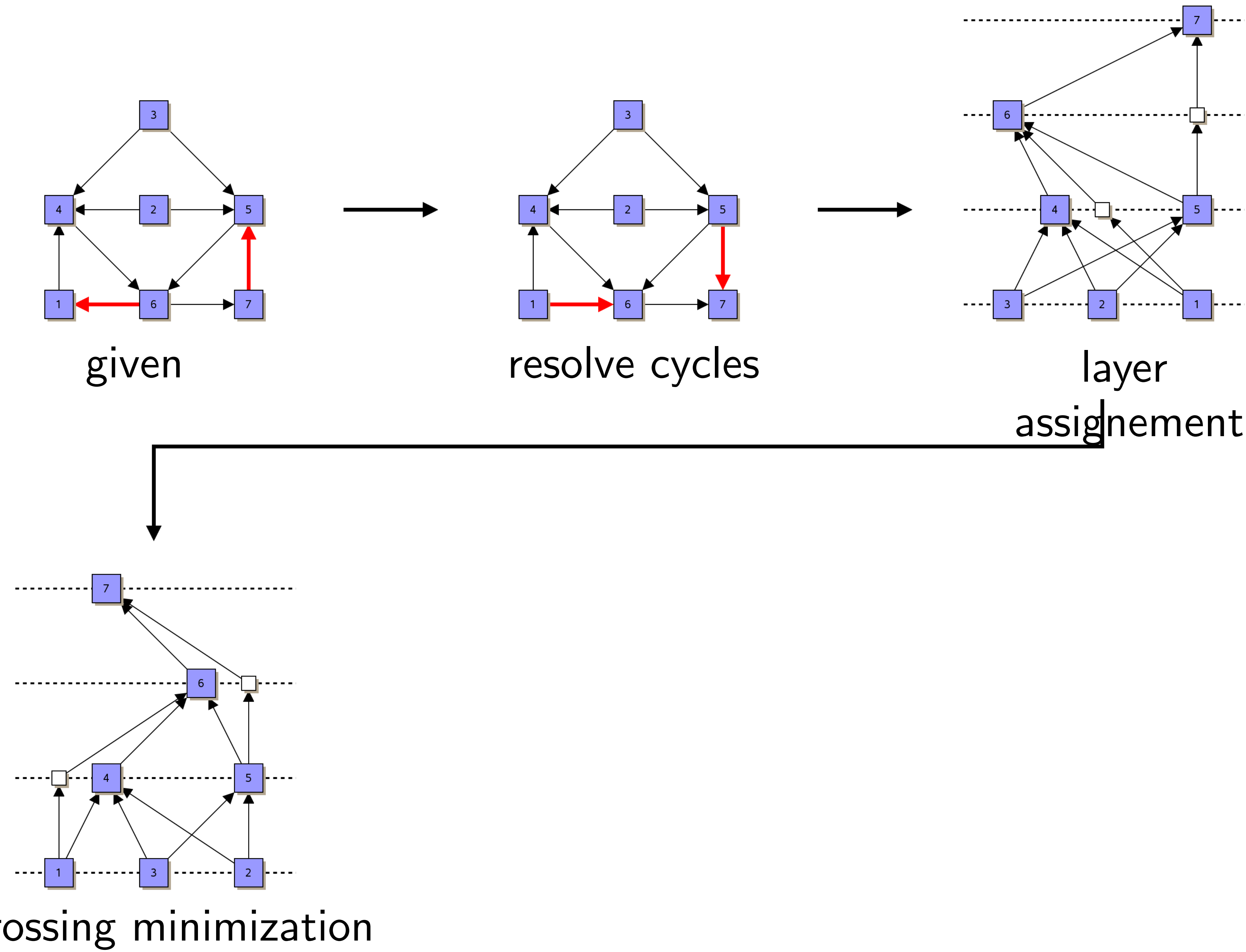


given

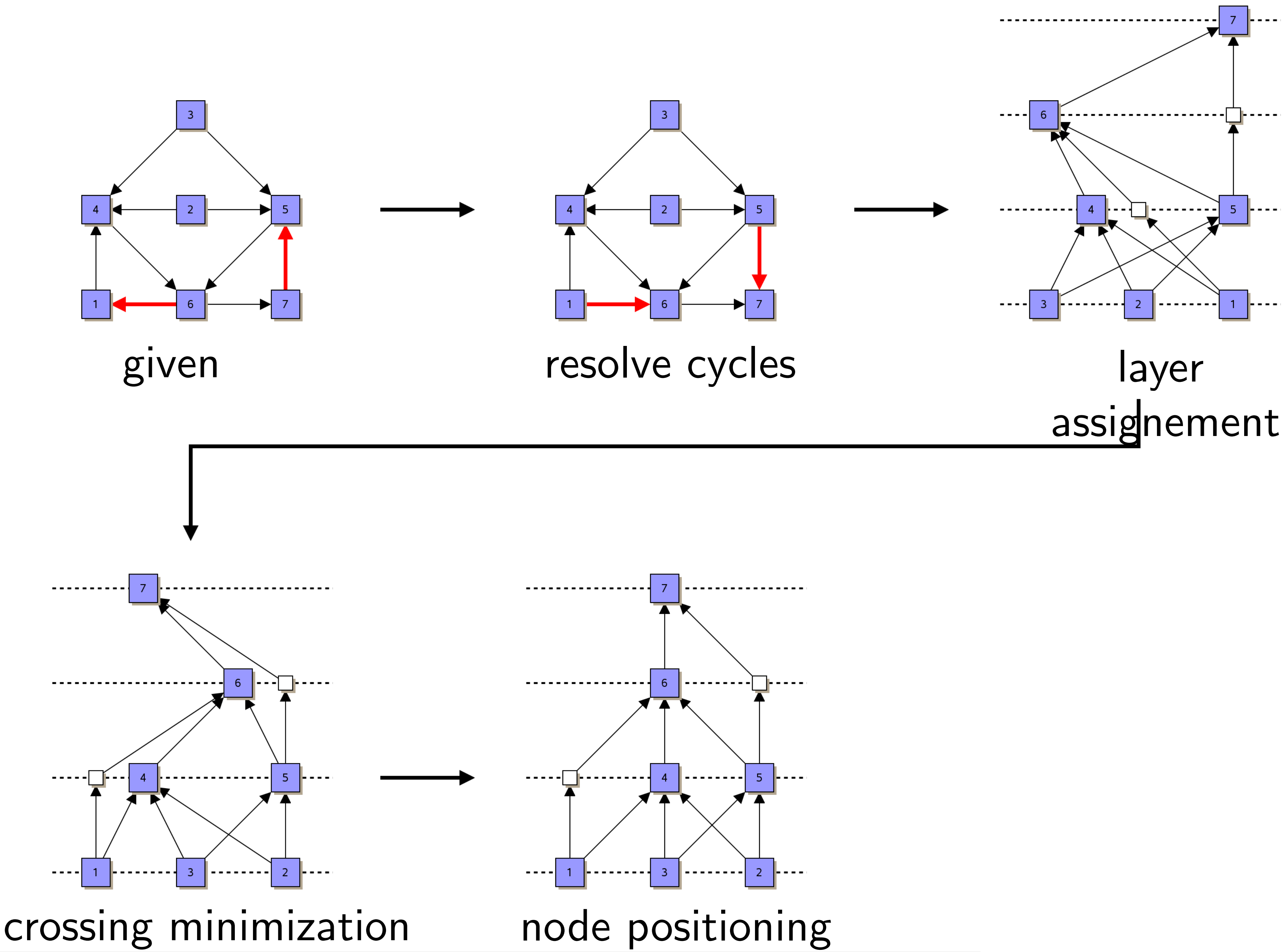
resolve cycles

layer assignement

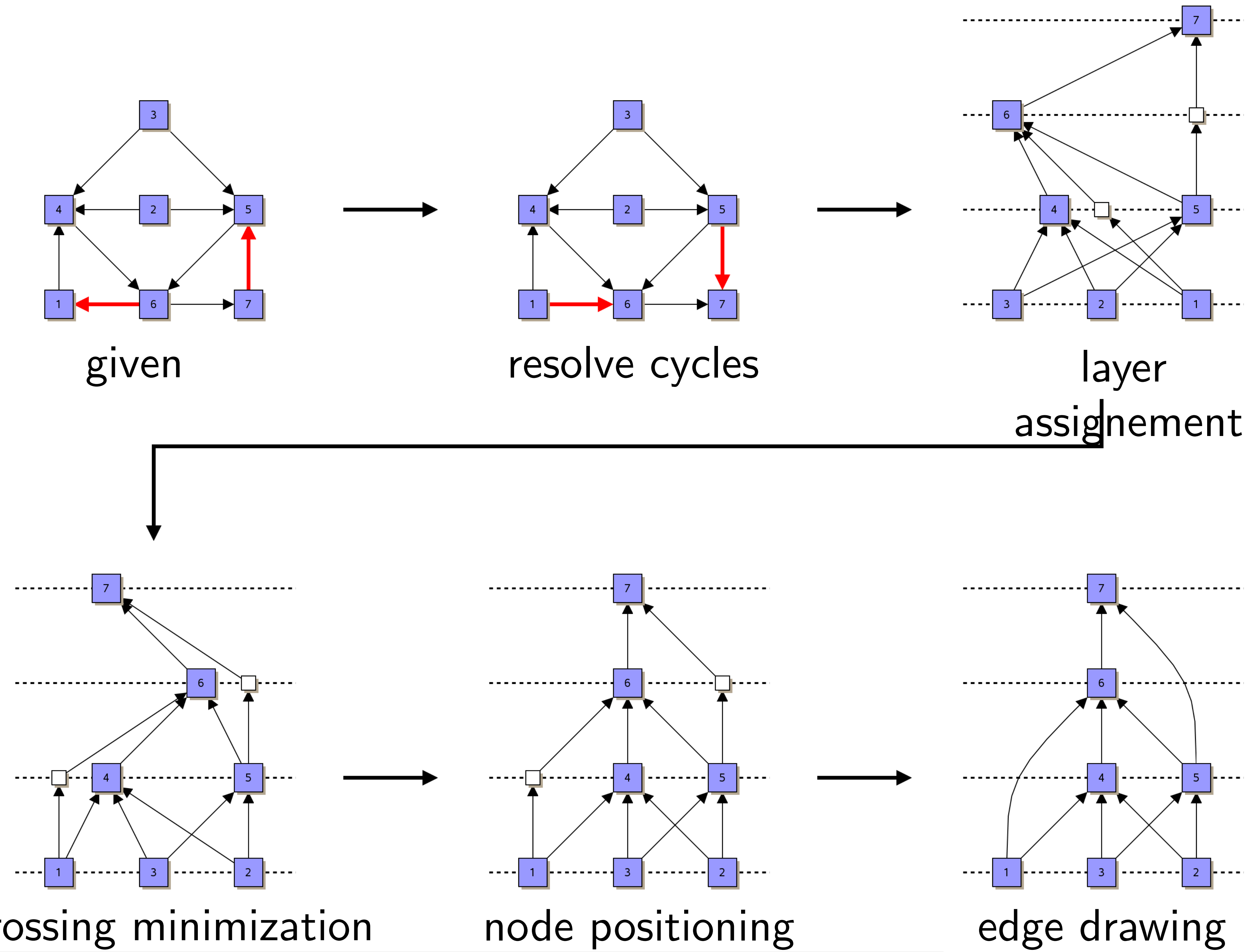
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



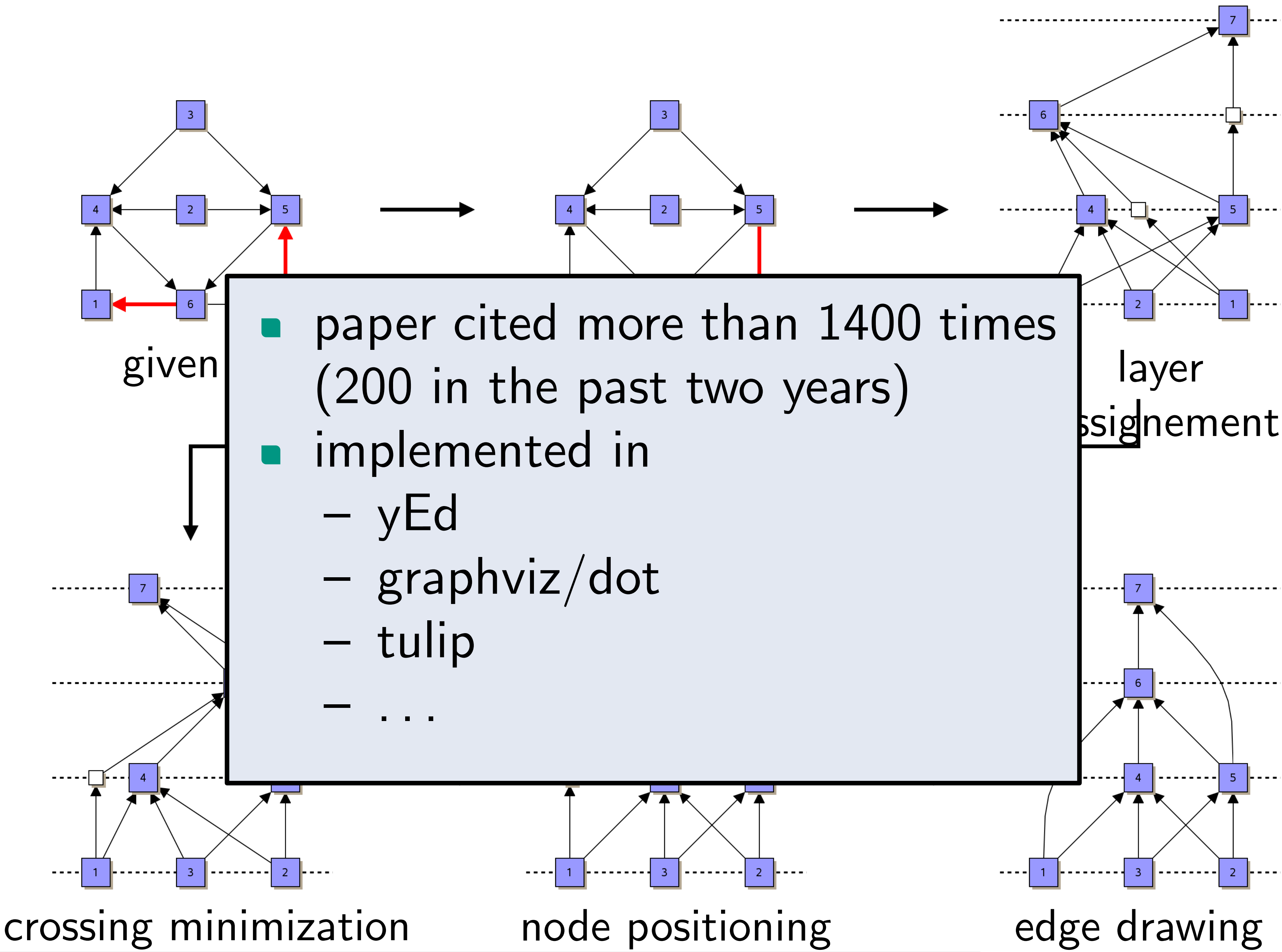
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)

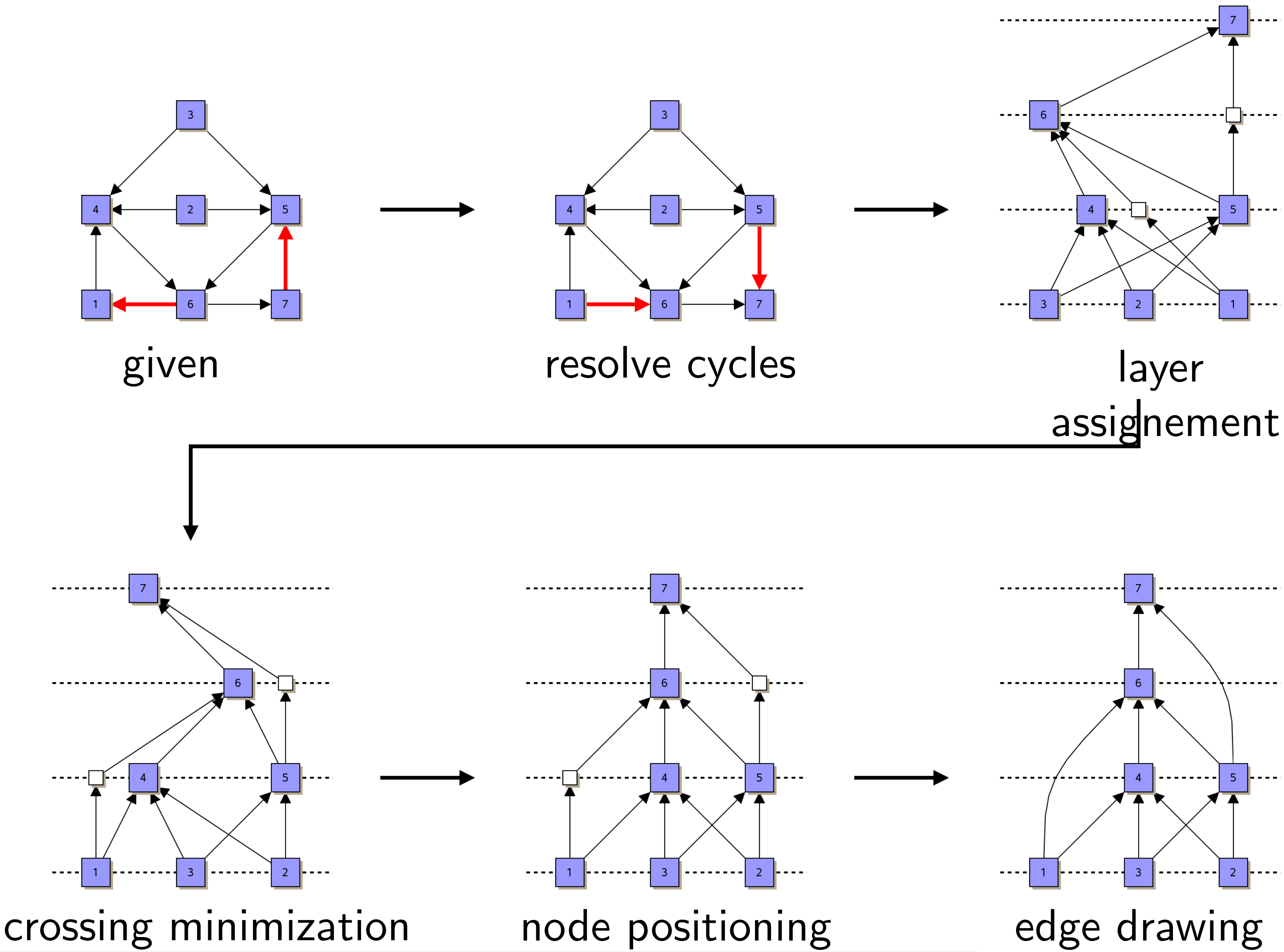


Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)

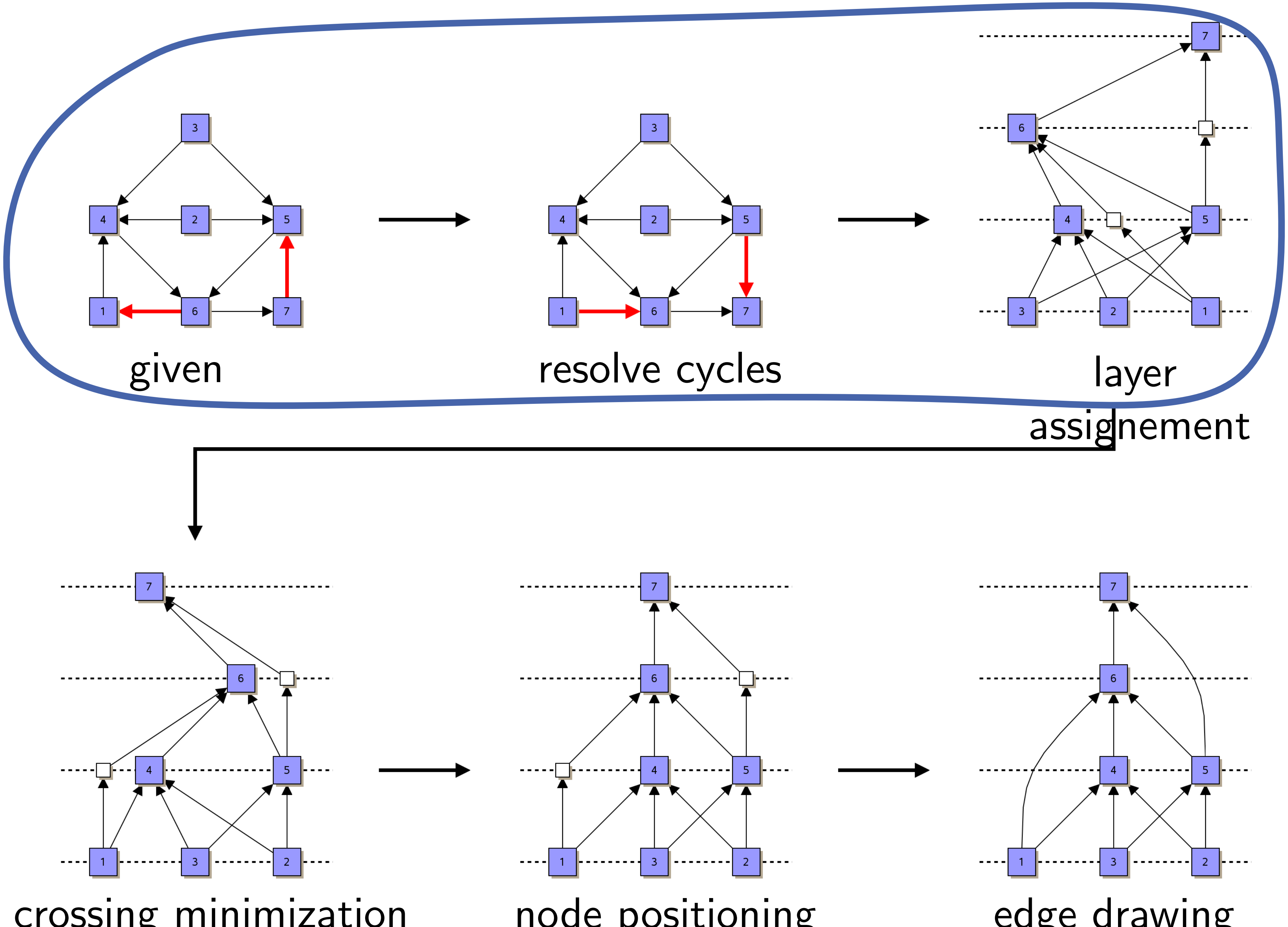


- paper cited more than 1400 times (200 in the past two years)
- implemented in
  - yEd
  - graphviz/dot
  - tulip
  - . . .

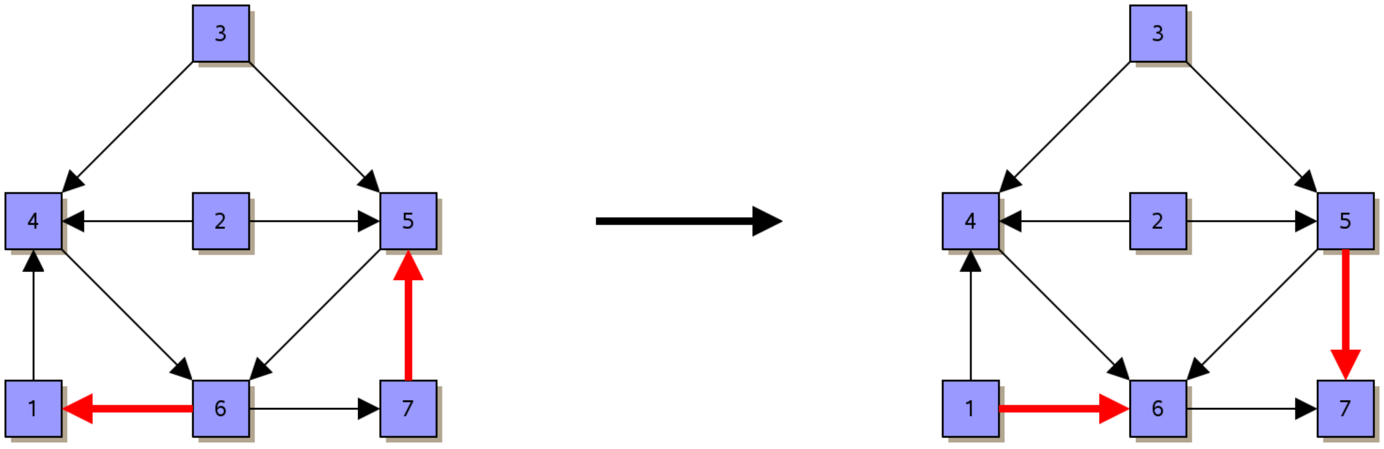
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



Step 1: Resolve Cycles



How would you proceed?

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')

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

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')

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

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

- Why D' does not contain cycles?
- Is $D'' = (V, A' \cup \text{rev}(A \setminus A'))$ acyclic?
- What is the running time?
- What one can say about $|A'|$?

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')

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

- $D' = (V, A')$ is a DAG
- $A \setminus A'$ is a feedback arc set
- Running time $O(|V| + |A|)$
- $|A'| \geq |A|/2$

Heuristic 1 (Berger, Shor 1990)

Lemma 1: Let $D = (V, A)$ be a connected, directed digraph. Heuristic 1 produces an acyclic digraph $D' = (V, A')$.

proof:

For the sake of contradiction assume there is a cycle C. Let u be the first visited vertex of C. Either incomming or outgoing edges of u are not in A', i.e. D' can not contain a cycle.

Heuristic 1 (Berger, Shor 1990)

Lemma 2: The digraph $D'' = (V, A' \cup \text{rev}(A \setminus A'))$, where A' is produced by Heuristic 1, is acyclic.

proof:

Heuristic 1 (Berger, Shor 1990)

Lemma 2: The digraph $D'' = (V, A' \cup \text{rev}(A \setminus A'))$, where A' is produced by Heuristic 1, is acyclic.

proof:

- For the sake of contr. assume there is a cycle C in D''.
- Let u be the first visited vertex of C. Cycle C contains a reversed edge incident to u, otherwise u can not have both incomming and outgoing edges in C.

Heuristic 1 (Berger, Shor 1990)

Lemma 2: The digraph $D'' = (V, A' \cup \text{rev}(A \setminus A'))$, where A' is produced by Heuristic 1, is acyclic.

proof:

- For the sake of contr. assume there is a cycle C in D''.
- Let u be the first visited vertex of C. Cycle C contains a reversed edge incident to u, otherwise u can not have both incomming and outgoing edges in C.
- W.l.o.g. assume (u, v) is the reversed edge. I.e. the original edge was (v, u), i.e. $(v, u) \in A \setminus A'$. Therefore, no other incomming edge to u is in A'. I.e. u has no incomming edges in C that are in A'.

Heuristic 1 (Berger, Shor 1990)

Lemma 2: The digraph $D'' = (V, A' \cup \text{rev}(A \setminus A'))$, where A' is produced by Heuristic 1, is acyclic.

proof:

- For the sake of contr. assume there is a cycle C in D''.
- Let u be the first visited vertex of C. Cycle C contains a reversed edge incident to u, otherwise u can not have both incomming and outgoing edges in C.
- W.l.o.g. assume (u, v) is the reversed edge. I.e. the original edge was (v, u), i.e. $(v, u) \in A \setminus A'$. Therefore, no other incomming edge to u is in A'. I.e. u has no incomming edges in C that are in A'.
- Therefore the incomming edge to u in C is also a reversed edge. I.e. both incomming and outgoing edges of u in C are in $A \setminus A'$, which is impossible, as u is the first vertex visited by the algorithm in C.

Heuristic 2 (Eades, Lin, Smyth 1993)

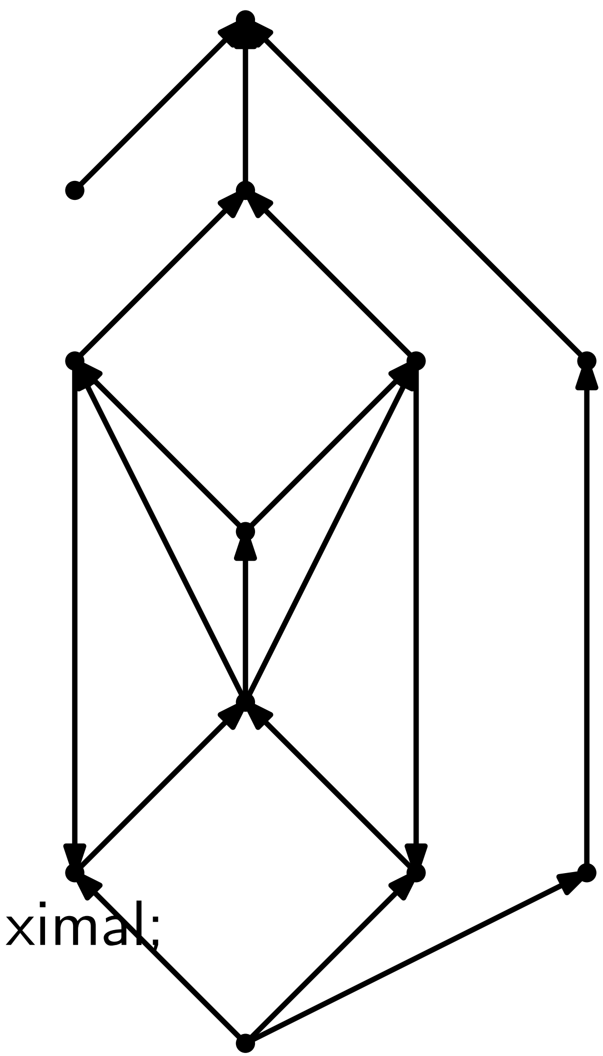
$A' := \emptyset$;

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

  while in V exists 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' := ∅;
while V ≠ ∅ do
    while in V exists a sink v do
        A' ← A' ∪ N←(v)
        remove v and N←(v): {V, n, m}_sink
```

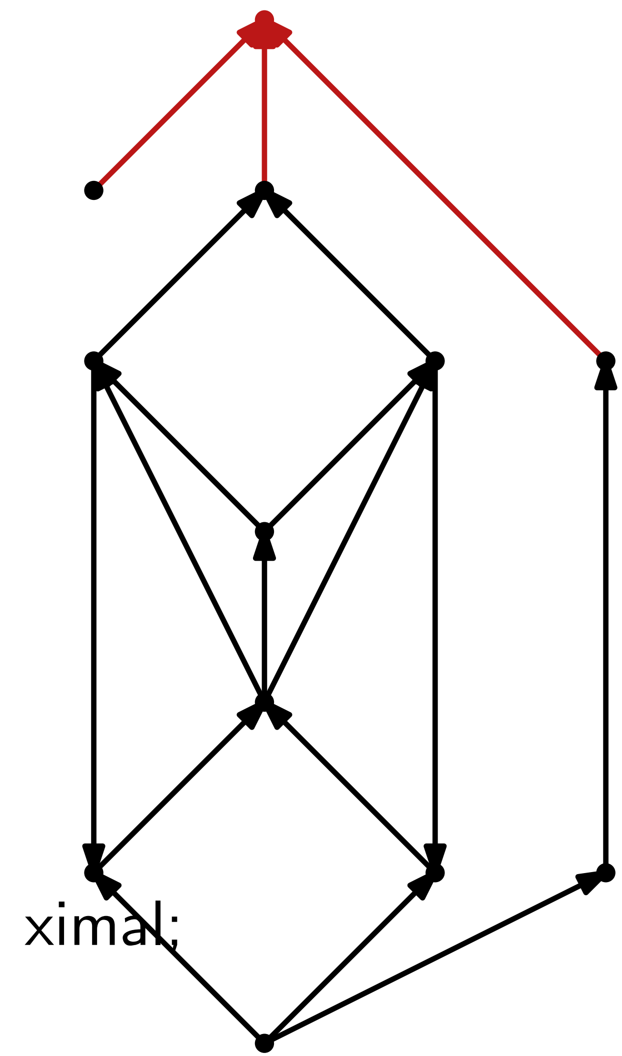
$A' := \emptyset$;

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

  while in V exists 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)

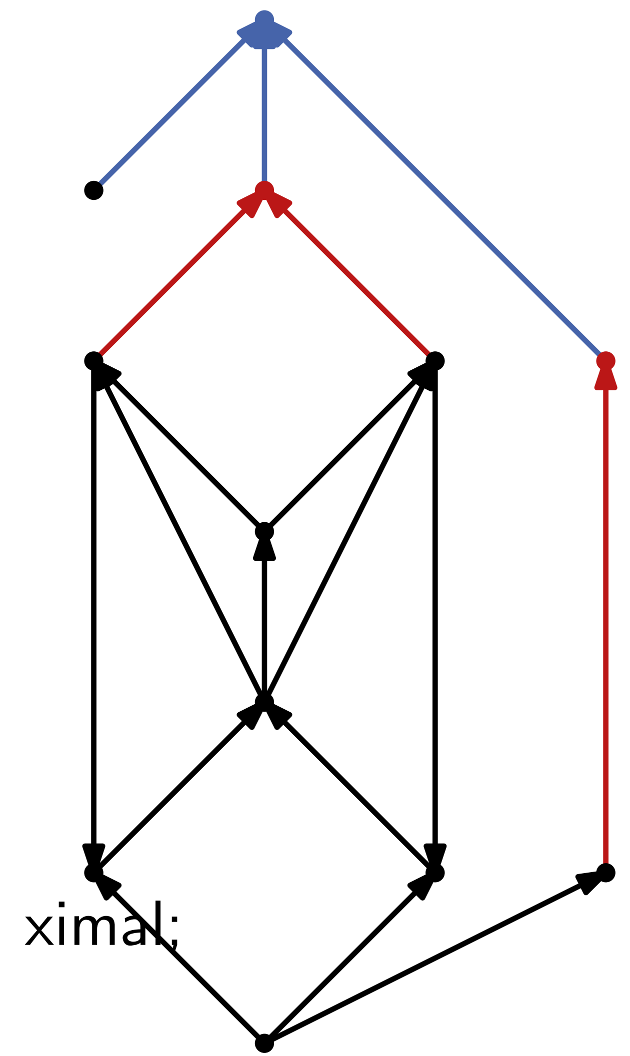
$A' := \emptyset$;

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

  while in V exists 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)

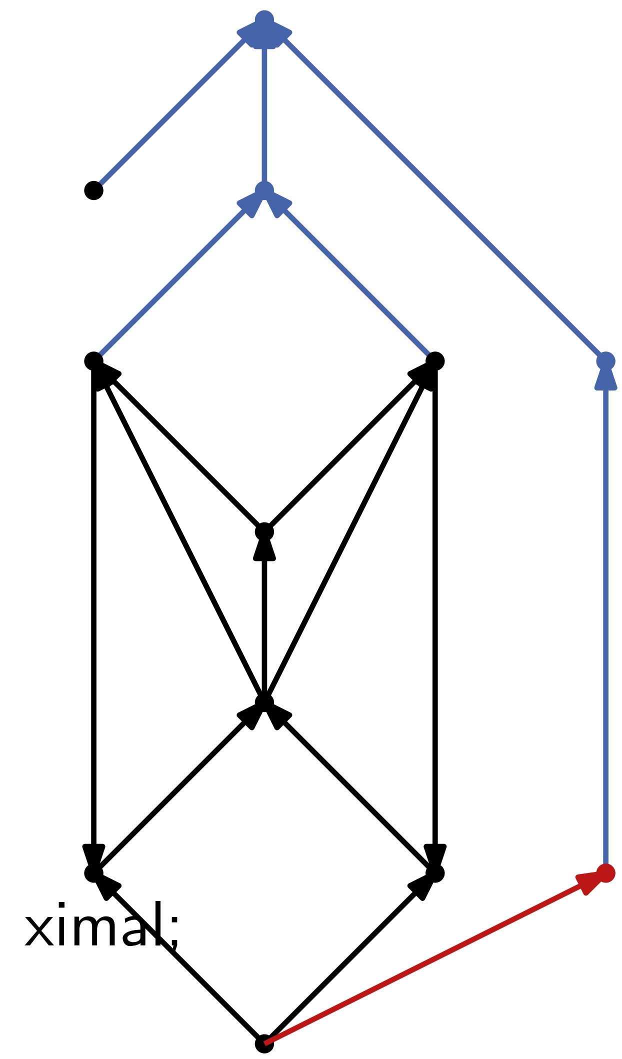
$A' := \emptyset$;

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

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

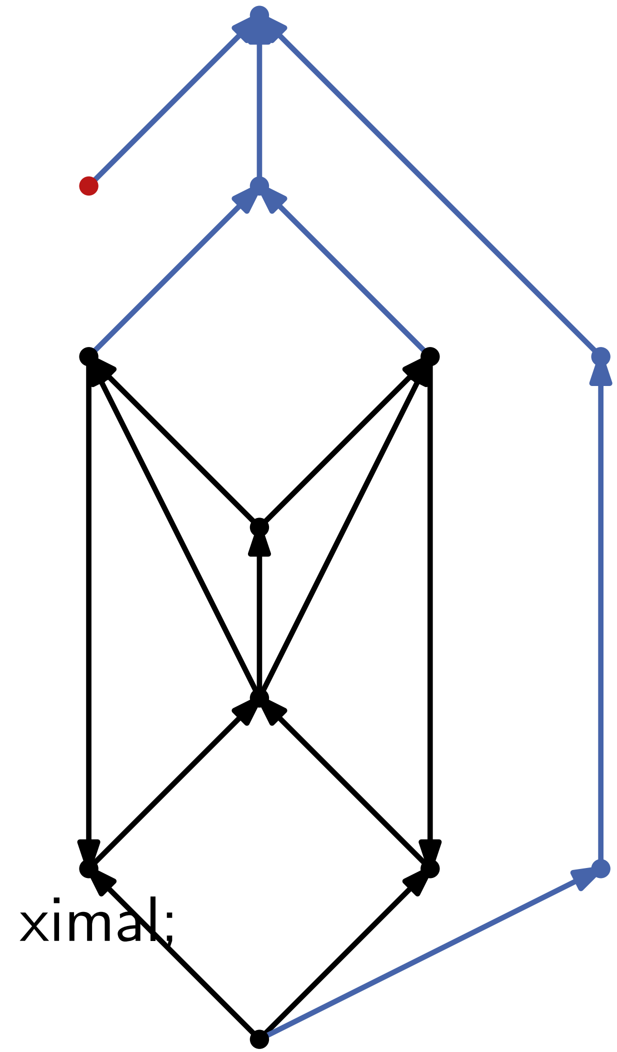
    $A' \leftarrow A' \cup N^{\leftarrow}(v)$

    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 in V exists 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 node from V: $\{V, n, m\}_{\text{iso}}$



Heuristic 2 (Eades, Lin, Smyth 1993)

$A' := \emptyset$;

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

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

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

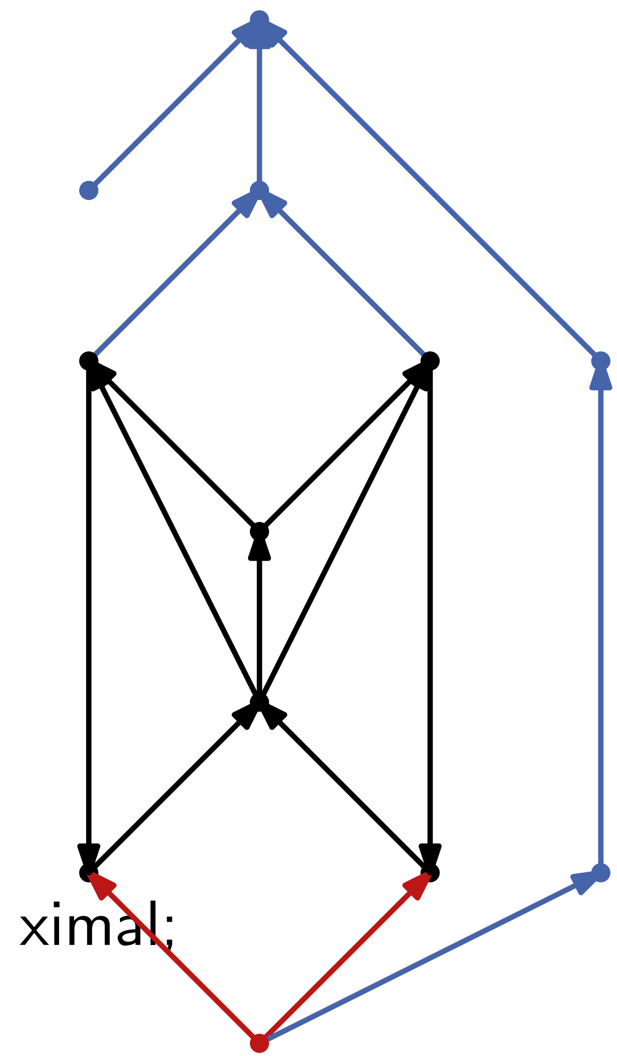
    remove v and $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$

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

  while in V exists 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 in V exists 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 node from V: $\{V, n, m\}_{\text{iso}}$

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

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

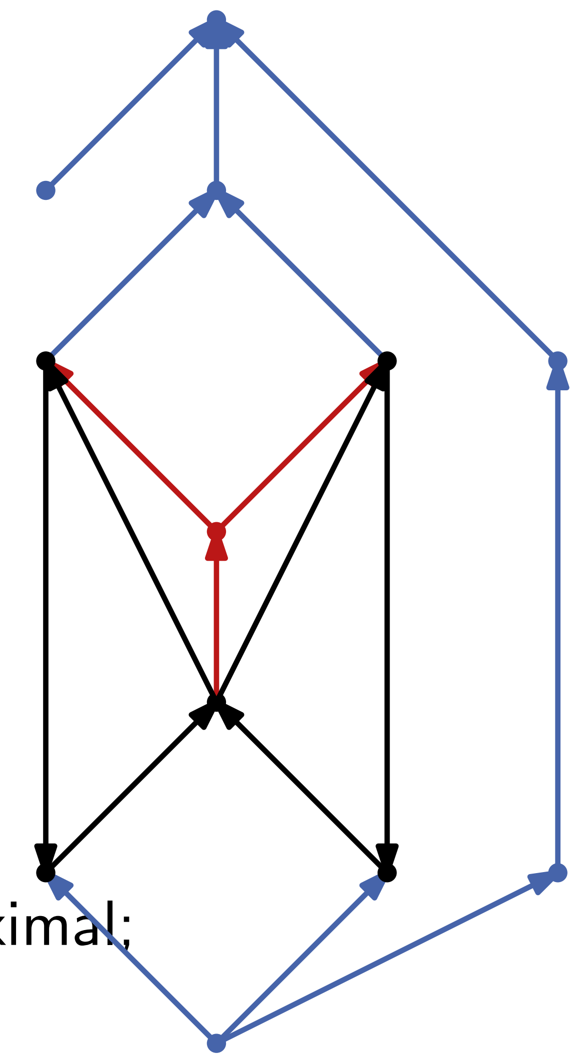
    remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

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

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

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



Heuristic 2 (Eades, Lin, Smyth 1993)

$A' := \emptyset$;

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

  while in V exists 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 node from V: $\{V, n, m\}_{\text{iso}}$

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

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

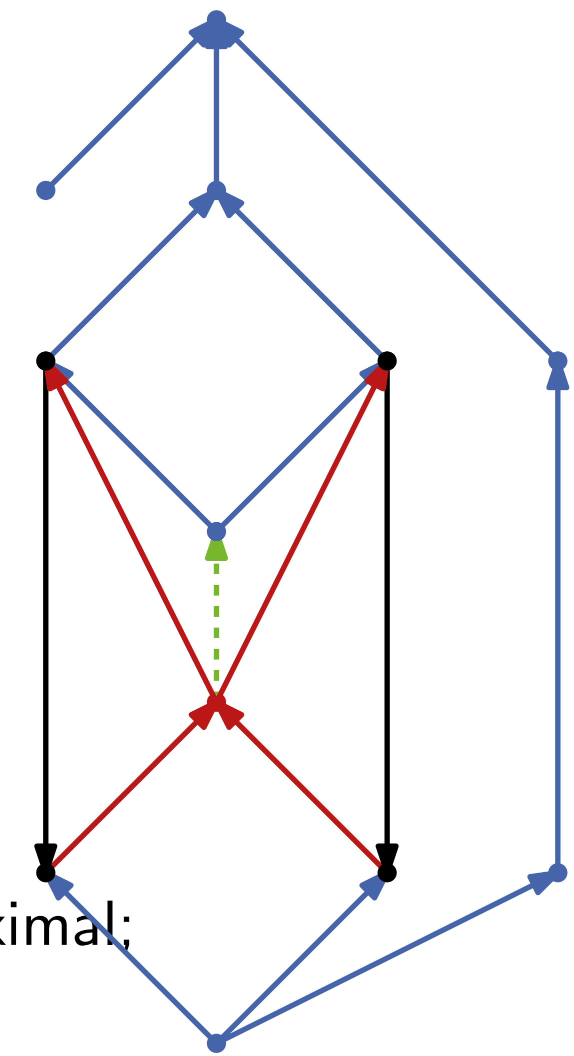
    remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

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

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

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



Heuristic 2 (Eades, Lin, Smyth 1993)

$A' := \emptyset$;

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

  while in V exists 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 node from V: $\{V, n, m\}_{\text{iso}}$

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

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

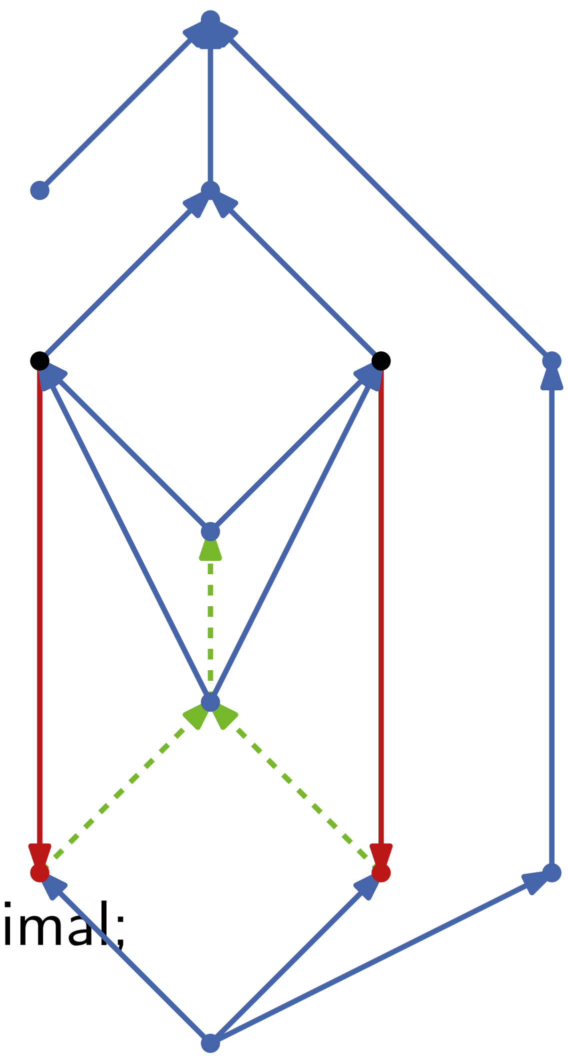
    remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

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

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

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



Heuristic 2 (Eades, Lin, Smyth 1993)

$A' := \emptyset$;

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

  while in V exists 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 node from V: $\{V, n, m\}_{\text{iso}}$

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

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

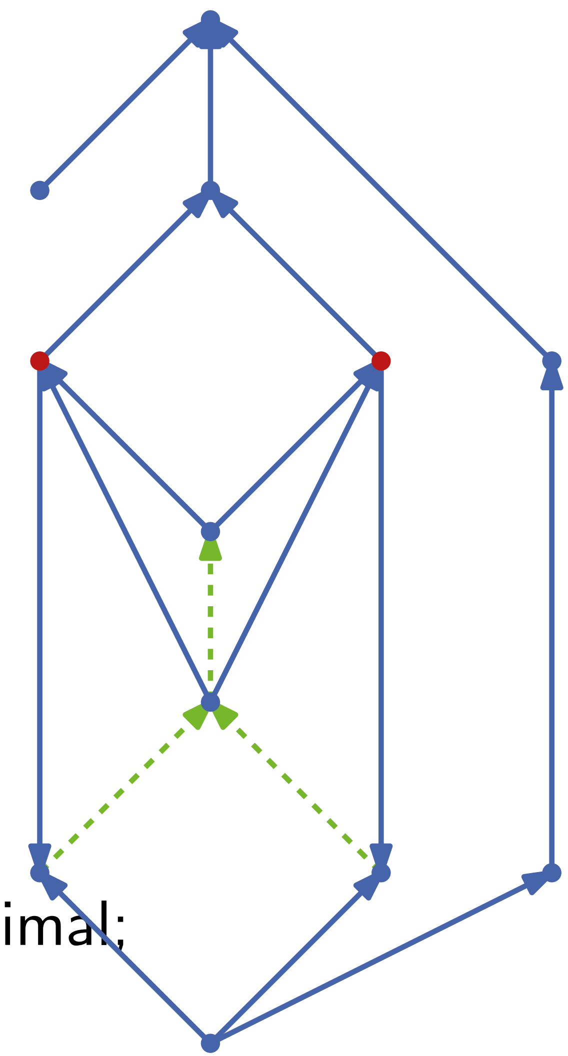
    remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

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

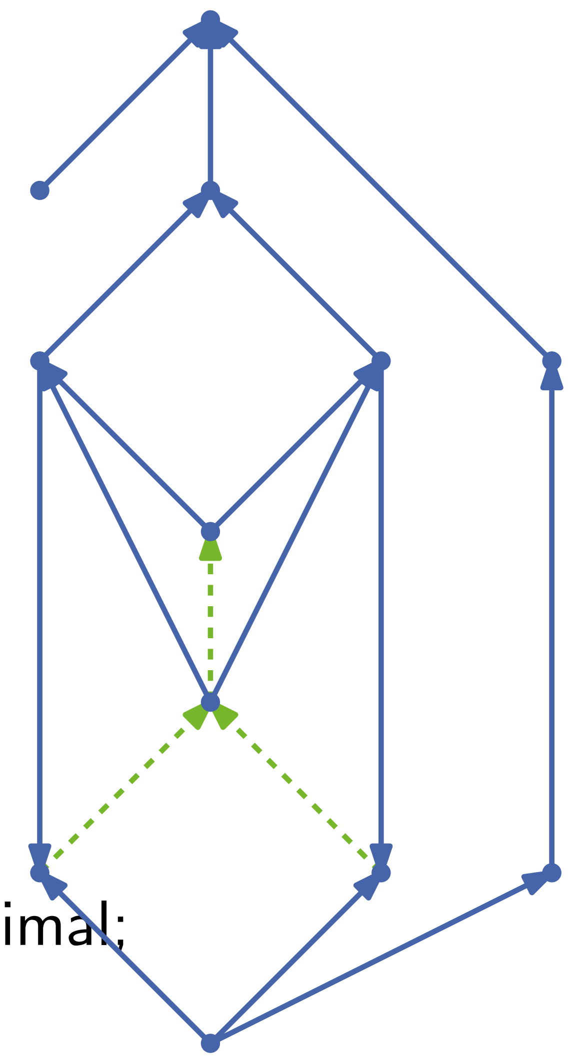
    $A' \leftarrow A' \cup N^{\rightarrow}(v)$

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



Heuristic 2 (Eades, Lin, Smyth 1993)

$A' := \emptyset$;
while $V \neq \emptyset$ **do**
    while in V exists 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 node from V: $\{V, n, m\}_{\text{iso}}$
    while in V exists 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 $v \in V$ such that $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$ maximal;
        $A' \leftarrow A' \cup N^{\rightarrow}(v)$
        remove v and $N(v)$: $\{V, n, m\}_{\{=,<\}}$



Heuristic 2 (Eades, Lin, Smyth 1993)

$A' := \emptyset$;

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

    while in V exists 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 node from V: $\{V, n, m\}_{\text{iso}}$

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

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

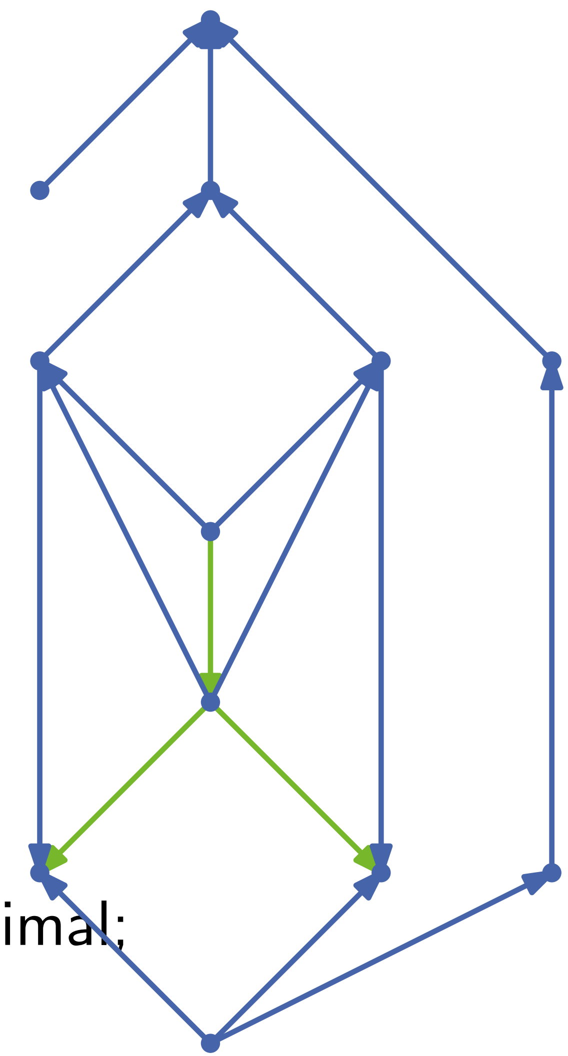
        remove v and $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

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

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

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



Heuristic 2 – Analysis

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

Further methods:

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

Heuristic 2 – Analysis

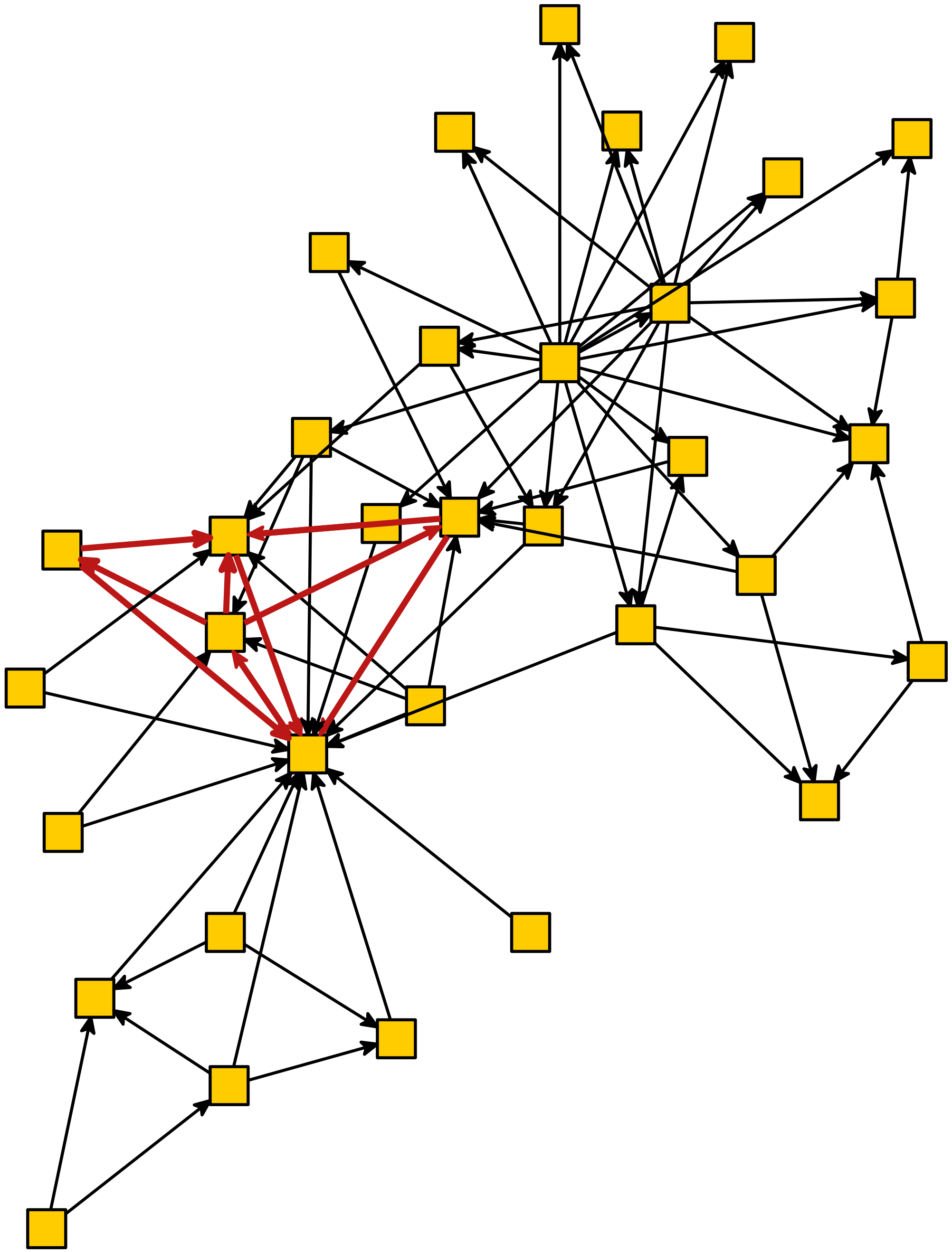
Theorem 1: Let $D = (V, A)$ be a connected, direted graph without 2-cycles. Heuristic 2 computes a set of edges A' with $|A'| \geq |A|/2 + |V|/6$.
The running time is $O(|A|)$.

Further methods:

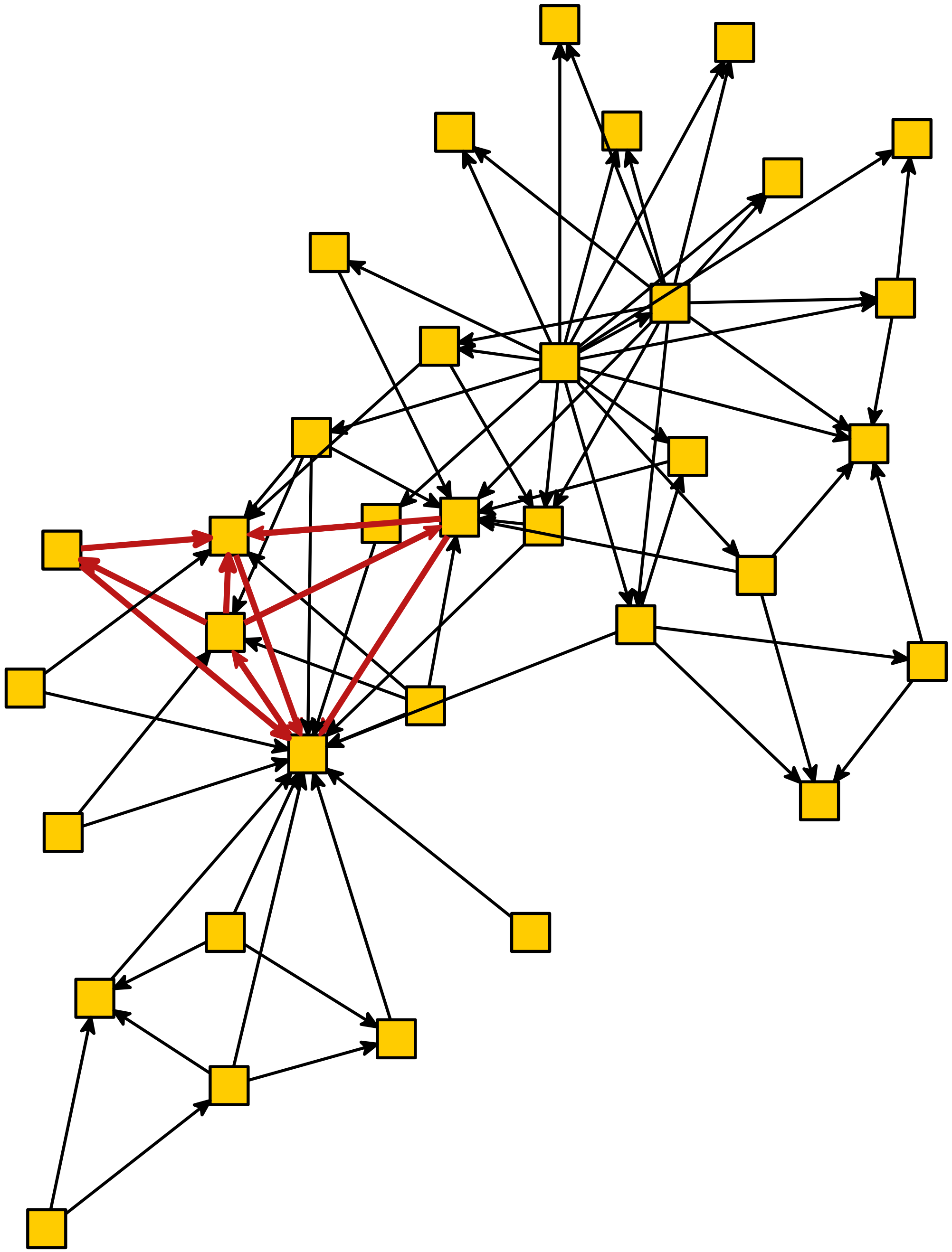
- $|A'| \geq |A| \left(1/2 + \Omega\left(\frac{1}{\sqrt{\deg_{\max}(D)}}\right)\right)$ (Berger, Shor 1990)
- exact solution with integer linear programming, using branch-and-cut technique (Grötschel et al. 1985)

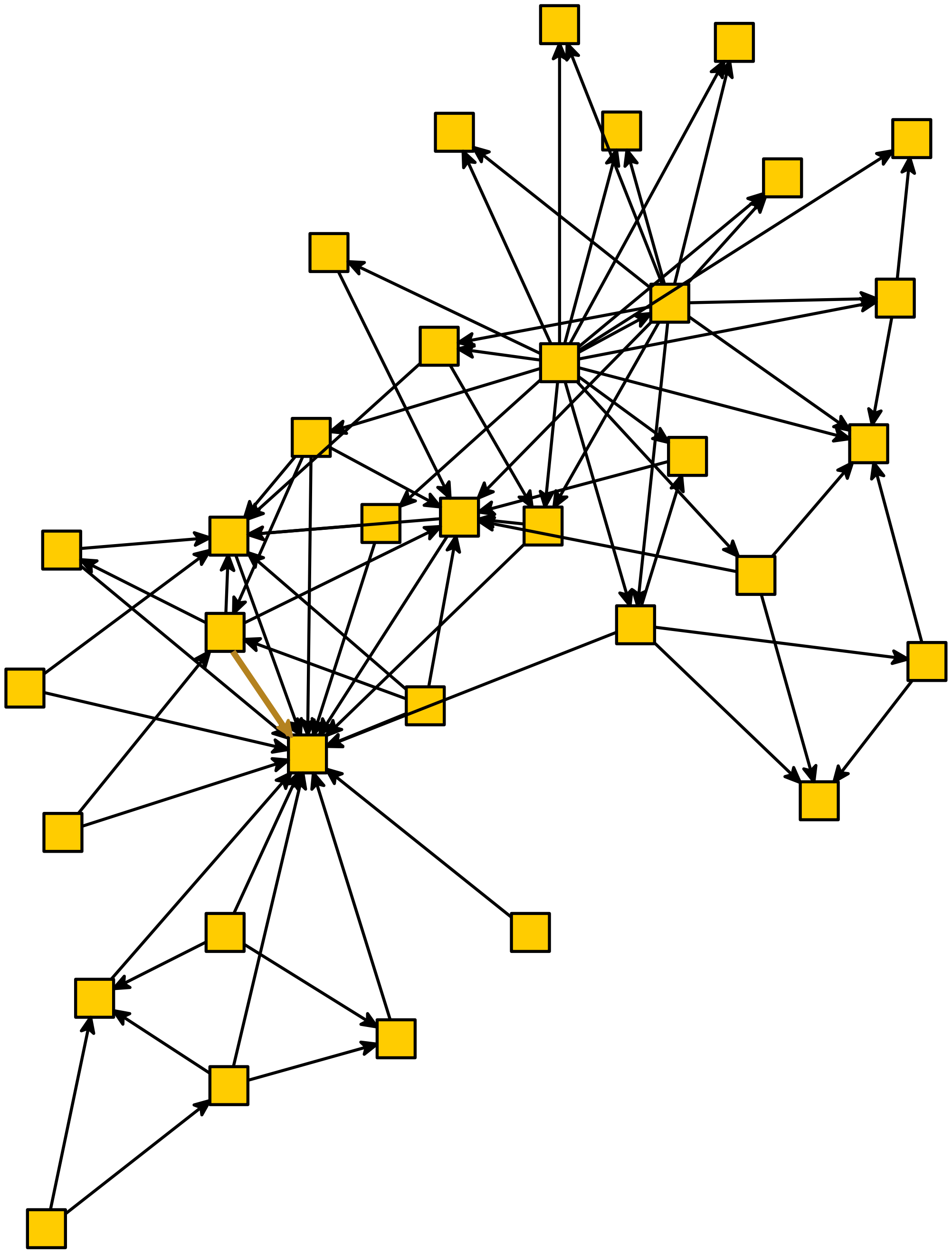
For $|A| \in O(|V|)$ Heuristic 2 performs similarly.

Example

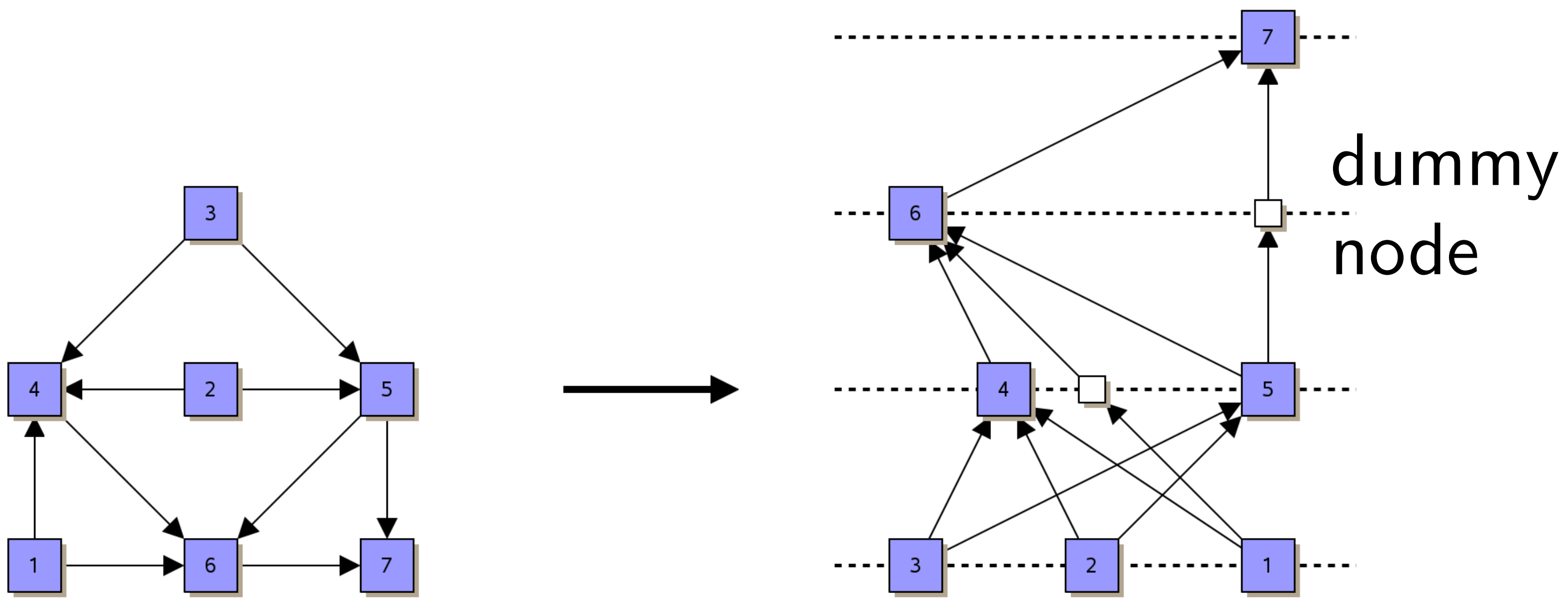


Example





Step 2: Layer Assignement



How would you proceed?

Step 2: Layer Assignement

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

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

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**) $L_1, \dots, L_h$ 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

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

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 neighbourse 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 neighbourse lie below v
- the resulting height h is minimized

Algorithm

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

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

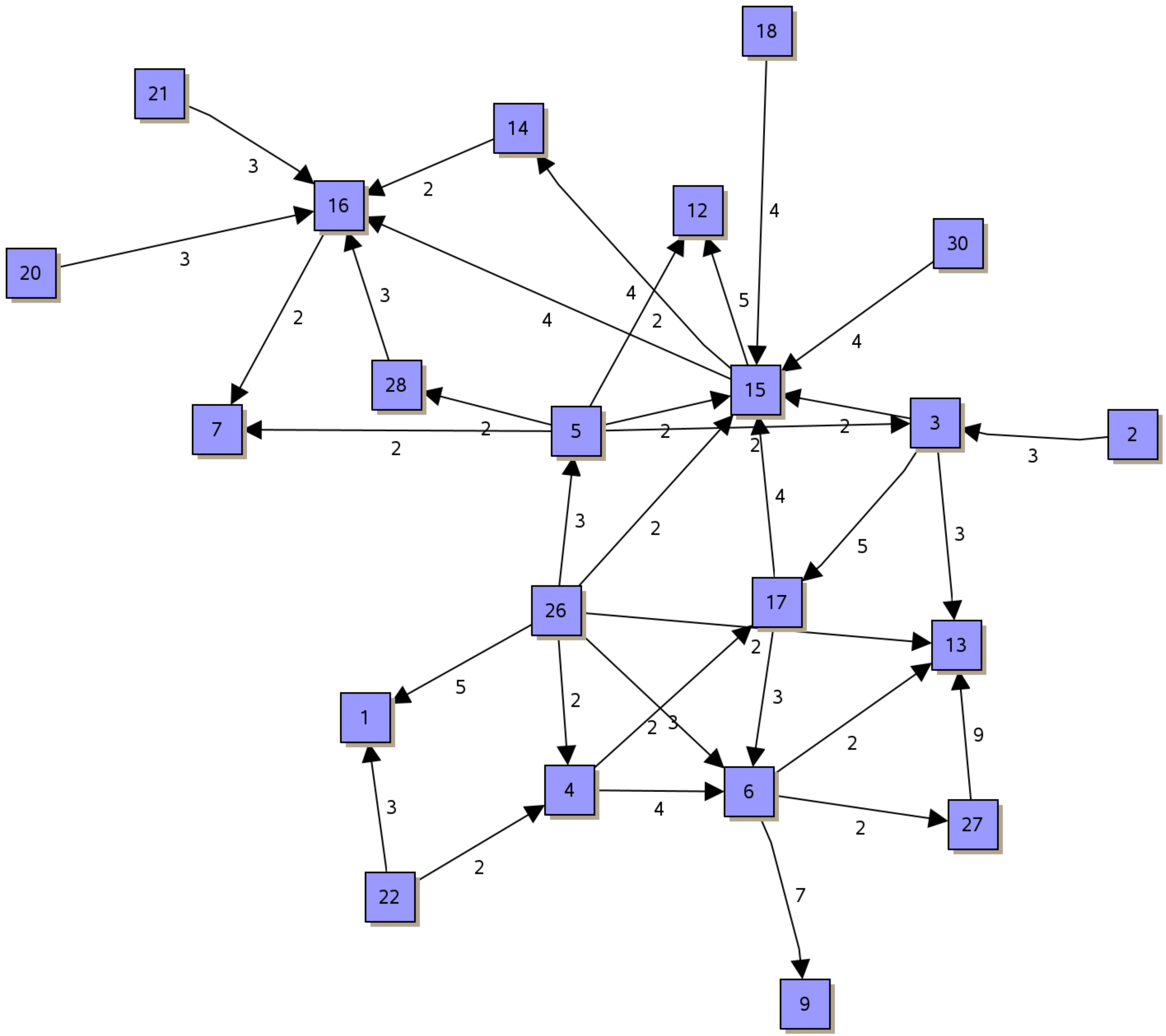
- all incomming neighbourse lie below v
- the resulting height h is minimized

Algorithm

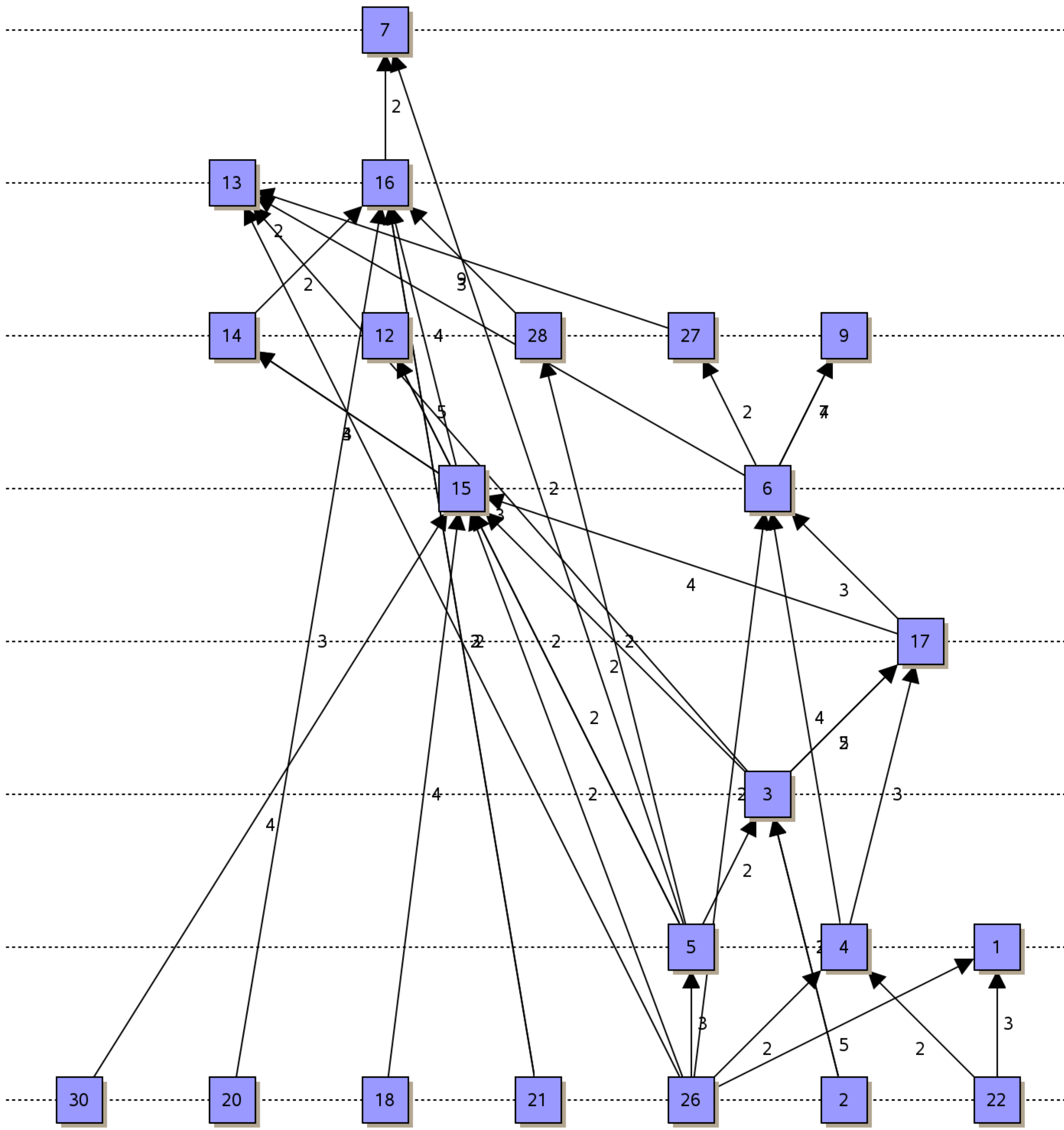
- $L_1 \leftarrow$ the set of sources in D
- set $y(u) \leftarrow \max_{v \in N^{\leftarrow}(u)}\{y(v)\} + 1$

How can we implement the algorithm in $O(|V| + |A|)$ time?

Example



Example



Total Edge Length

Can be formulated as an integer linear program:

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

Total Edge Length

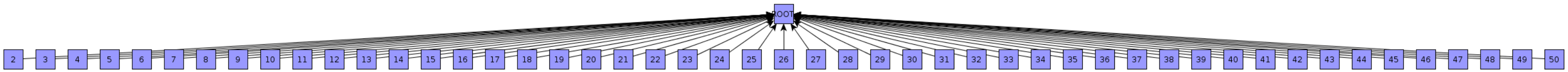
Can be formulated as an integer linear program:

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

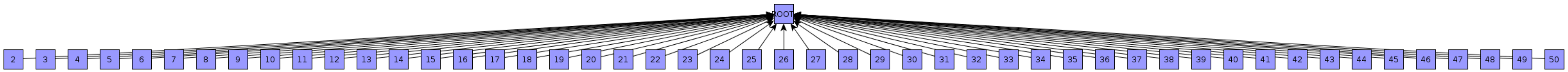
One can show that:

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

Width of the Layout



Width of the Layout



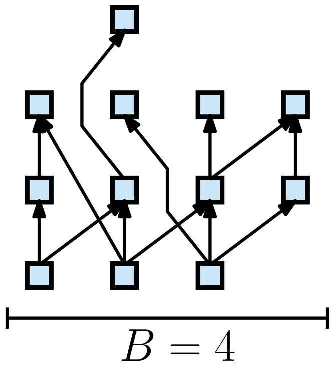
$\rightarrow$ bound the width!

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

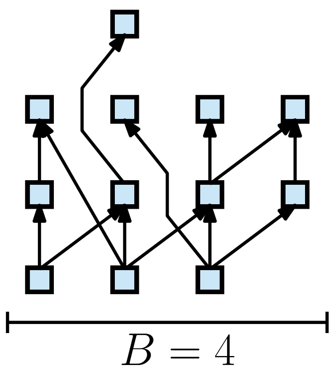


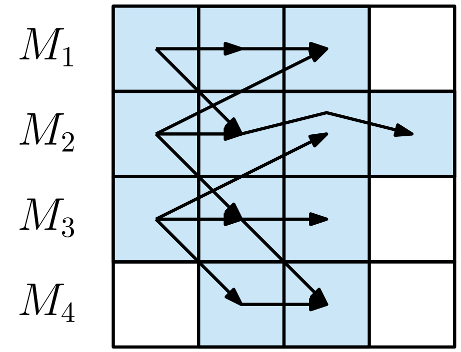
Layer Assignement with Fixed Width

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

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 $T = 3$.

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 $< 4/3$.

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 $< 4/3$.

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

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 $T = 3$.

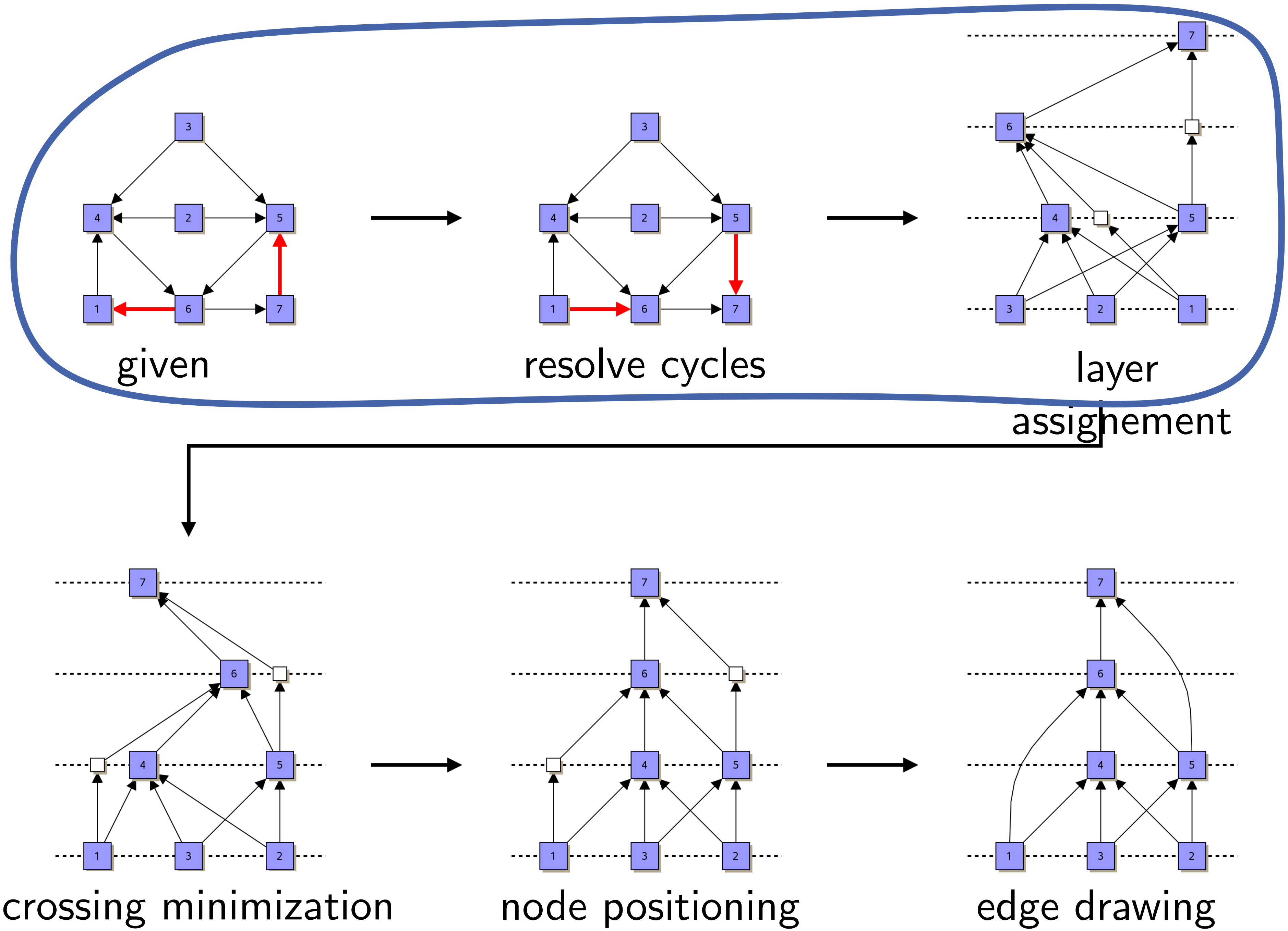
Corollary: If $\mathcal{P} \neq \mathcal{NP}$, there is no polynomial algorithm for MPCS with approximation factor $< 4/3$.

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


given
resolve cycles
layer assignement
crossing minimization
node positioning
edge drawing