Karlsruhe Institute of Technology

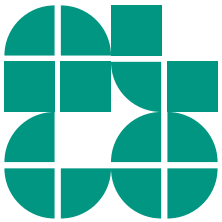
Algorithmen zur Visualisierung von Graphen

Lagenlayouts

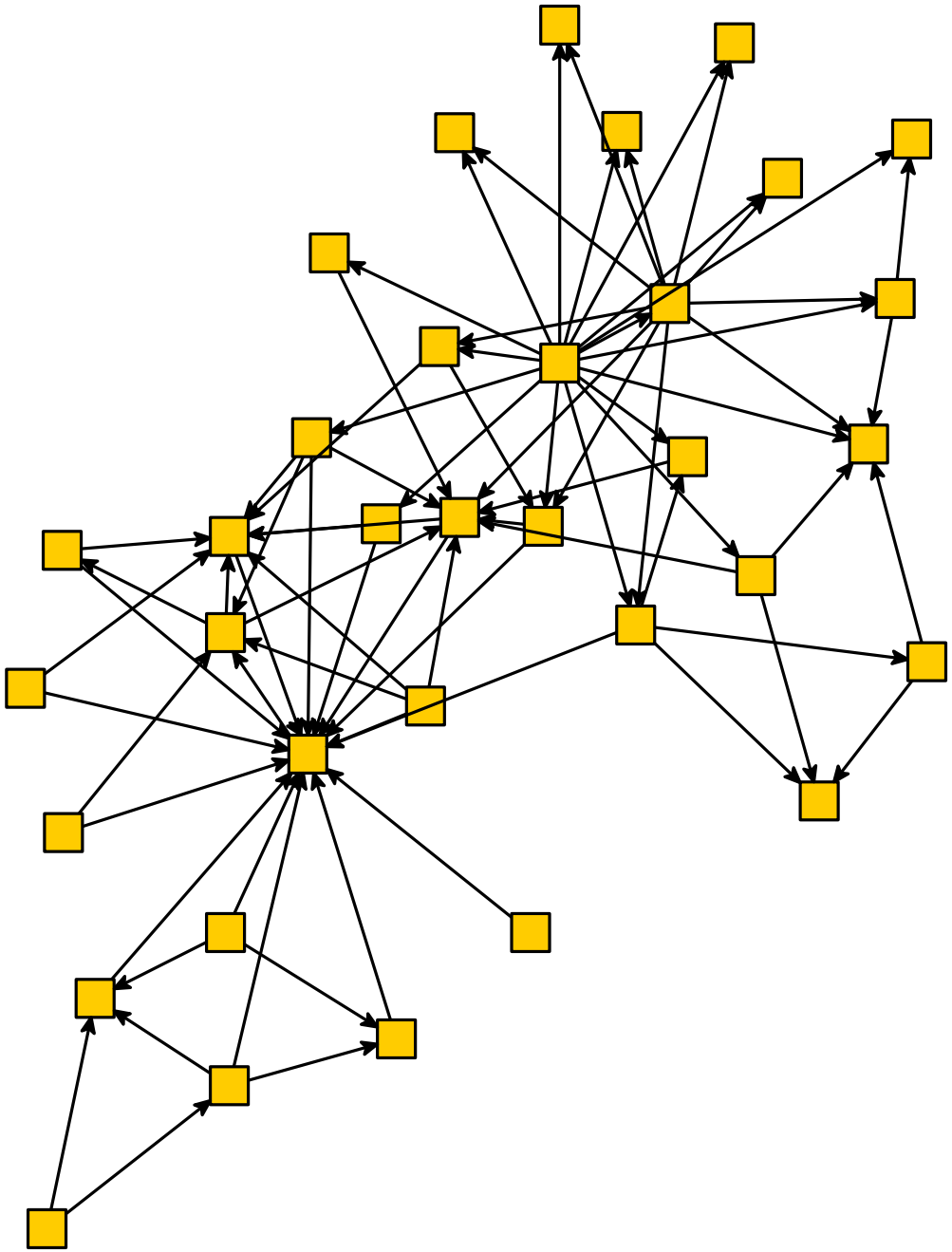
INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Tamara Mchedlidze · **Martin Nöllenburg**

10.12.2014



Ein Beispielgraph...

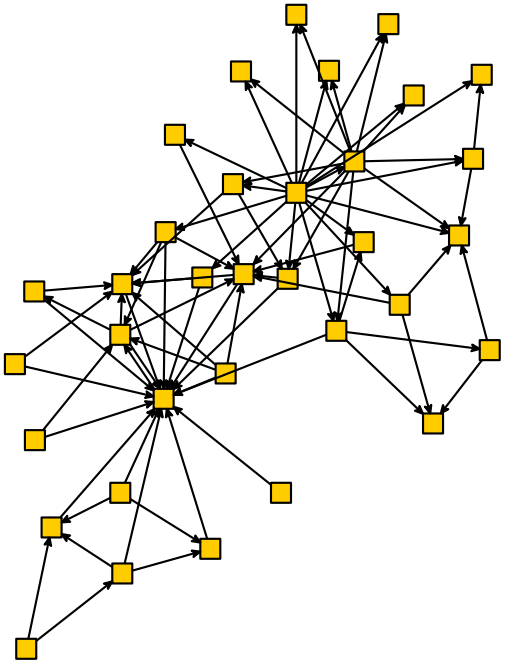


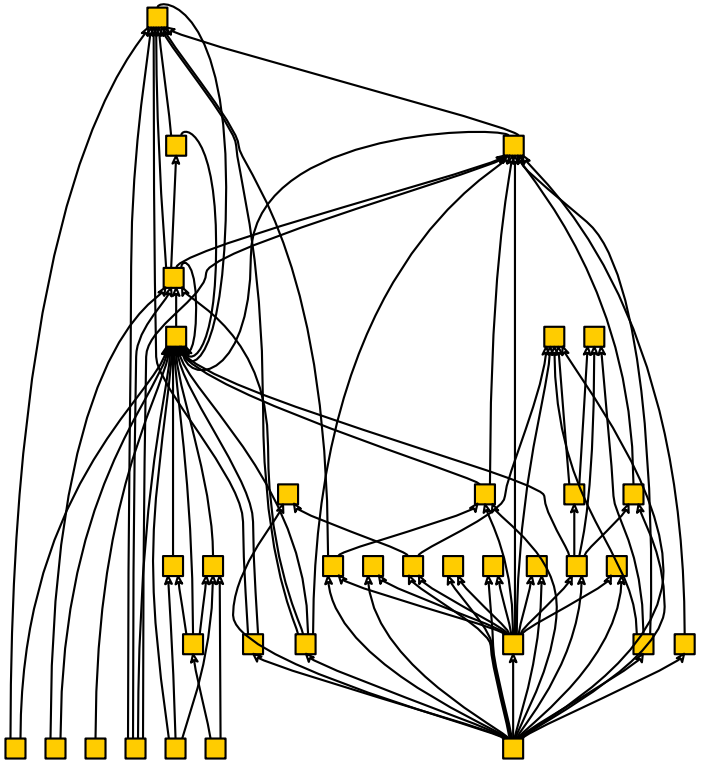
- Welche Eigenschaften haben wir?
- Welche Ästhetikkriterien sind sinnvoll?

Lagenlayouts

Geg.: gerichteter Graph $D = (V, A)$

Ges.: Zeichnung von D, die die Hierarchie verdeutlicht





Lagenlayouts

Geg.: gerichteter Graph $D = (V, A)$

Ges.: Zeichnung von D, die die Hierarchie verdeutlicht

Kriterien:

- möglichst viele Kanten aufwärtsgerichtet
- Kanten möglichst geradlinig und kurz
- Zuordnung der Knoten auf (wenige) horizontale Ebenen
- möglichst wenige Kantenkreuzungen
- Knoten gleichmäßig verteilt

Lagenlayouts

Geg.: gerichteter Graph $D = (V, A)$

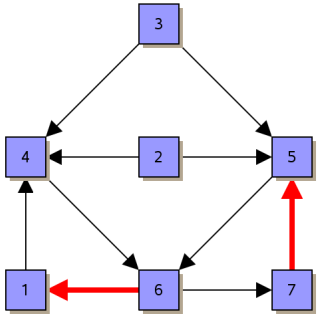
Ges.: Zeichnung von D, die die Hierarchie verdeutlicht

Kriterien:

- möglichst viele Kanten aufwärtsgerichtet
- Kanten möglichst geradlinig und kurz
- Zuordnung der Knoten auf (wenige) horizontale Ebenen
- möglichst wenige Kantenkreuzungen
- Knoten gleichmäßig verteilt

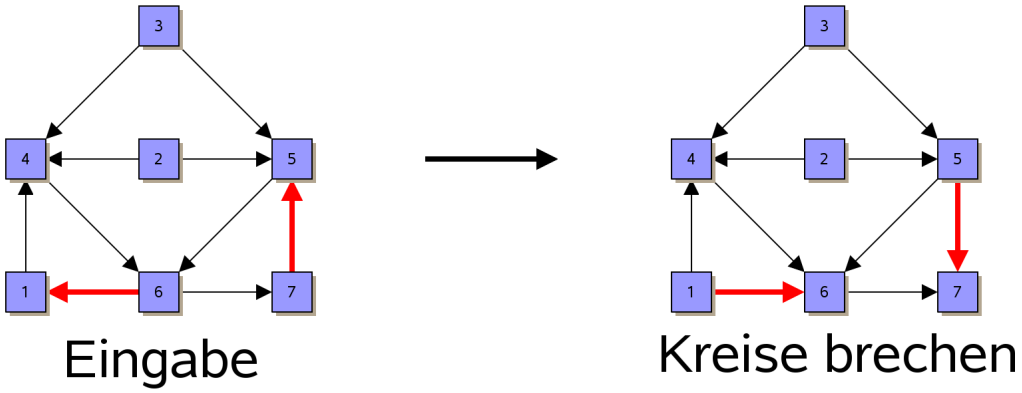
Optimierungskriterien widersprechen sich zum Teil

Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)

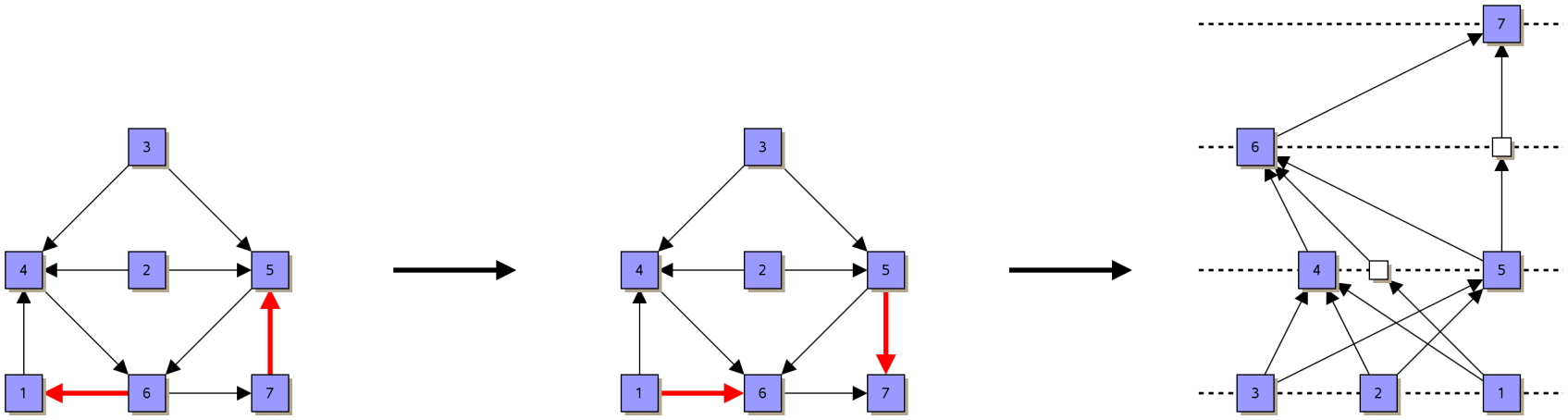


Eingabe

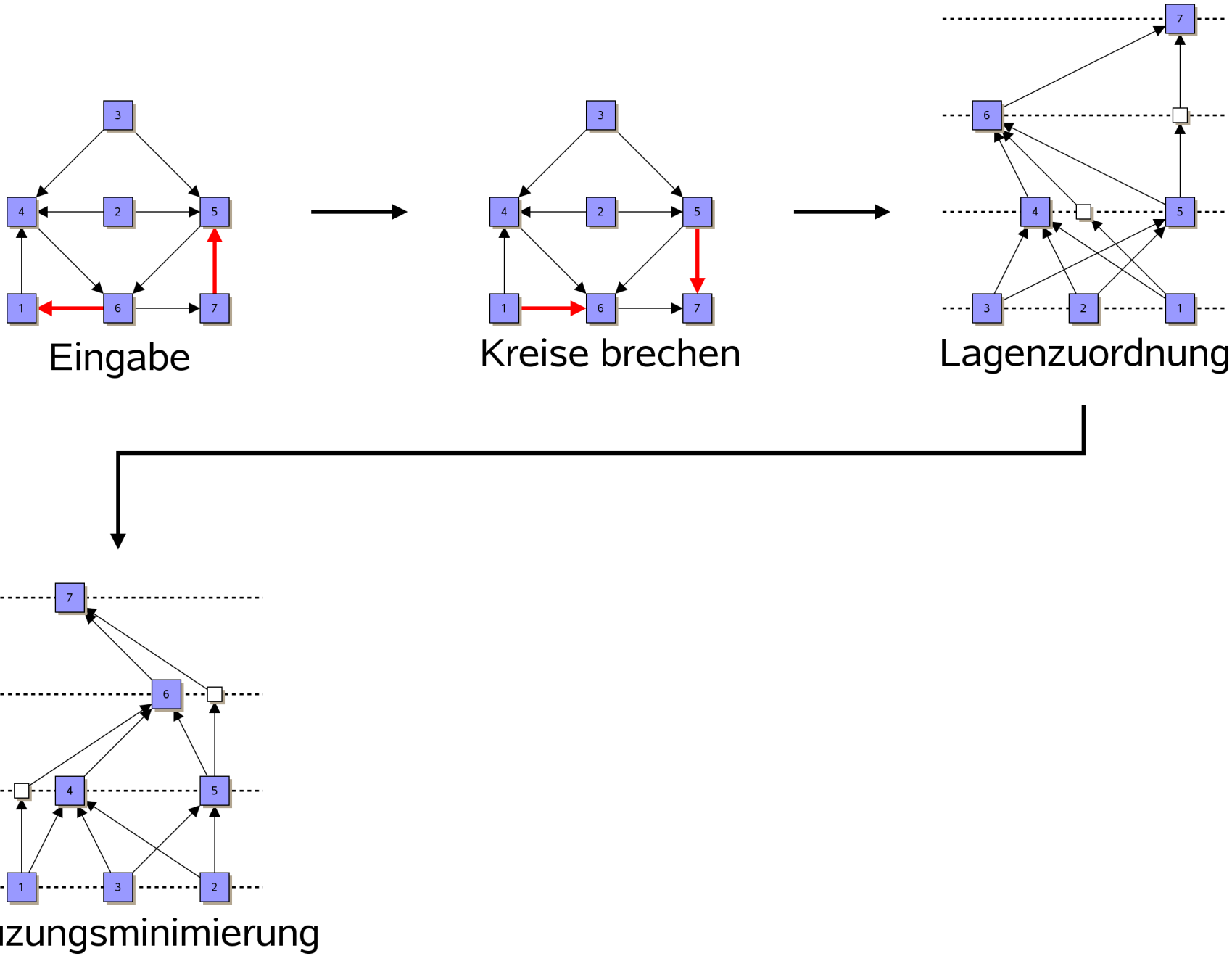
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



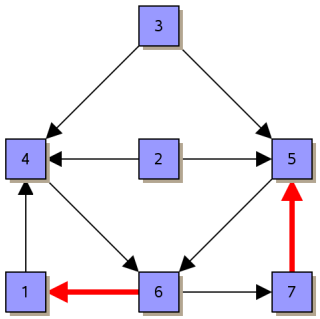
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



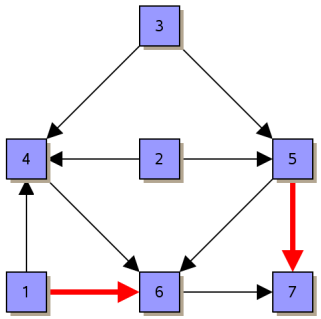
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



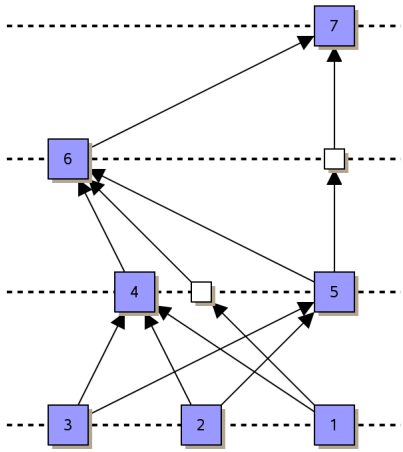
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



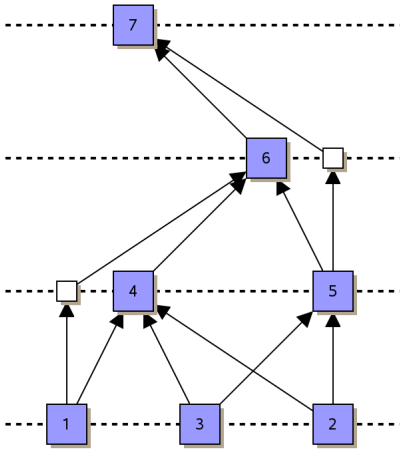
Eingabe



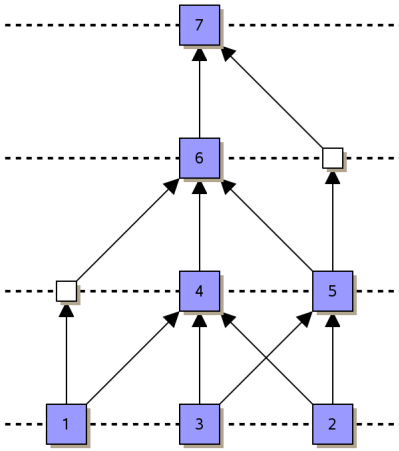
Kreise brechen



Lagenzuordnung

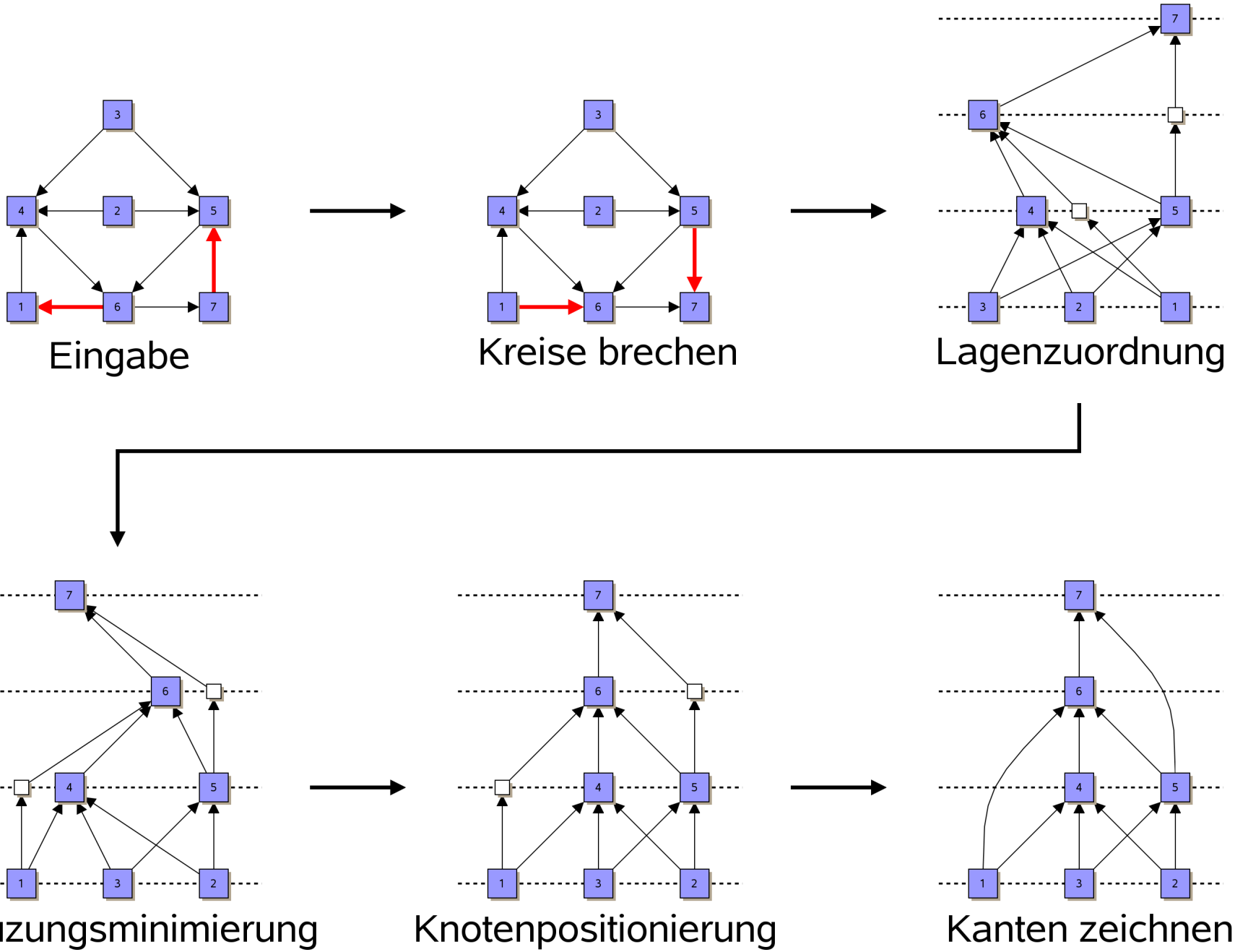


Kreuzungsminimierung

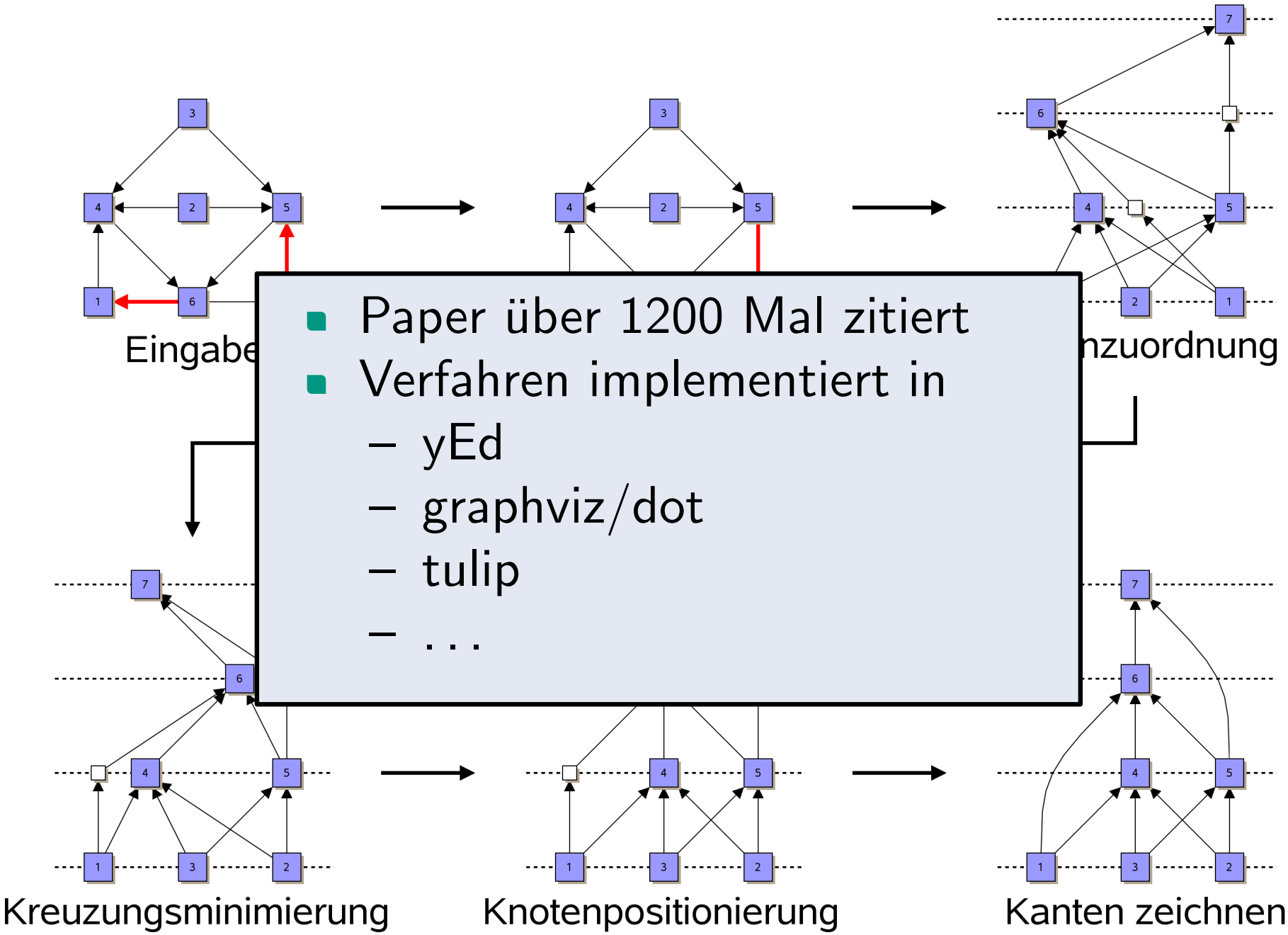


Knotenpositionierung

Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)

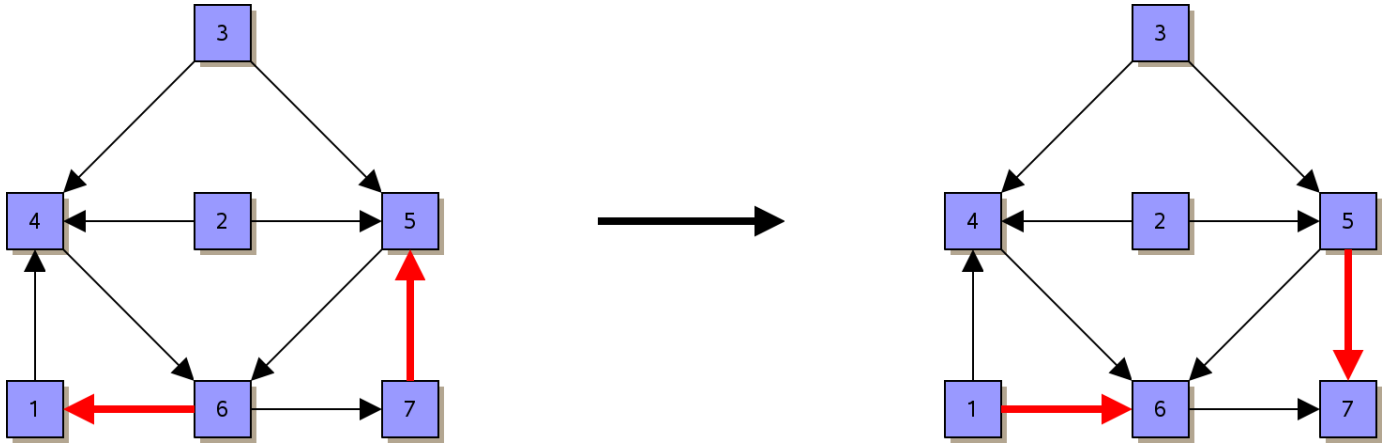


Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



- Paper über 1200 Mal zitiert
- Verfahren implementiert in
  - yEd
  - graphviz/dot
  - tulip
  - ...

Schritt 1: Kreise entfernen



Wie würden Sie vorgehen?

Feedback Arc Set

Idee:

- finde maximalen azyklischen Teilgraphen
- invertiere Richtung der übrigen Kanten

Feedback Arc Set

Idee:
- finde maximalen azyklischen Teilgraphen
- invertiere Richtung der übrigen Kanten

Maximum Acyclic Subgraph

geg: gerichteter Graph $D = (V, A)$

ges: azyklischer Teilgraph $D' = (V, A')$ mit $|A'|$ maximal

Feedback Arc Set

Idee:
- finde maximalen azyklischen Teilgraphen
- invertiere Richtung der übrigen Kanten

Maximum Acyclic Subgraph

geg: gerichteter Graph $D = (V, A)$
ges: azyklischer Teilgraph $D' = (V, A')$ mit $|A'|$ maximal

Minimum Feedback Arc Set (FAS)

geg: gerichteter Graph $D = (V, A)$
ges: $A_f \subset A$, mit $D_f = (V, A \setminus A_f)$ azyklisch und $|A_f|$ minimal

Feedback Arc Set

Idee:
- finde maximalen azyklischen Teilgraphen
- invertiere Richtung der übrigen Kanten

Maximum Acyclic Subgraph

geg: gerichteter Graph $D = (V, A)$
ges: azyklischer Teilgraph $D' = (V, A')$ mit $|A'|$ maximal

Minimum Feedback Arc Set (FAS)

geg: gerichteter Graph $D = (V, A)$
ges: $A_f \subset A$, mit $D_f = (V, A \setminus A_f)$ azyklisch und $|A_f|$ minimal

Minimum Feedback Set (FS)

geg: gerichteter Graph $D = (V, A)$
ges: $A_f \subset A$, mit $D_f = (V, A \setminus A_f \cup \mathrm{rev}(A_f))$ azyklisch und $|A_f|$ minimal

Feedback Arc Set

Idee:
- finde maximalen azyklischen Teilgraphen
- invertiere Richtung der übrigen Kanten

Maximum Acyclic Subgraph

geg: gerichteter Graph $D = (V, A)$
ges: azyklischer Teilgraph $D' = (V, A')$ mit $|A'|$ maximal

Minimum Feedback Arc Set (FAS)

geg: gerichteter Graph $D = (V, A)$
ges: $A_f \subset A$, mit $D_f = (V, A \setminus A_f)$ azyklisch und $|A_f|$ minimal

Minimum Feedback Set (FS)

geg: gerichteter Graph $D = (V, A)$
ges: $A_f \subset A$, mit $D_f = (V, A \setminus A_f \cup \mathrm{rev}(A_f))$ azyklisch und $|A_f|$ minimal

Alle drei Probleme sind NP-schwer!

Heuristik 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

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

  entferne v und $N(v)$ aus D.

return (V, A')

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

Heuristik 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

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

  entferne v und $N(v)$ aus D.

return (V, A')

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

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

- Warum gibt es keine Kreise in D'?
- Ist $D'' = (V, A' \cup \text{rev}(A \setminus A'))$ azyklisch?
- Wie ist die Laufzeit?
- Was kann man über $|A'|$ sagen?

Heuristik 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

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

  entferne v und $N(v)$ aus D.

return (V, A')

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

- $D' = (V, A')$ ist ein DAG
- $A \setminus A'$ ist ein feedback arc set
- Laufzeit $O(|V| + |A|)$
- $|A'| \geq |A|/2$

Heuristik 2 (Eades, Lin, Smyth 1993)

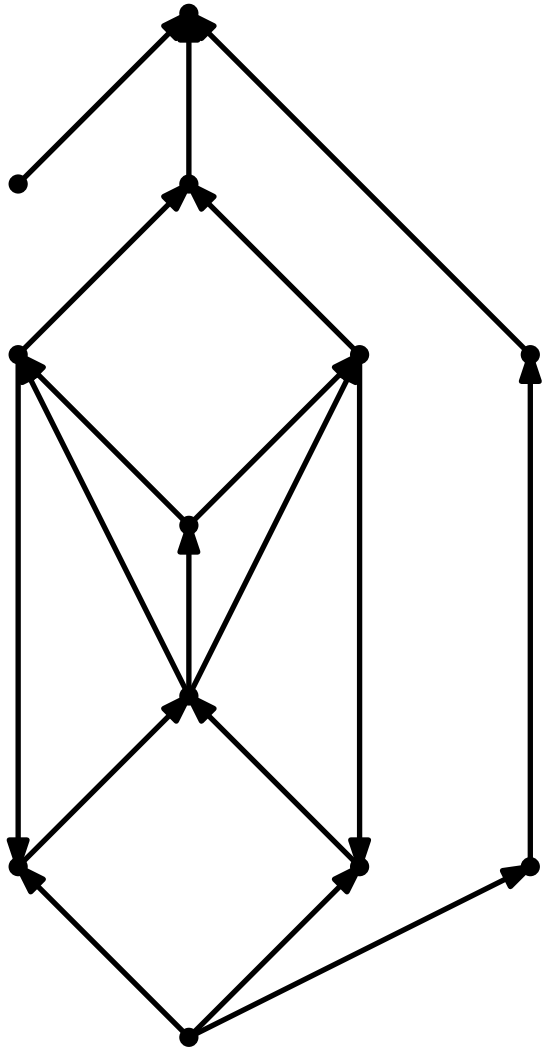
1 $A' := \emptyset$;

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

3 | **while** in V existiert eine Senke v **do**

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

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



Heuristik 2 (Eades, Lin, Smyth 1993)

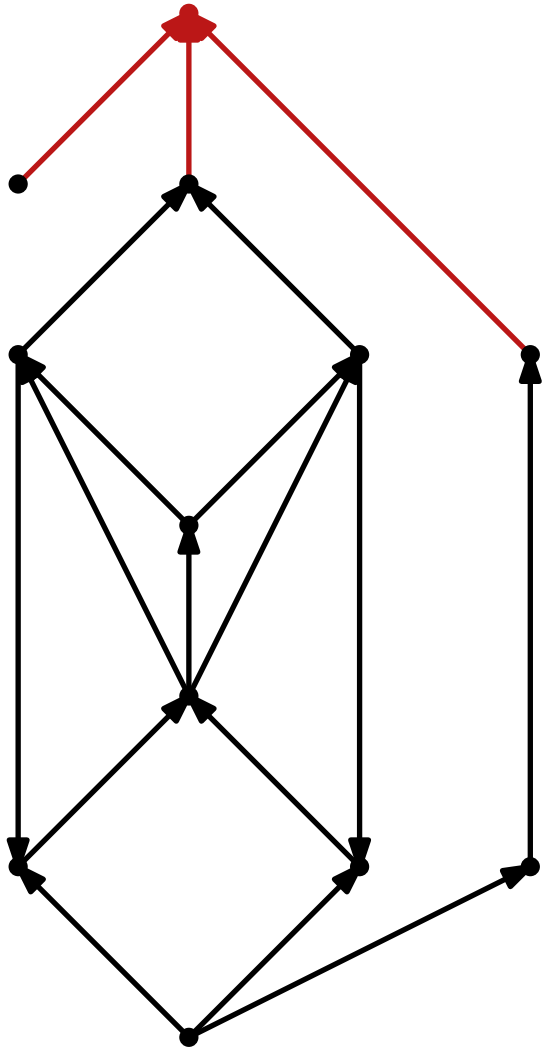
1 $A' := \emptyset$;

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

3 | **while** in V existiert eine Senke v **do**

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

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



Heuristik 2 (Eades, Lin, Smyth 1993)

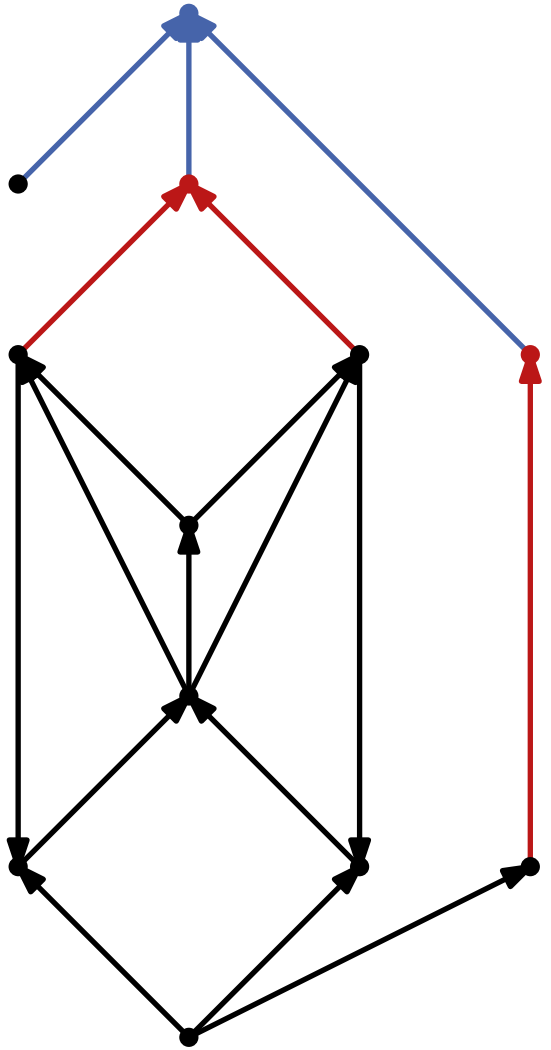
1 $A' := \emptyset$;

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

3 | **while** in V existiert eine Senke v **do**

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

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



Heuristik 2 (Eades, Lin, Smyth 1993)

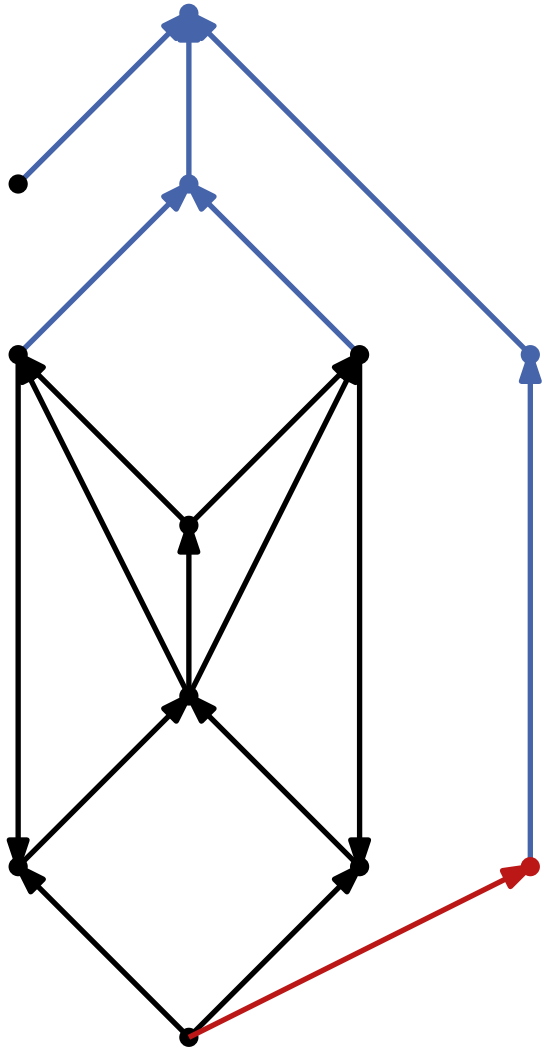
1 $A' := \emptyset$;

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

3 | **while** in V existiert eine Senke v **do**

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

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



Heuristik 2 (Eades, Lin, Smyth 1993)

1 $A' := \emptyset$;

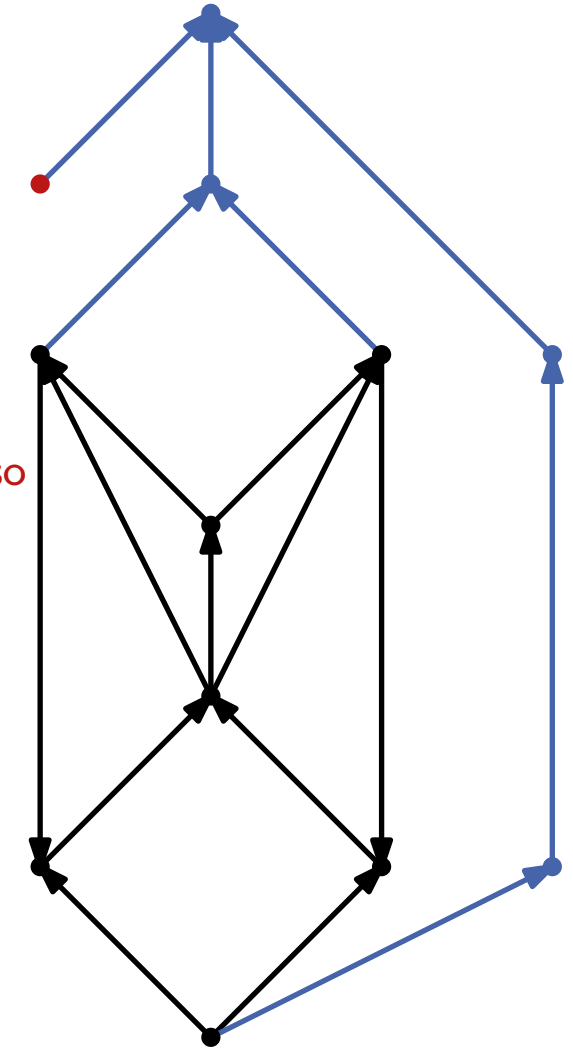
2 **while** $V \neq \emptyset$ **do**

3 | **while** in V existiert eine Senke v **do**

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

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

6 | Entferne alle isolierten Knoten aus V: $\{V, n, m\}_{\text{iso}}$



Heuristik 2 (Eades, Lin, Smyth 1993)

1 $A' := \emptyset$;

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

3 **while** in V existiert eine Senke v **do**

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

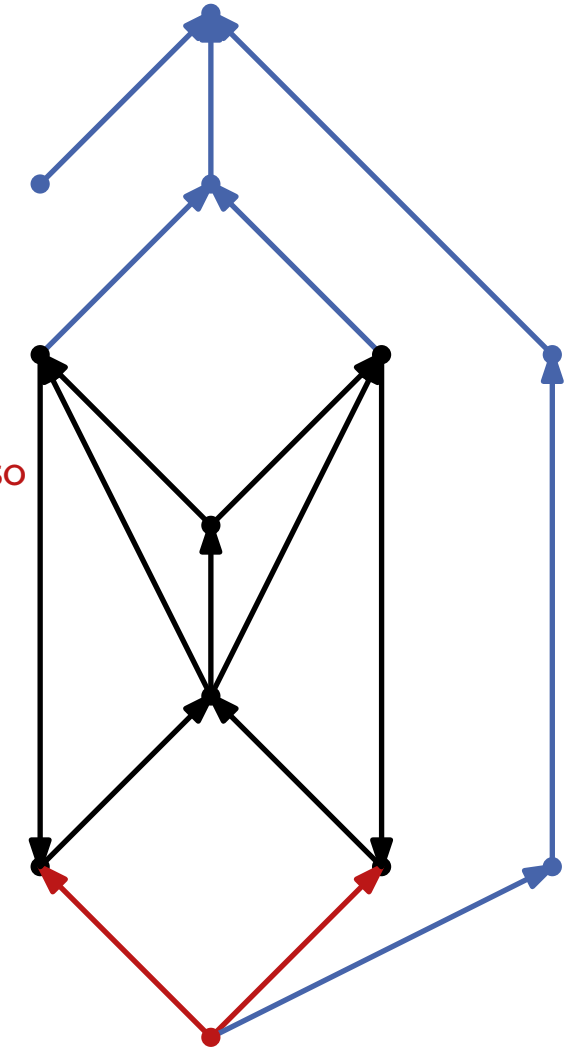
5 entferne v und $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$

6 Entferne alle isolierten Knoten aus V: $\{V, n, m\}_{\text{iso}}$

7 **while** in V existiert eine Quelle v **do**

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

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



Heuristik 2 (Eades, Lin, Smyth 1993)

Karlsruhe Institute of Technology

1 $A' := \emptyset$;

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

3 **while** in V existiert eine Senke v **do**

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

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

6 Entferne alle isolierten Knoten aus V: $\{V, n, m\}_{\text{iso}}$

7 **while** in V existiert eine Quelle v **do**

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

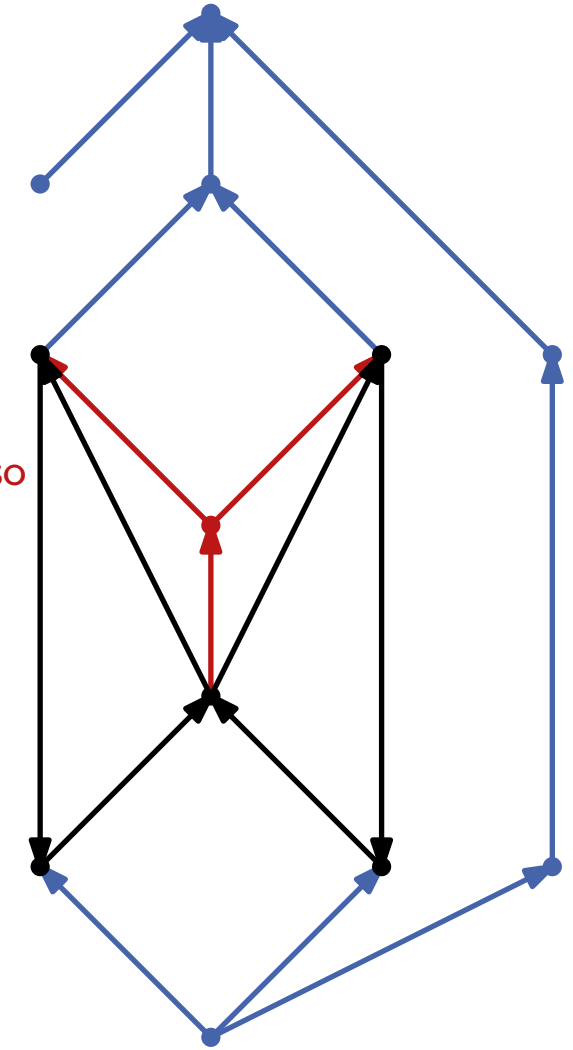
9 entferne v und $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

11 sei $v \in V$ mit $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$ maximal;

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

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



Heuristik 2 (Eades, Lin, Smyth 1993)

Karlsruhe Institute of Technology

1 $A' := \emptyset$;

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

3 **while** in V existiert eine Senke v **do**

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

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

6 Entferne alle isolierten Knoten aus V: $\{V, n, m\}_{\text{iso}}$

7 **while** in V existiert eine Quelle v **do**

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

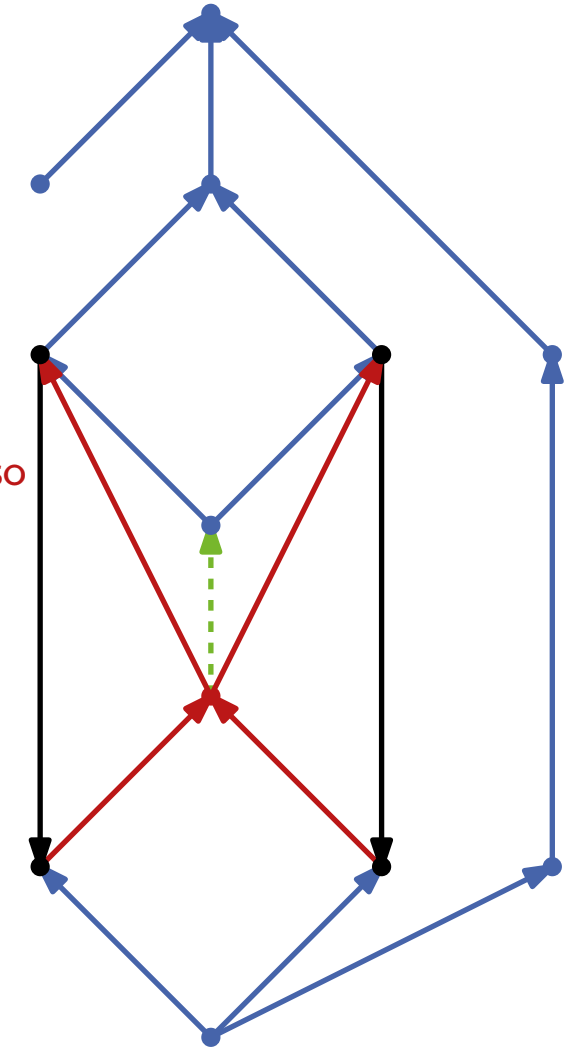
9 entferne v und $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

11 sei $v \in V$ mit $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$ maximal;

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

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



Heuristik 2 (Eades, Lin, Smyth 1993)

1 $A' := \emptyset$;

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

3 **while** in V existiert eine Senke v **do**

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

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

6 Entferne alle isolierten Knoten aus V: $\{V, n, m\}_{\text{iso}}$

7 **while** in V existiert eine Quelle v **do**

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

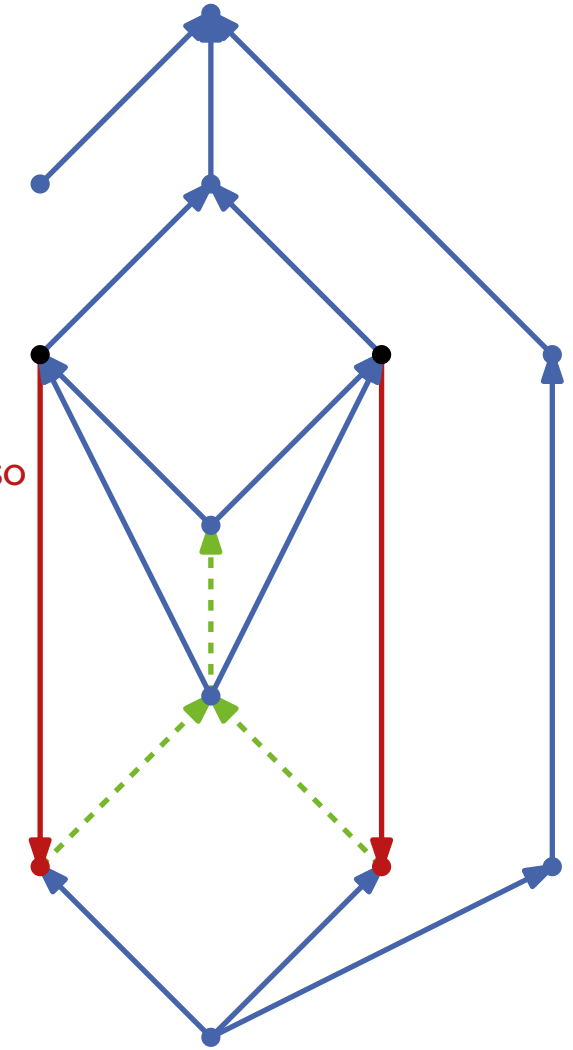
9 entferne v und $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

11 sei $v \in V$ mit $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$ maximal;

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

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



Heuristik 2 (Eades, Lin, Smyth 1993)

1 $A' := \emptyset$;

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

3 **while** in V existiert eine Senke v **do**

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

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

6 Entferne alle isolierten Knoten aus V: $\{V, n, m\}_{\text{iso}}$

7 **while** in V existiert eine Quelle v **do**

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

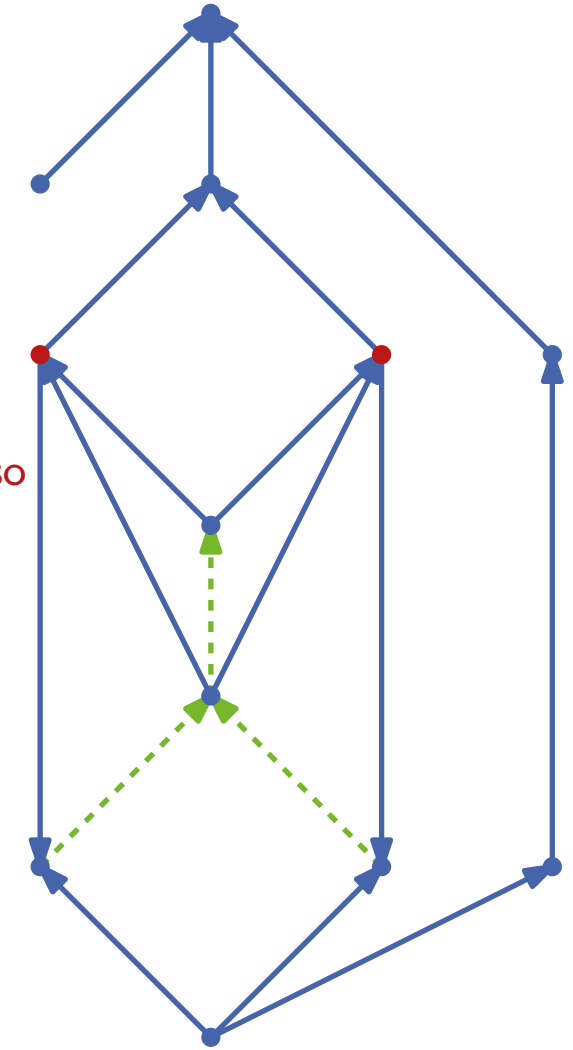
9 entferne v und $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

11 sei $v \in V$ mit $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$ maximal;

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

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



Heuristik 2 (Eades, Lin, Smyth 1993)

1 $A' := \emptyset$;

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

3 **while** in V existiert eine Senke v **do**

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

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

6 Entferne alle isolierten Knoten aus V: $\{V, n, m\}_{\text{iso}}$

7 **while** in V existiert eine Quelle v **do**

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

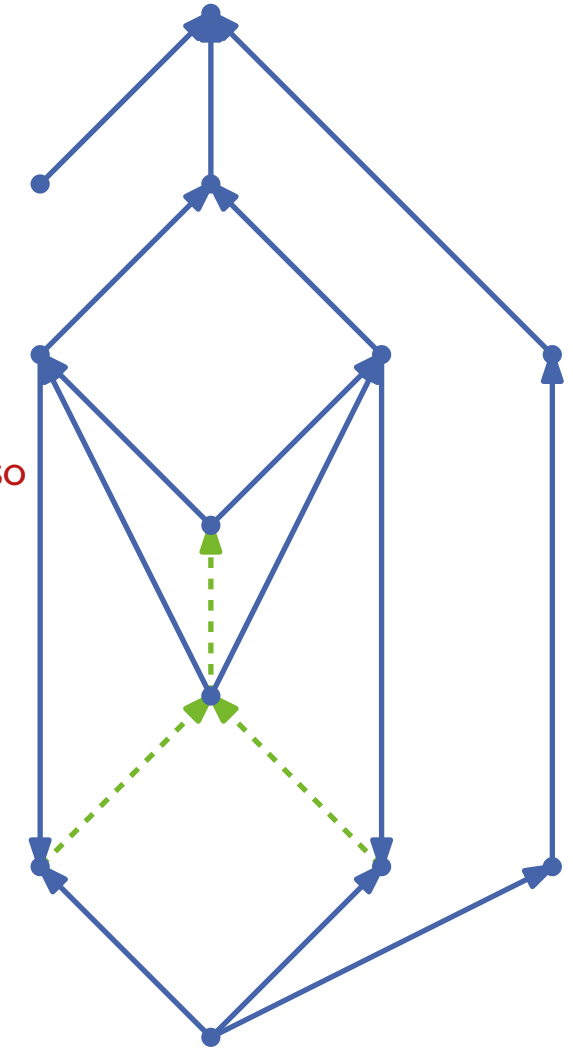
9 entferne v und $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

11 sei $v \in V$ mit $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$ maximal;

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

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



Heuristik 2 (Eades, Lin, Smyth 1993)

Karlsruhe Institute of Technology

1 $A' := \emptyset$;

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

3 **while** in V existiert eine Senke v **do**

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

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

6 Entferne alle isolierten Knoten aus V: $\{V, n, m\}_{\text{iso}}$

7 **while** in V existiert eine Quelle v **do**

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

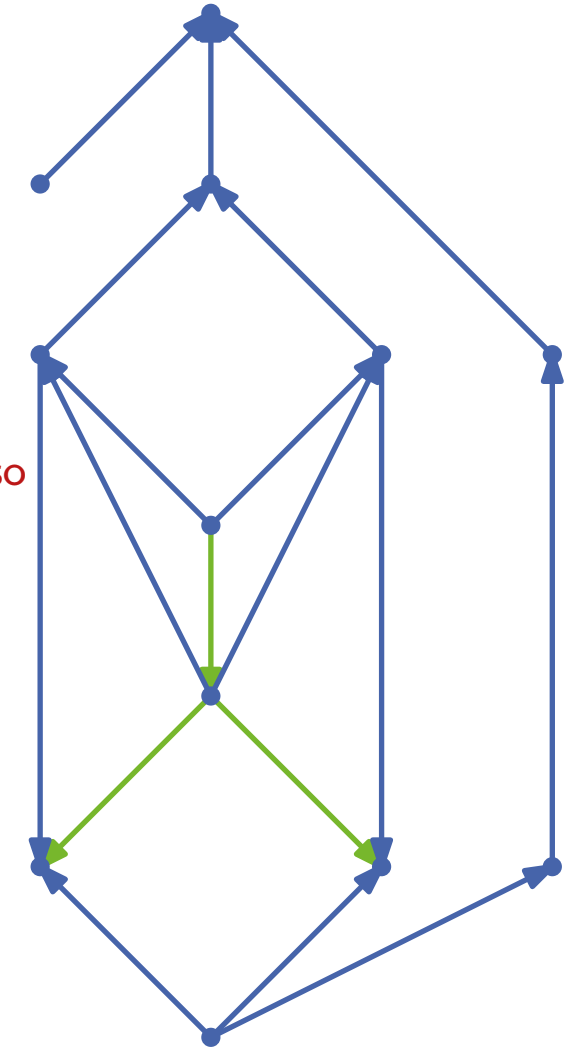
9 entferne v und $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

11 sei $v \in V$ mit $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$ maximal;

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

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



Heuristik 2 – Analyse

Satz 1: Sei $D = (V, A)$ ein zusammenhängender, gerichteter Graph ohne 2-Kreise. Dann berechnet die Heuristik 2 eine Kantenmenge A' mit $|A'| \geq |A|/2 + |V|/6$.
Die Laufzeit liegt in $O(|A|)$.

Heuristik 2 – Analyse

Satz 1: Sei $D = (V, A)$ ein zusammenhängender, gerichteter Graph ohne 2-Kreise. Dann berechnet die Heuristik 2 eine Kantenmenge A' mit $|A'| \geq |A|/2 + |V|/6$.
Die Laufzeit liegt in $O(|A|)$.

Weitere Methoden:

- $|A'| \geq |A| \left(1/2 + \Omega\left(\frac{1}{\sqrt{\deg_{\max}(D)}}\right)\right)$ (Berger, Shor 1990)
- exakte Lösung mit ganzzahliger linearer Programmierung, branch-and-cut Technik (Grötschel et al. 1985)

Heuristik 2 – Analyse

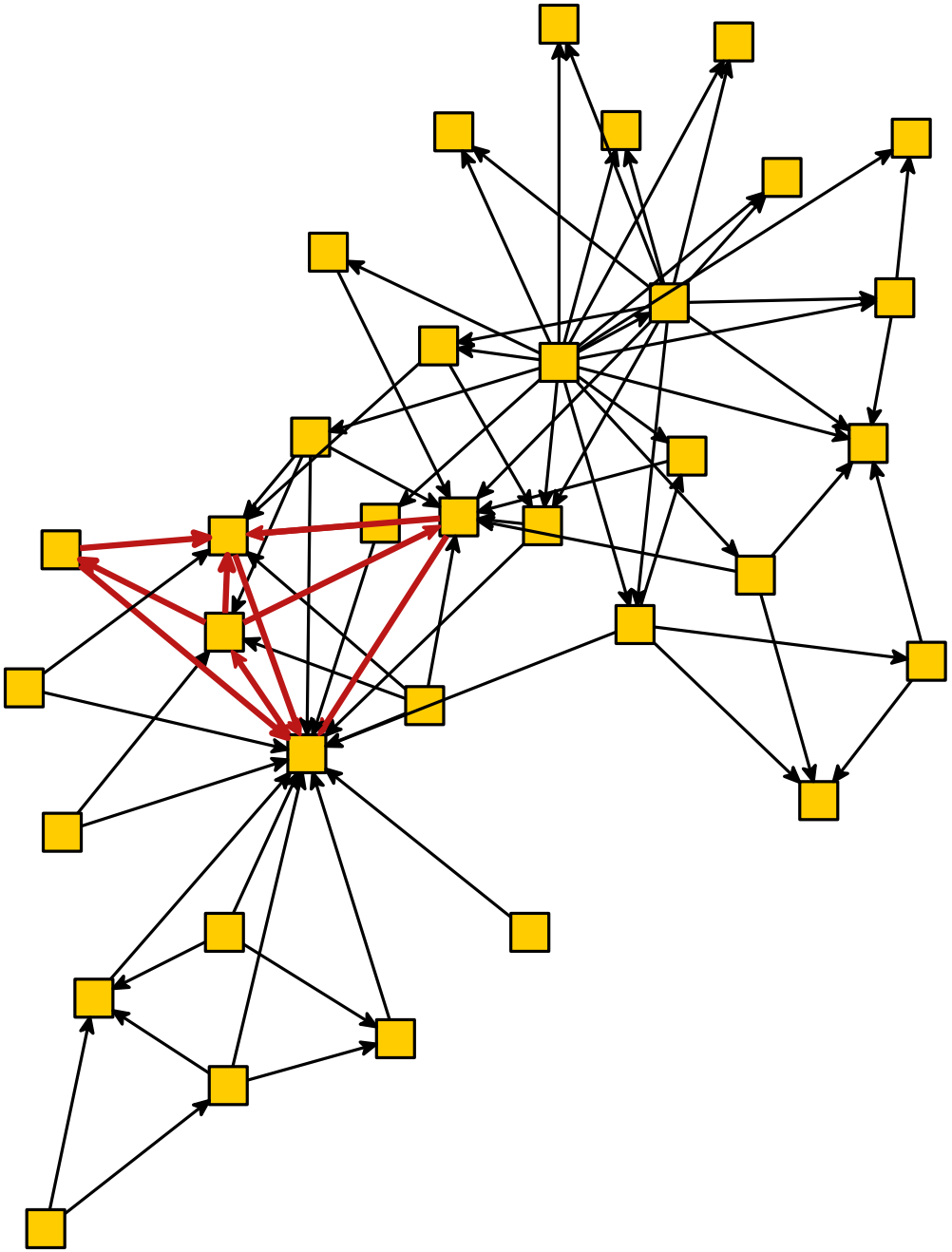
Satz 1: Sei $D = (V, A)$ ein zusammenhängender, gerichteter Graph ohne 2-Kreise. Dann berechnet die Heuristik 2 eine Kantenmenge A' mit $|A'| \geq |A|/2 + |V|/6$.
Die Laufzeit liegt in $O(|A|)$.

Weitere Methoden:

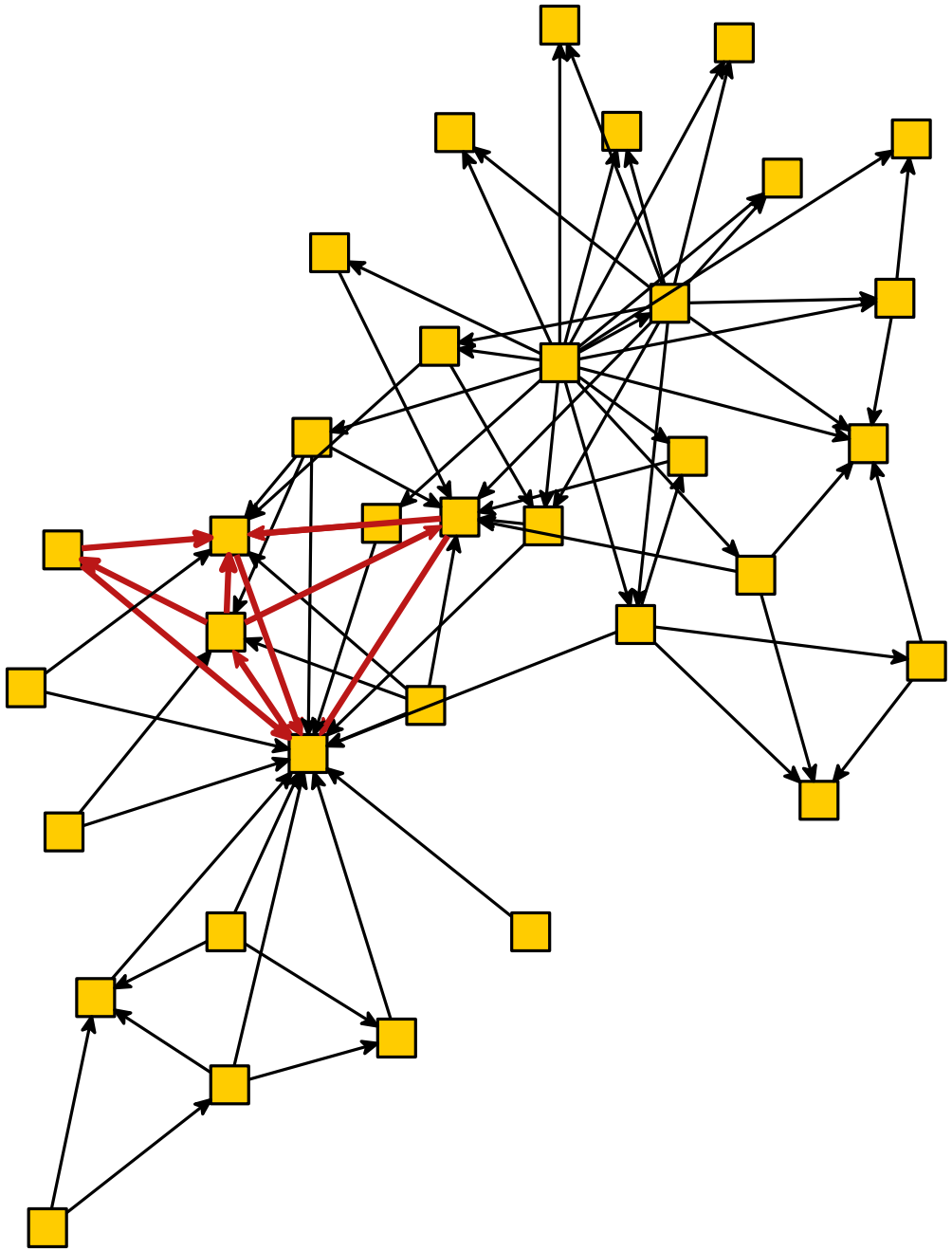
- $|A'| \geq |A| \left(1/2 + \Omega\left(\frac{1}{\sqrt{\deg_{\max}(D)}}\right)\right)$ (Berger, Shor 1990)
- exakte Lösung mit ganzzahliger linearer Programmierung, branch-and-cut Technik (Grötschel et al. 1985)

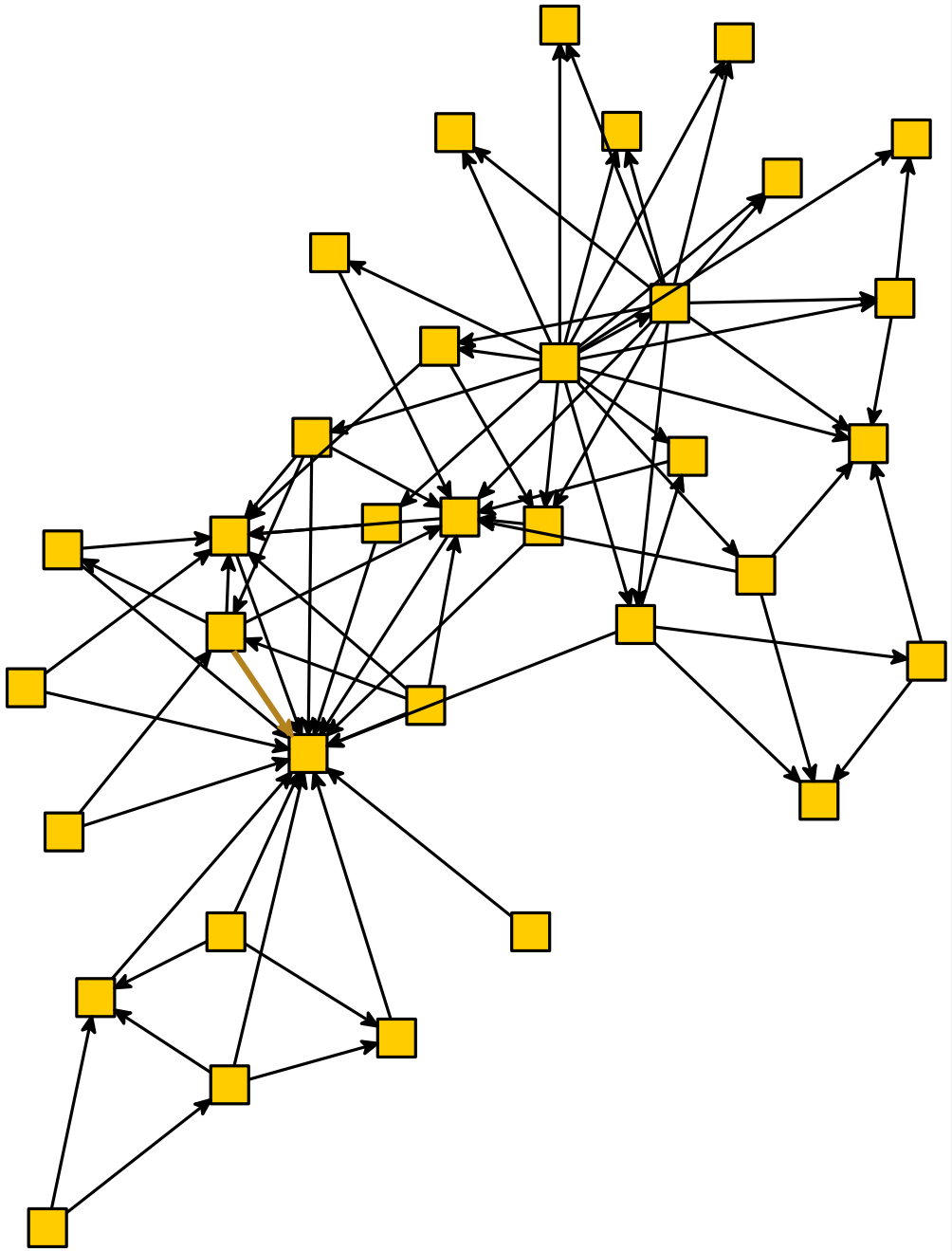
Für $|A| \in O(|V|)$ ist Heuristik 2 mindestens genauso gut.

Beispielgraph

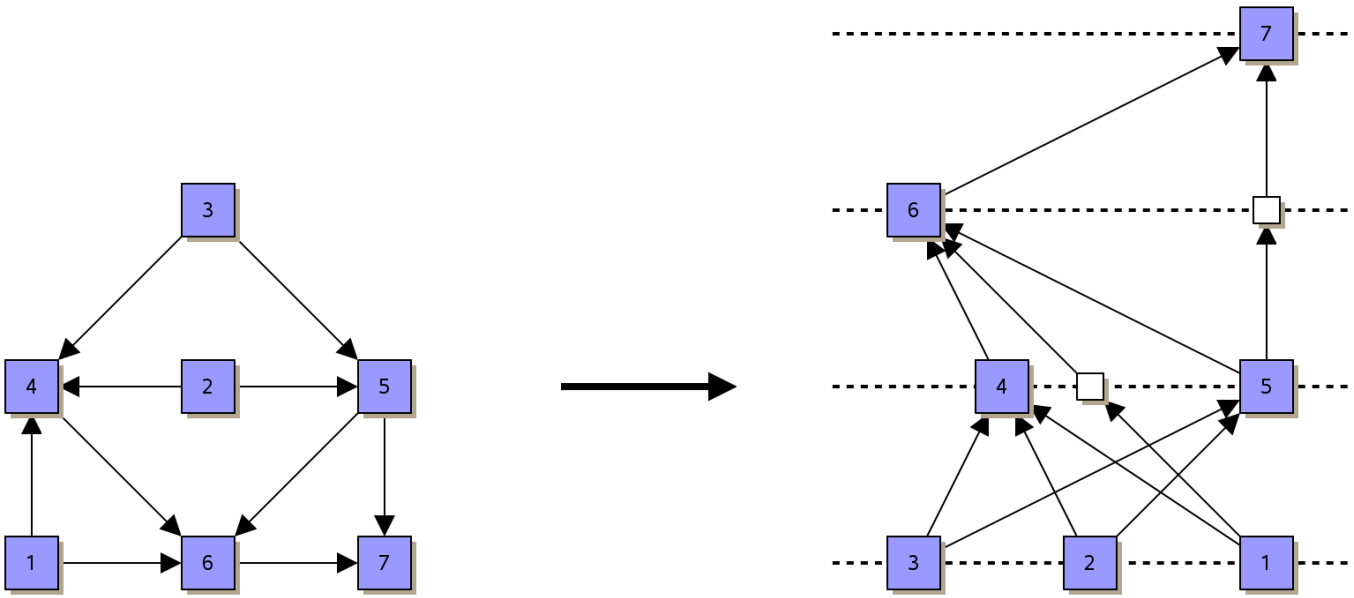


Beispielgraph





Schritt 2: Lagenzuordnung



Wie würden Sie vorgehen?

Lagenzuordnung

Geg.: gerichteter azyklischer Graph (DAG) $D = (V, A)$

Ges.: Partition der Knotenmenge V in disjunkte **Lagen**
$L_1, \ldots, L_h$ mit $(u, v) \in A$, $u \in L_i$, $v \in L_j \Rightarrow i < j$

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

Lagenzuordnung

Geg.: gerichteter azyklischer Graph (DAG) $D = (V, A)$

Ges.: Partition der Knotenmenge V in disjunkte **Lagen**
$L_1, \ldots, L_h$ mit $(u, v) \in A$, $u \in L_i$, $v \in L_j \Rightarrow i < j$

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

Kriterien

- minimiere Lagenanzahl h ($=$ Höhe des Layouts)
- minimiere Breite, d.h. $\max\{|L_i| \mid 1 \leq i \leq h\}$
- minimiere längste Kante, d.h.
  $\max\{j - i \mid (u, v) \in A,\ u \in L_i,\ v \in L_j\}$
- minimiere Gesamtkantenlänge ($\hat{=}$ Anzahl Dummyknoten)

Höhenminimierung

Idee: ordne jeden Knoten v der Lage L_i zu, für die i die Länge des längsten einfachen Wegs von einer Quelle zu v ist

- alle Vorgängerknoten liegen unterhalb von v
- die Gesamthöhe h ist minimal

Höhenminimierung

Idee: ordne jeden Knoten v der Lage L_i zu, für die i die Länge des längsten einfachen Wegs von einer Quelle zu v ist

- alle Vorgängerknoten liegen unterhalb von v
- die Gesamthöhe h ist minimal

Algorithmus

- $L_1 \leftarrow$ Menge aller Quellen in D
- setze $y(u) \leftarrow \max_{v \in N^{\leftarrow}(u)}\{y(v)\} + 1$

Höhenminimierung

Idee: ordne jeden Knoten v der Lage L_i zu, für die i die Länge des längsten einfachen Wegs von einer Quelle zu v ist

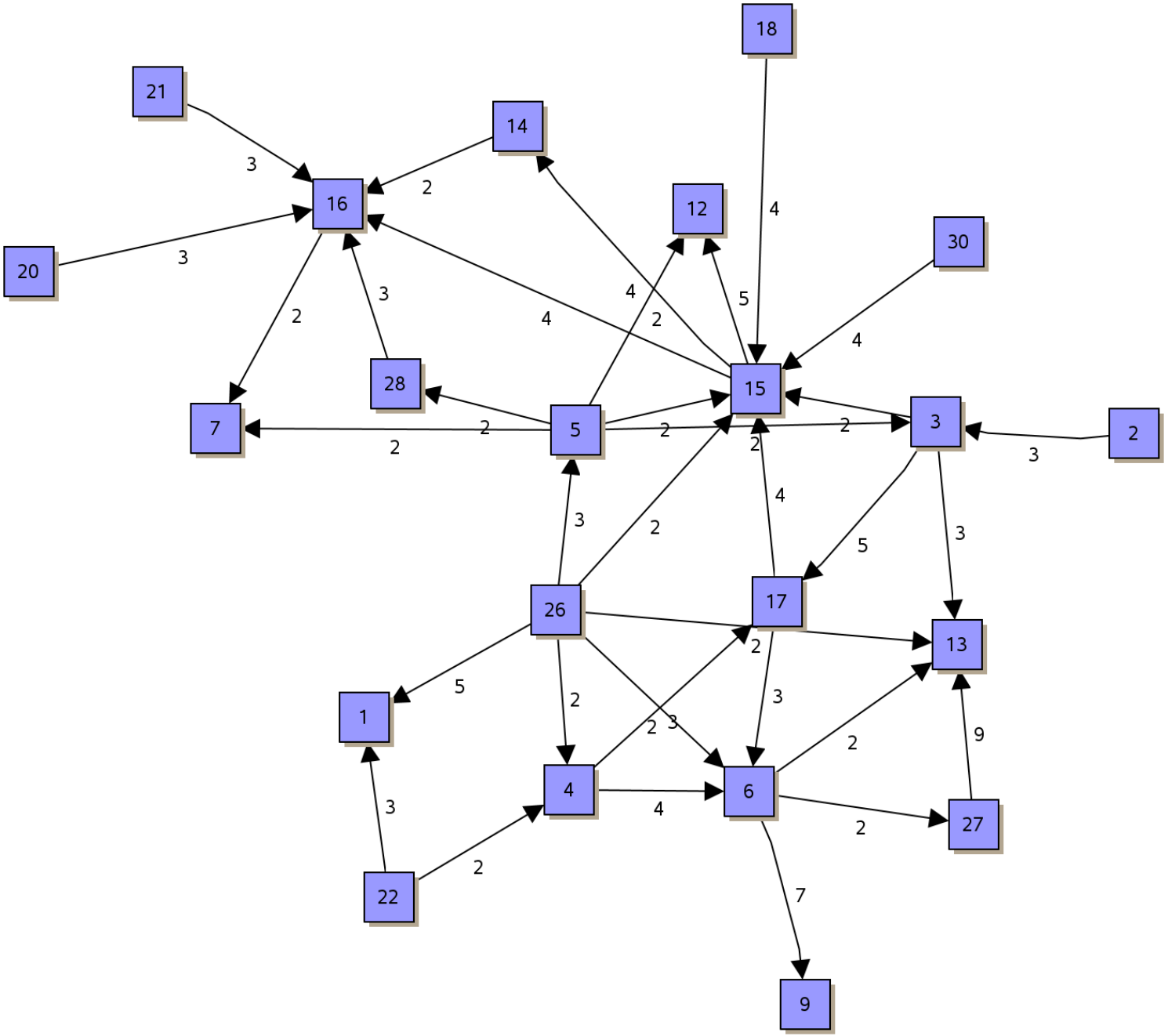
- alle Vorgängerknoten liegen unterhalb von v
- die Gesamthöhe h ist minimal

Algorithmus

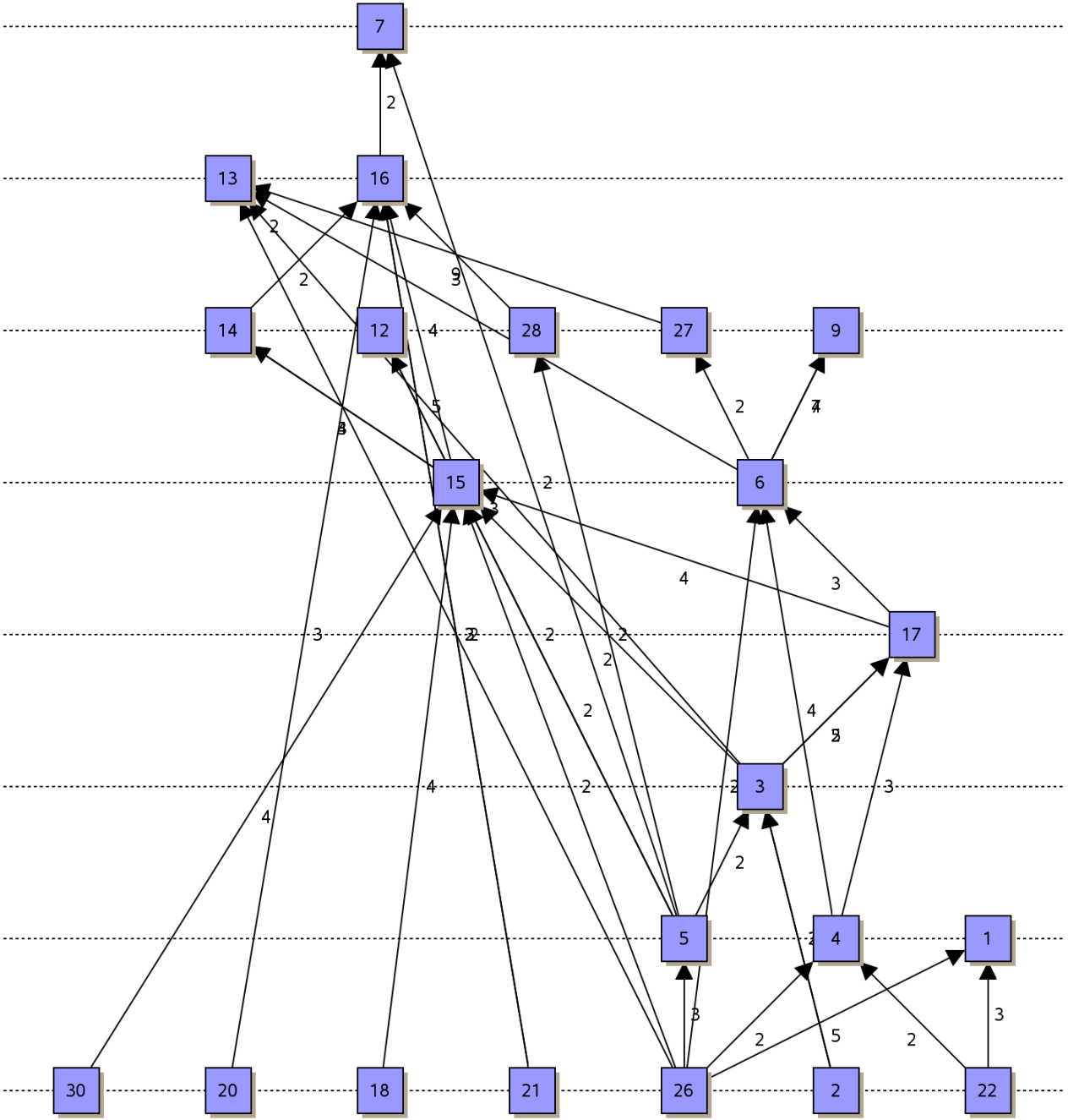
- $L_1 \leftarrow$ Menge aller Quellen in D
- setze $y(u) \leftarrow \max_{v \in N^{\leftarrow}(u)}\{y(v)\} + 1$

Wie kann man das in Linearzeit $O(|V| + |A|)$ implementieren?

Beispiel


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

Beispiel



Minimierung der Gesamtkantenlänge

Karlsruhe Institute of Technology

Lässt sich als ganzzahliges lineares Programm formulieren:

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

Minimierung der Gesamtkantenlänge

Karlsruhe Institute of Technology

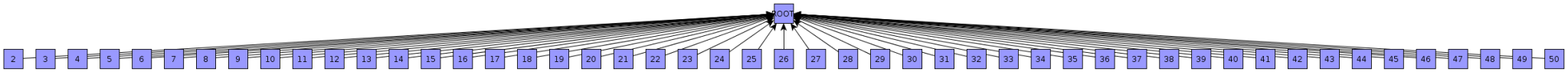
Lässt sich als ganzzahliges lineares Programm formulieren:

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

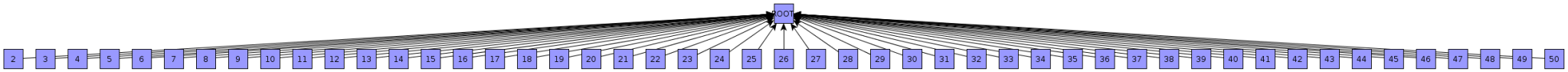
Man kann zeigen:

- Constraint-Matrix ist **total unimodular**
- $\Rightarrow$ Lösung des relaxierten linearen Programms ist ganzzahlig
- Gesamtkantenlänge kann mit LP-Solver in polynomieller Zeit minimiert werden

Problem der Höhen-/Längenminimierung



Problem der Höhen-/Längenminimierung



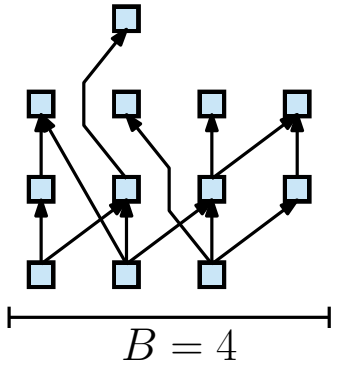
$\rightarrow$ beschränke die Breite!

Lagenzuordnung bei fester Breite

Fixed-Width Layer Assignment

geg: azyklischer gerichteter Graph $D = (V, A)$, Breite B

ges: höhenminimale Lagenzuordnung $\mathcal{L}$ mit maximal B Knoten pro Lage

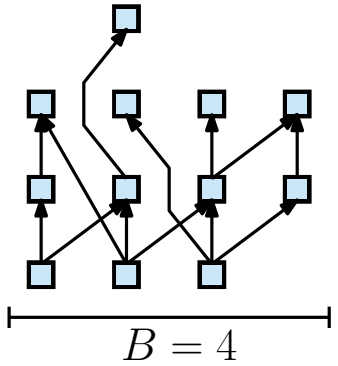


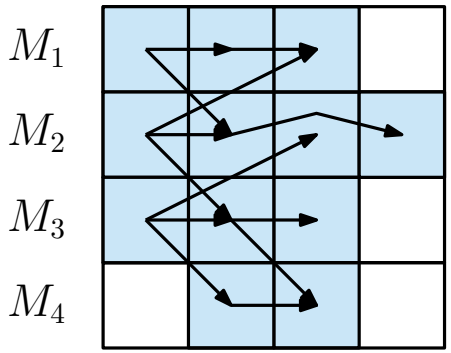
Lagenzuordnung bei fester Breite

Fixed-Width Layer Assignment

geg: azyklischer gerichteter Graph $D = (V, A)$, Breite B

ges: höhenminimale Lagenzuordnung $\mathcal{L}$ mit maximal B Knoten pro Lage





$\rightarrow$ äquivalent zu folgendem Schedulingproblem:

Minimum Precedence Constrained Scheduling (MPCS)

geg: n Jobs $J_1, \ldots, J_n$ mit identischer Bearbeitungsdauer 1, Vorgängerbedingungen $J_i < J_k$, B identische Maschinen

ges: Schedule mit minimaler Gesamtlänge, der alle Vorgängerbedingungen einhält

Komplexität

Satz 2: Es ist NP-vollständig zu entscheiden, ob es für n Jobs $J_1, \ldots, J_n$ identischer Länge, partielle Vorgängerordnung $<$ und Maschinenanzahl B einen Schedule der Länge höchstens T gibt, selbst für $T = 3$.

Komplexität

Satz 2: Es ist NP-vollständig zu entscheiden, ob es für n Jobs $J_1, \ldots, J_n$ identischer Länge, partielle Vorgängerordnung $<$ und Maschinenanzahl B einen Schedule der Länge höchstens T gibt, selbst für $T = 3$.

Korollar: Falls $\mathcal{P} \neq \mathcal{NP}$ gibt es keinen polynomiellen Approximationsalgorithmus für MPCS mit Approximationsfaktor $< 4/3$.

Komplexität

Satz 2: Es ist NP-vollständig zu entscheiden, ob es für n Jobs $J_1, \ldots, J_n$ identischer Länge, partielle Vorgängerordnung $<$ und Maschinenanzahl B einen Schedule der Länge höchstens T gibt, selbst für $T = 3$.

Korollar: Falls $\mathcal{P} \neq \mathcal{NP}$ gibt es keinen polynomiellen Approximationsalgorithmus für MPCS mit Approximationsfaktor $< 4/3$.

Satz 3: MPCS hat einen polynomiellen Approximationsalgorithmus mit Faktor $\leq 2 - \frac{1}{B}$.

Komplexität

Satz 2: Es ist NP-vollständig zu entscheiden, ob es für n Jobs $J_1, \ldots, J_n$ identischer Länge, partielle Vorgängerordnung $<$ und Maschinenanzahl B einen Schedule der Länge höchstens T gibt, selbst für $T = 3$.

Korollar: Falls $\mathcal{P} \neq \mathcal{NP}$ gibt es keinen polynomiellen Approximationsalgorithmus für MPCS mit Approximationsfaktor $< 4/3$.

Satz 3: MPCS hat einen polynomiellen Approximationsalgorithmus mit Faktor $\leq 2 - \frac{1}{B}$.

List-Scheduling-Algorithmus:

- ordne Jobs beliebig als Liste $\mathcal{L}$
- wenn Maschine frei ist, wähle ersten verfügbaren Job aus $\mathcal{L}$; Maschine idle falls kein Job verfügbar

Zusammenfassung