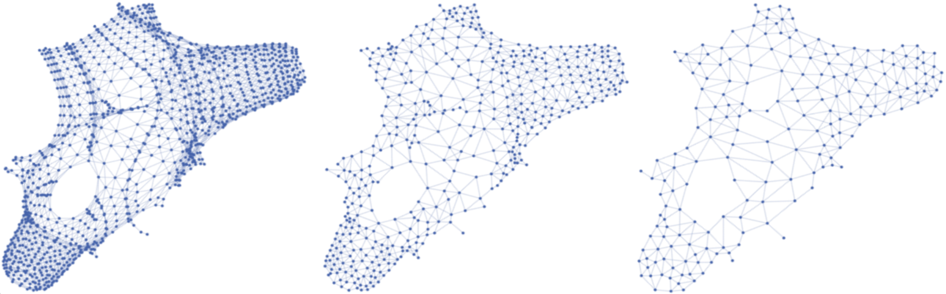
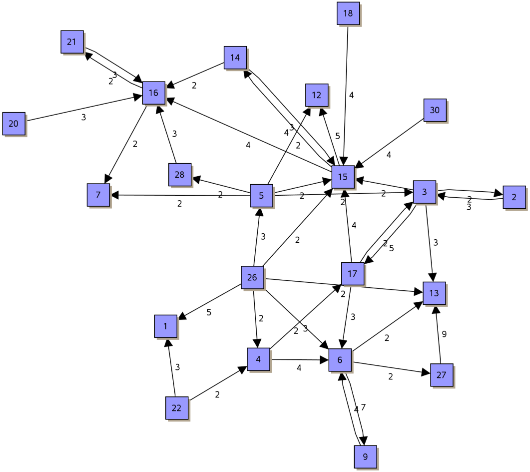
Algorithmen zur Visualisierung von Graphen

Lagenlayouts I

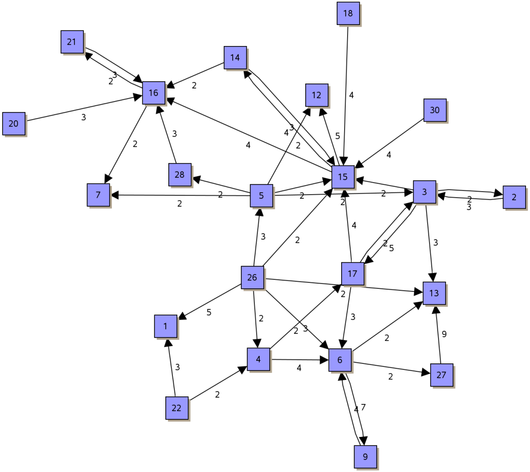
Marcus Krug | WS 2011/12

INSTITUTE OF THEORETICAL INFORMATICS
KARLSRUHE INSTITUTE OF TECHNOLOGY (KIT)




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

E-Mail-Graph der Fakultät für Informatik



Lagenlayouts

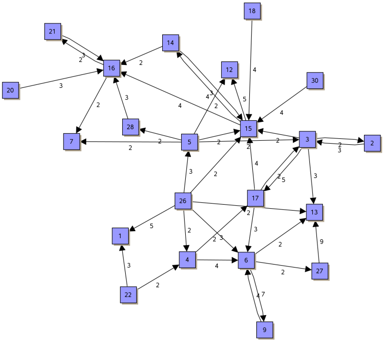
Problemstellung

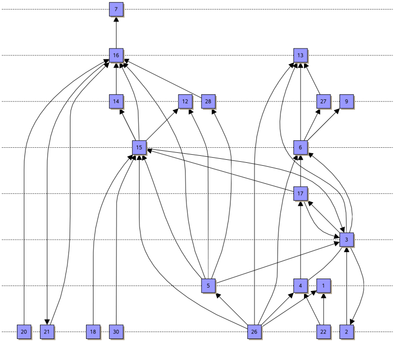
- Gegeben: gerichteter Graph $D = (V, A)$
- Gesucht: Zeichnung von D, die Hierarchie möglichst gut wiedergibt

Lagenlayouts

Problemstellung

- Gegeben: gerichteter Graph $D = (V, A)$
- Gesucht: Zeichnung von D, die Hierarchie möglichst gut wiedergibt





Lagenlayouts

Problemstellung

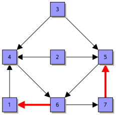
- Gegeben: gerichteter Graph $D = (V, A)$
- Gesucht: Zeichnung von D, die Hierarchie möglichst gut wiedergibt

Desiderata

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

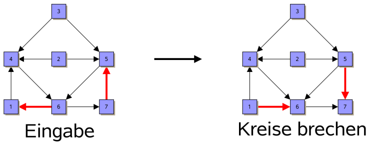
! Kriterien widersprechen sich

Klassisches Vorgehen (Sugiyama)

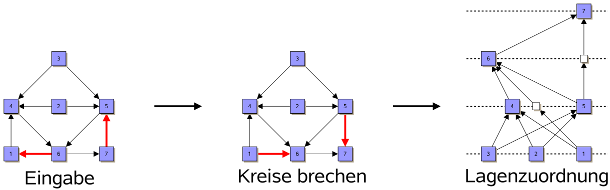


Eingabe

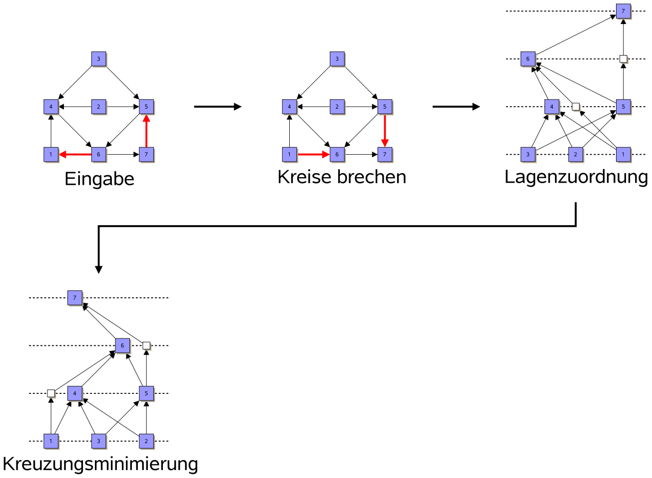
Klassisches Vorgehen (Sugiyama)



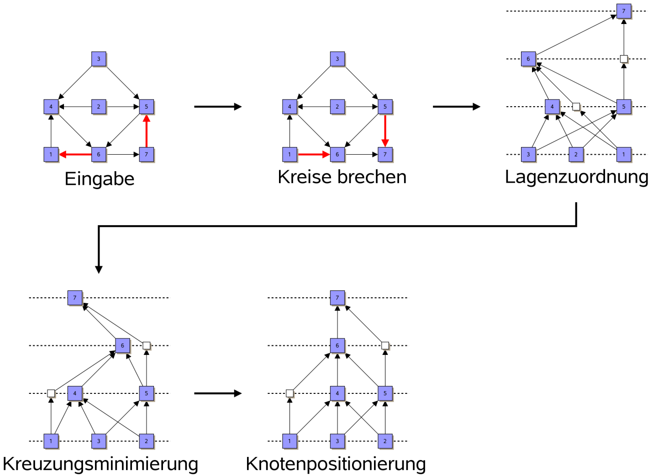
Klassisches Vorgehen (Sugiyama)



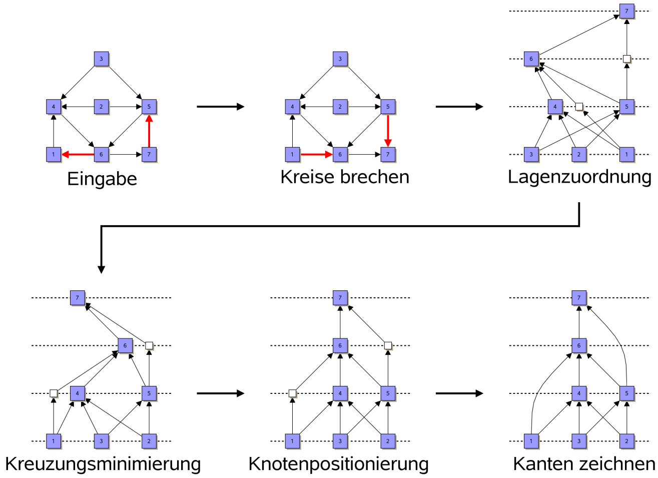
Klassisches Vorgehen (Sugiyama)



Klassisches Vorgehen (Sugiyama)

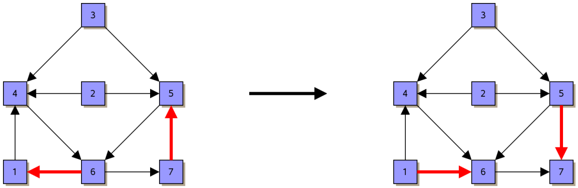


Klassisches Vorgehen (Sugiyama)



Kreise brechen

1. Schritt: Behandlung von Gerichteten Kreisen



Behandlung von Gerichteten Kreisen

Vorgehen

- Finde maximalen azyklischen Subgraph durch Entfernen von Kanten A_f
- füge Inversen zu Kanten in A_f ein

Behandlung von Gerichteten Kreisen

Vorgehen

- Finde maximalen azyklischen Subgraph durch Entfernen von Kanten A_f
- füge Inversen zu Kanten in A_f ein

Problem MINIMUM FEEDBACK ARC SET (FAS):

- Gegeben: gerichteter Graph $D = (V, A)$
- Finde minimale Menge $A_f \subseteq A$, so dass $D - A_f$ azyklisch ist

Behandlung von Gerichteten Kreisen

Vorgehen

- Finde maximalen azyklischen Subgraph durch Entfernen von Kanten A_f
- füge Inversen zu Kanten in A_f ein

Problem MINIMUM FEEDBACK ARC SET (FAS):

- Gegeben: gerichteter Graph $D = (V, A)$
- Finde minimale Menge $A_f \subseteq A$, so dass $D - A_f$ azyklisch ist

- FAS ist $\mathcal{NP}$-schwer

Behandlung von Gerichteten Kreisen

Greedy-Heuristik zur Berechnung eines azyklischen Graphen $D' = (V, A')$

(1) $A' := \emptyset$

(2) Betrachte Knoten in beliebiger Reihenfolge

füge entweder eingehende oder ausgehende Kanten zu A' hinzu (je nachdem welche Menge größer ist) und lösche Knoten

(3) $A_f := A \setminus A'$

Behandlung von Gerichteten Kreisen

Greedy-Heuristik zur Berechnung eines azyklischen Graphen $D' = (V, A')$

(1) $A' := \emptyset$

(2) Betrachte Knoten in beliebiger Reihenfolge

füge entweder eingehende oder ausgehende Kanten zu A' hinzu (je nachdem welche Menge größer ist) und lösche Knoten

(3) $A_f := A \setminus A'$

- Laufzeit $\mathcal{O}(n + m)$
- A' hat mindestens $|A|/2$ viele Kanten

Algorithmus 1: Greedy-Algorithmus (Eades, Lin & Smyth)

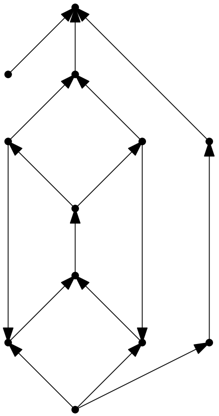
$A' := \emptyset$;

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

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

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

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



Algorithmus 1: Greedy-Algorithmus (Eades, Lin & Smyth)

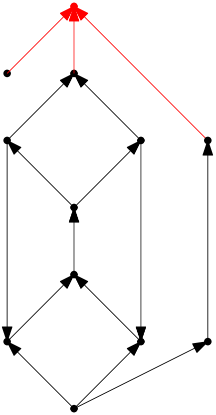
$A' := \emptyset$;

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

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

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

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



Algorithmus 1: Greedy-Algorithmus (Eades, Lin & Smyth)

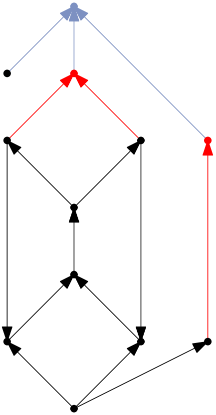
$A' := \emptyset$;

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

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

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

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



Algorithmus 1: Greedy-Algorithmus (Eades, Lin & Smyth)

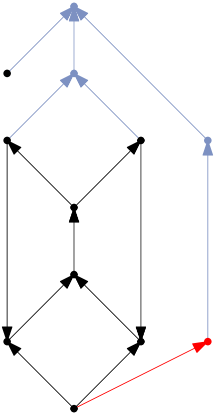
$A' := \emptyset$;

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

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

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

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



Algorithmus 1: Greedy-Algorithmus (Eades, Lin & Smyth)

$A' := \emptyset$;

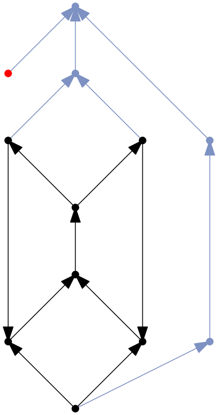
while $V \neq \emptyset$ **do**

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

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

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

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



Algorithmus 1: Greedy-Algorithmus (Eades, Lin & Smyth)

$A' := \emptyset$;

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

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

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

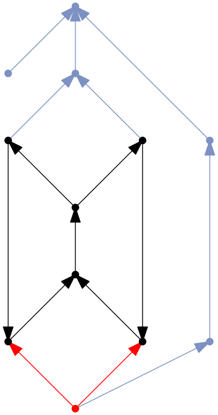
    entferne v und $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$

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

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

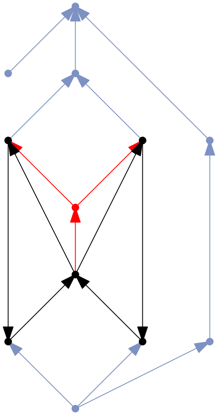
    $A' \leftarrow A' \cup N^{\rightarrow}(v)$

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



Algorithmus 1: Greedy-Algorithmus (Eades, Lin & Smyth)

```
A' := ∅;
while V ≠ ∅ do
    while in V existiert eine Senke v do
        A' ← A' ∪ N←(v)
        entferne v und N←(v): {V, n, m}_sink
    Entferne alle isolierten Knoten aus V: {V, n, m}_iso
    while in V existiert eine Quelle v do
        A' ← A' ∪ N→(v)
        entferne v und N→(v): {V, n, m}_source
    if V ≠ ∅ then
        sei v ∈ V mit |N→(v)| − |N←(v)| maximal;
        A' ← A' ∪ N→(v)
        entferne v und N→(v): {V, n, m}_{=,<}
```



Algorithmus 1: Greedy-Algorithmus (Eades, Lin & Smyth)

$A' := \emptyset$;

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

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

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

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

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

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

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

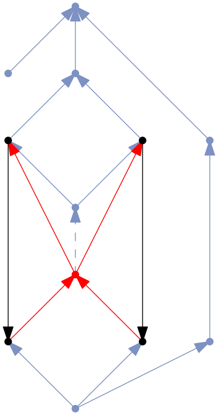
    entferne v und $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

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

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

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



Algorithmus 1: Greedy-Algorithmus (Eades, Lin & Smyth)

$A' := \emptyset$;

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

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

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

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

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

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

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

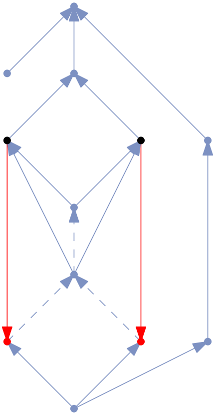
    entferne v und $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

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

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

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



Algorithmus 1: Greedy-Algorithmus (Eades, Lin & Smyth)

$A' := \emptyset$;

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

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

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

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

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

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

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

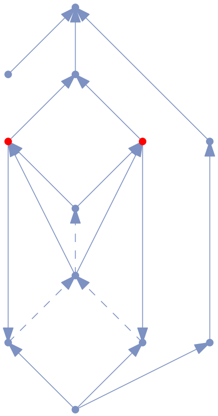
    entferne v und $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

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

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

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



Algorithmus 1: Greedy-Algorithmus (Eades, Lin & Smyth)

$A' := \emptyset$;

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

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

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

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

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

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

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

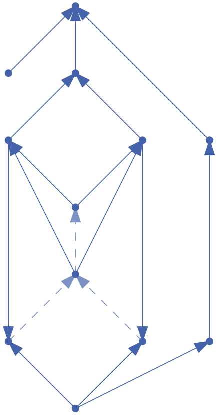
    entferne v und $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$

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

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

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

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



Behandlung von Gerichteten Kreisen

Verbesserte Greedy-Heuristik von Eades et al.

- Laufzeit $\mathcal{O}(n+m)$ Wie?
- A' hat mindestens $|A|/2 + |V|/6$ viele Kanten

Behandlung von Gerichteten Kreisen

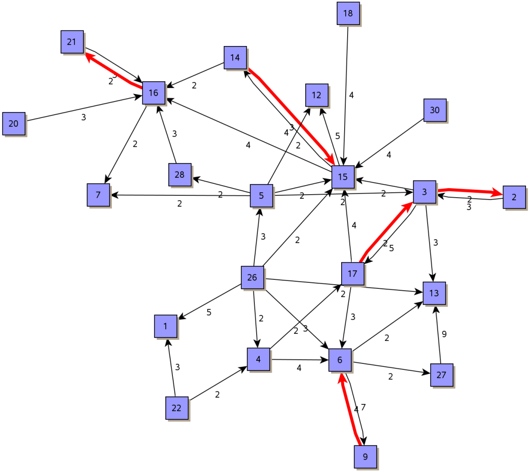
Verbesserte Greedy-Heuristik von Eades et al.

- Laufzeit $\mathcal{O}(n+m)$ Wie?
- A' hat mindestens $|A|/2+|V|/6$ viele Kanten

Weitere Methoden

- randomisiert: Zufällige Ordnung + Greedy: erwartet mindestens
$$\left(\frac{1}{2}+\Omega\left(\frac{1}{\sqrt{\Delta(G)}}\right)\right)|E|$$
[Berger & Shor, '90]
- Exakt: via Linear-Ordering Polytope + Cutting-Plane Method [Grötschel et al., '84]

E-Mail-Graph der Fakultät für Informatik



E-Mail-Graph der Fakultät für Informatik