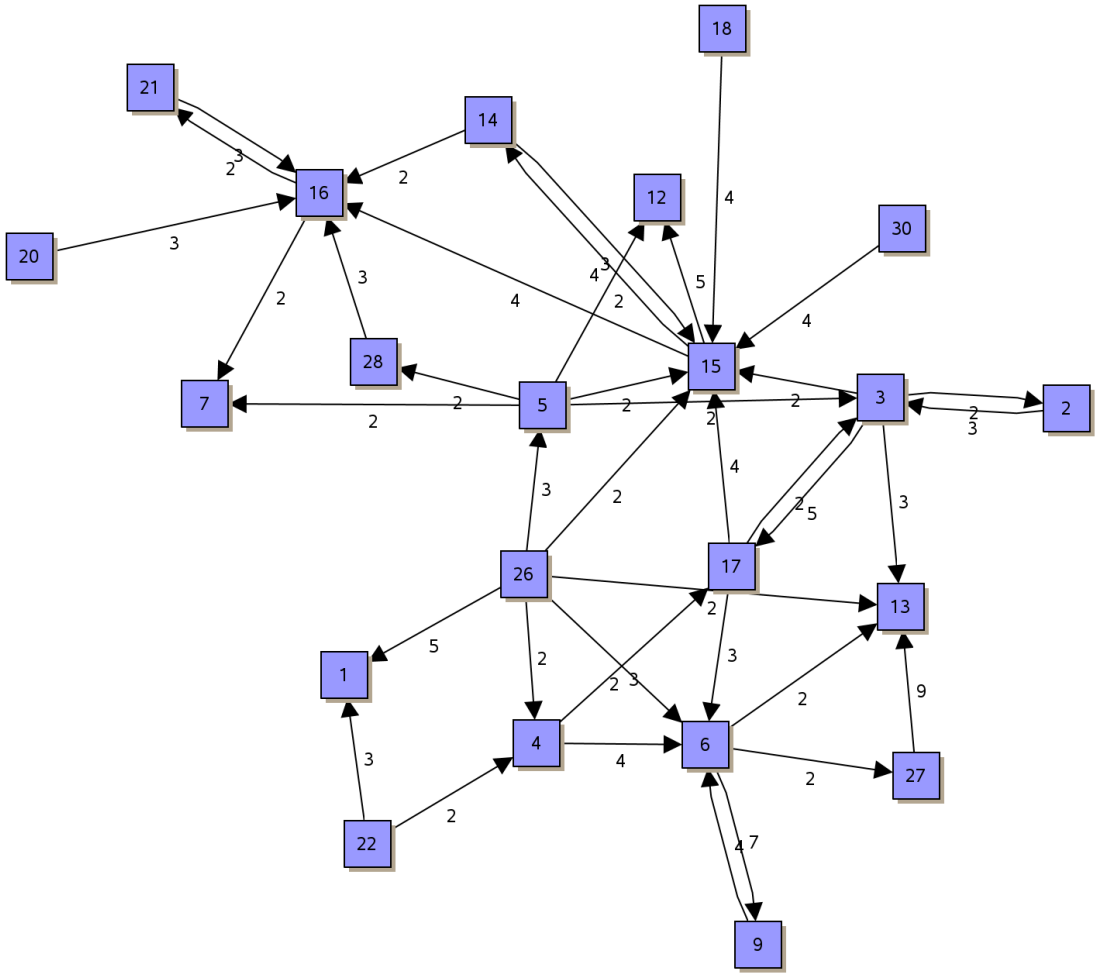
Algorithmen zur Visualisierung von Graphen
Lagenlayouts

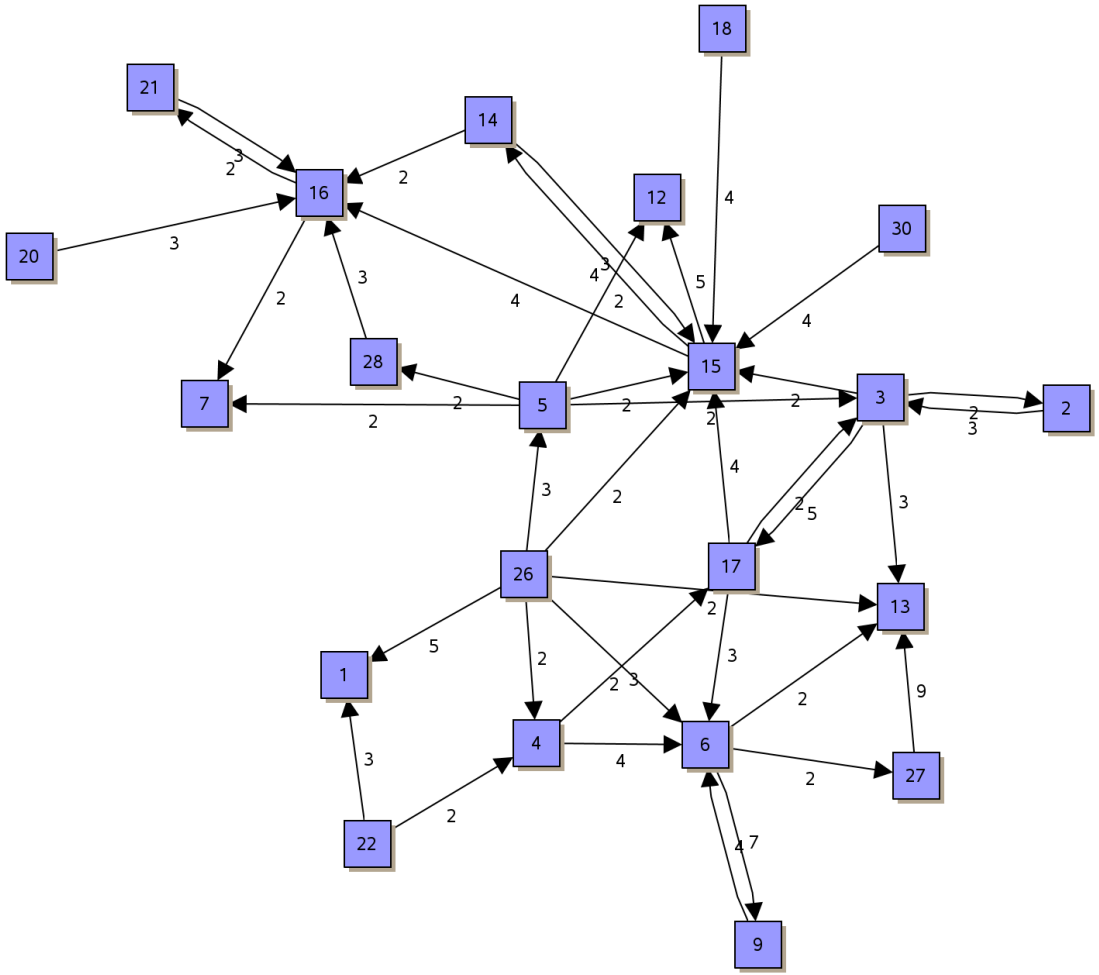
Marcus Krug

Lehrstuhl für Algorithmik I
Institut für Theoretische Informatik
Universität Karlsruhe (TH)

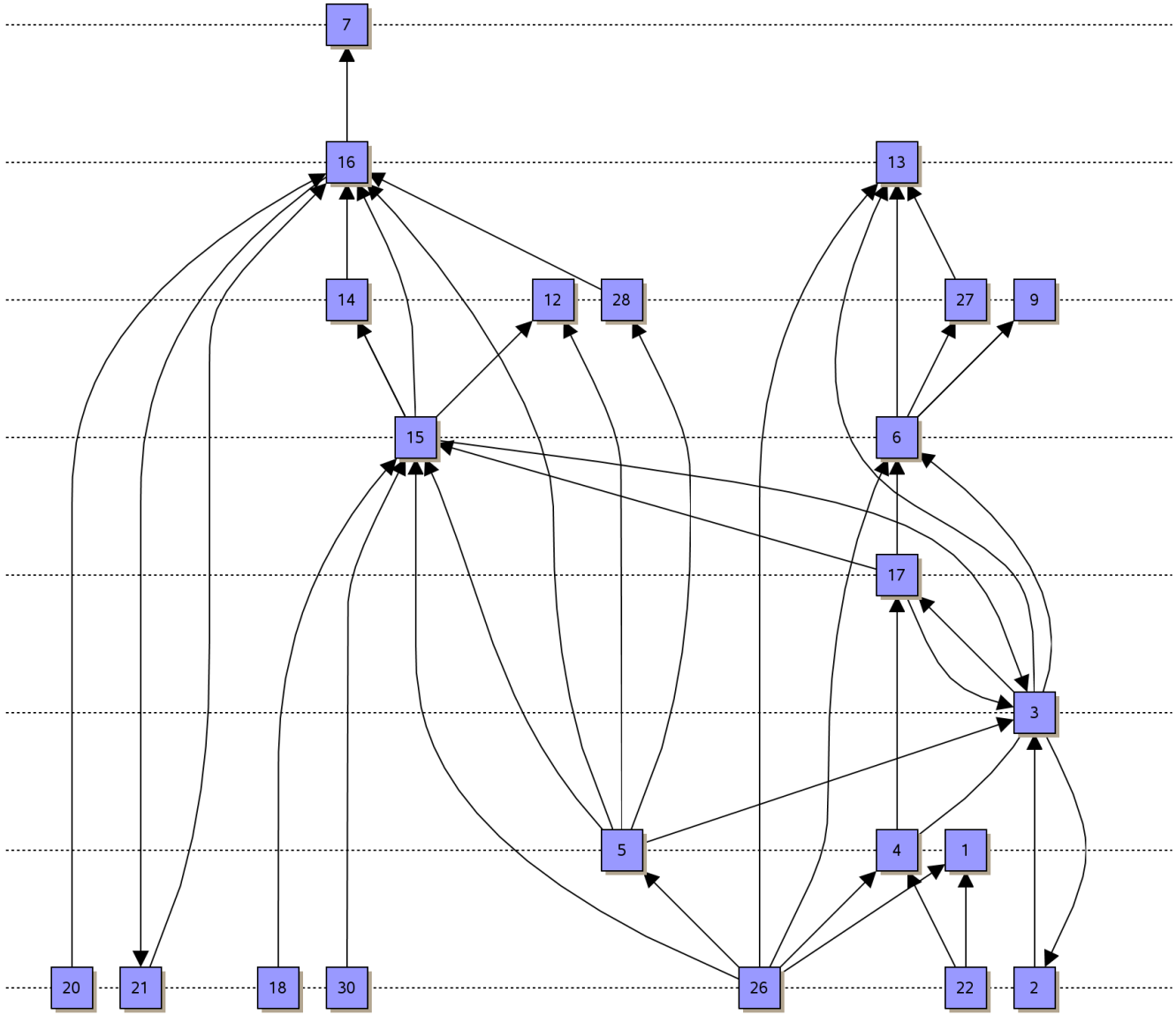
25.06.2009


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

E-Mail-Graph der Fakultät für Informatik



E-Mail-Graph der Fakultät für Informatik



Lagenlayouts

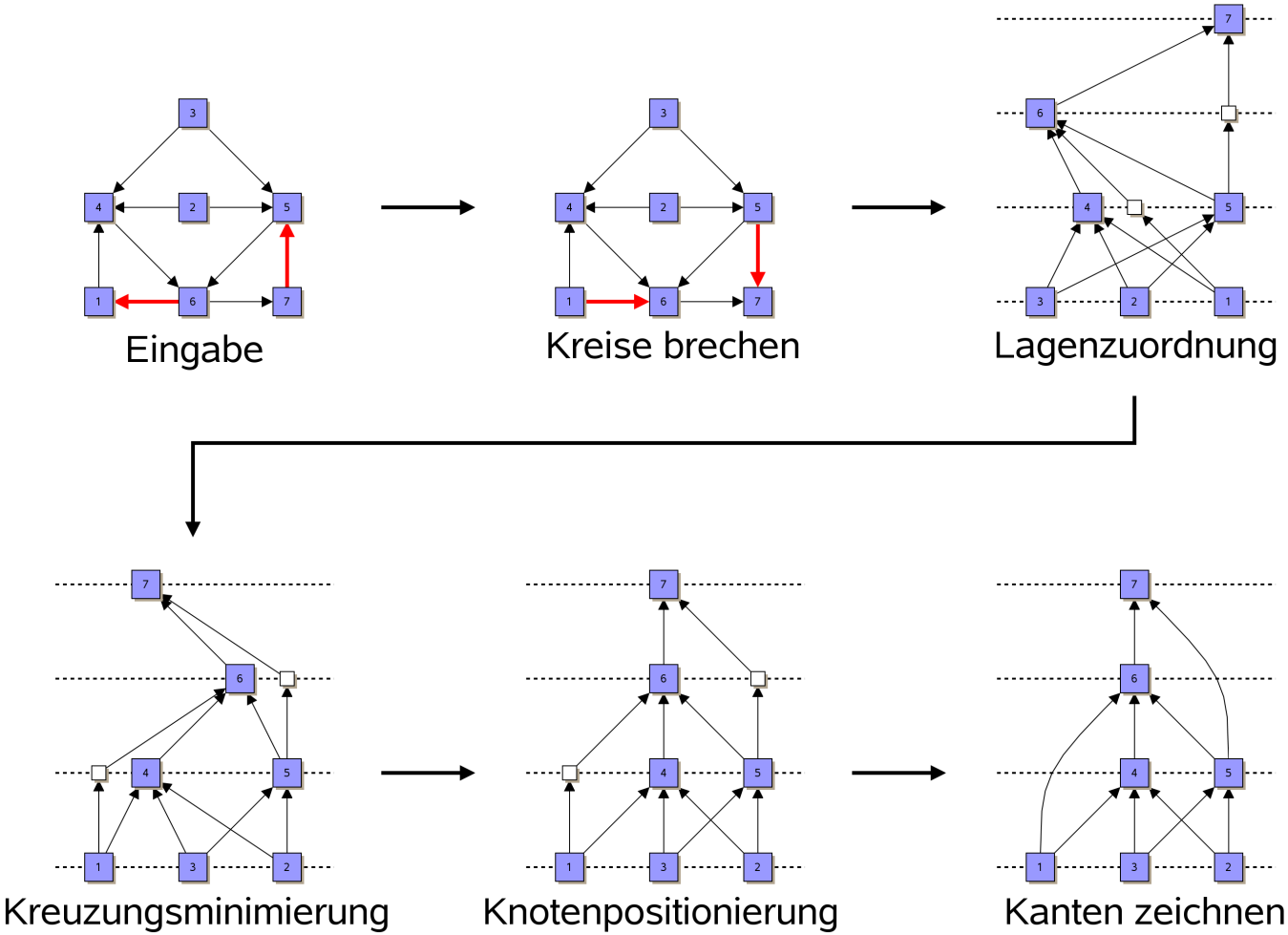
Problemstellung

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

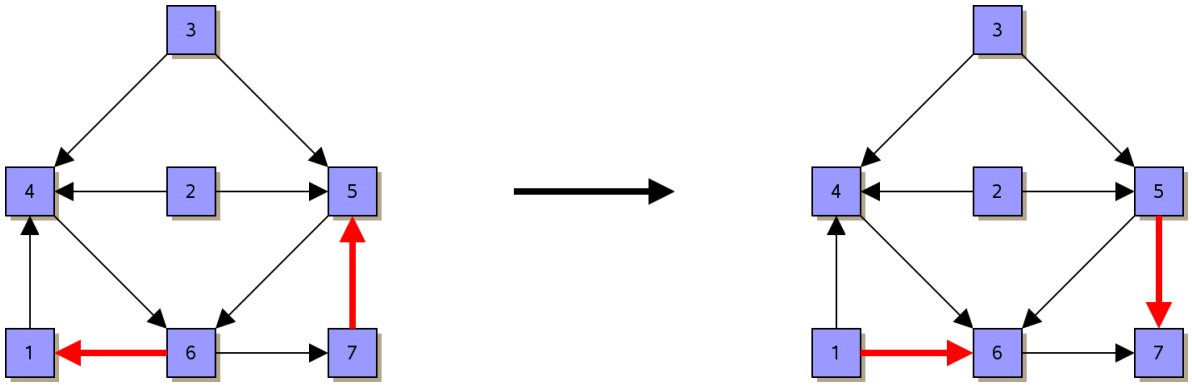
Desiderata

- ≫ Zuordnung der Knoten auf (wenige) horizontale Linien
- ≫ möglichst viele Kanten aufwärtsgerichtet
- ≫ möglichst wenige Kantenkreuzungen
- ≫ Kanten möglichst vertikal, geradlinig und kurz
- ≫ Knoten gleichmäßig verteilt
- ! Kriterien widersprechen sich

Klassisches Vorgehen (Sugiyama)



1. Schritt: Behandlung von Gerichteten Kreisen



Behandlung von Gerichteten Kreisen

Vorgehen

- Finde minimale Menge von Kanten A_f, die nicht aufwärts gezeichnet werden
- Entferne Kanten in A_f und füge Inversen 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) $A_f := A \setminus A'$

$\gg$ Laufzeit $\mathcal{O}(n + m)$

$\gg$ A' hat mindestens $1/2 \cdot |A|$ viele Kanten

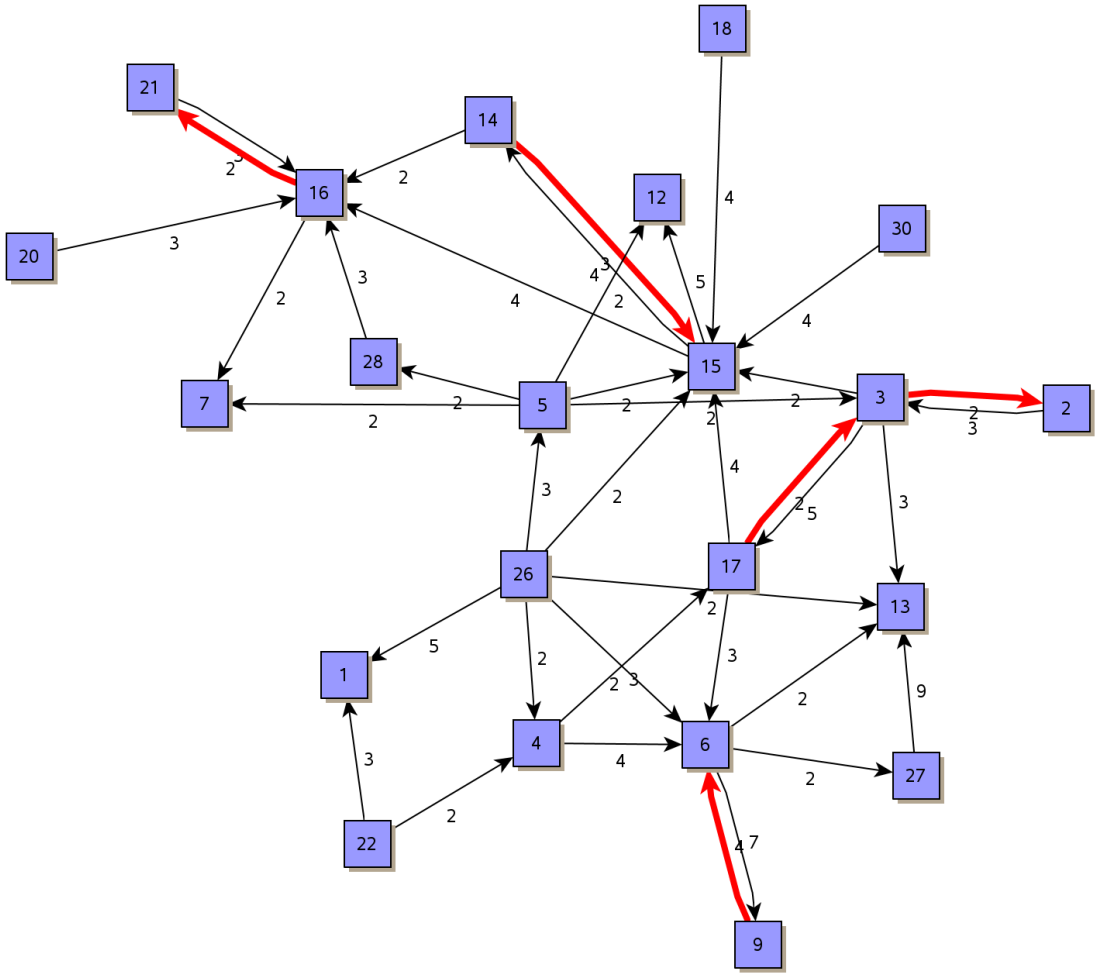
Behandlung von Gerichteten Kreisen

Verbesserte Greedy-Heuristik

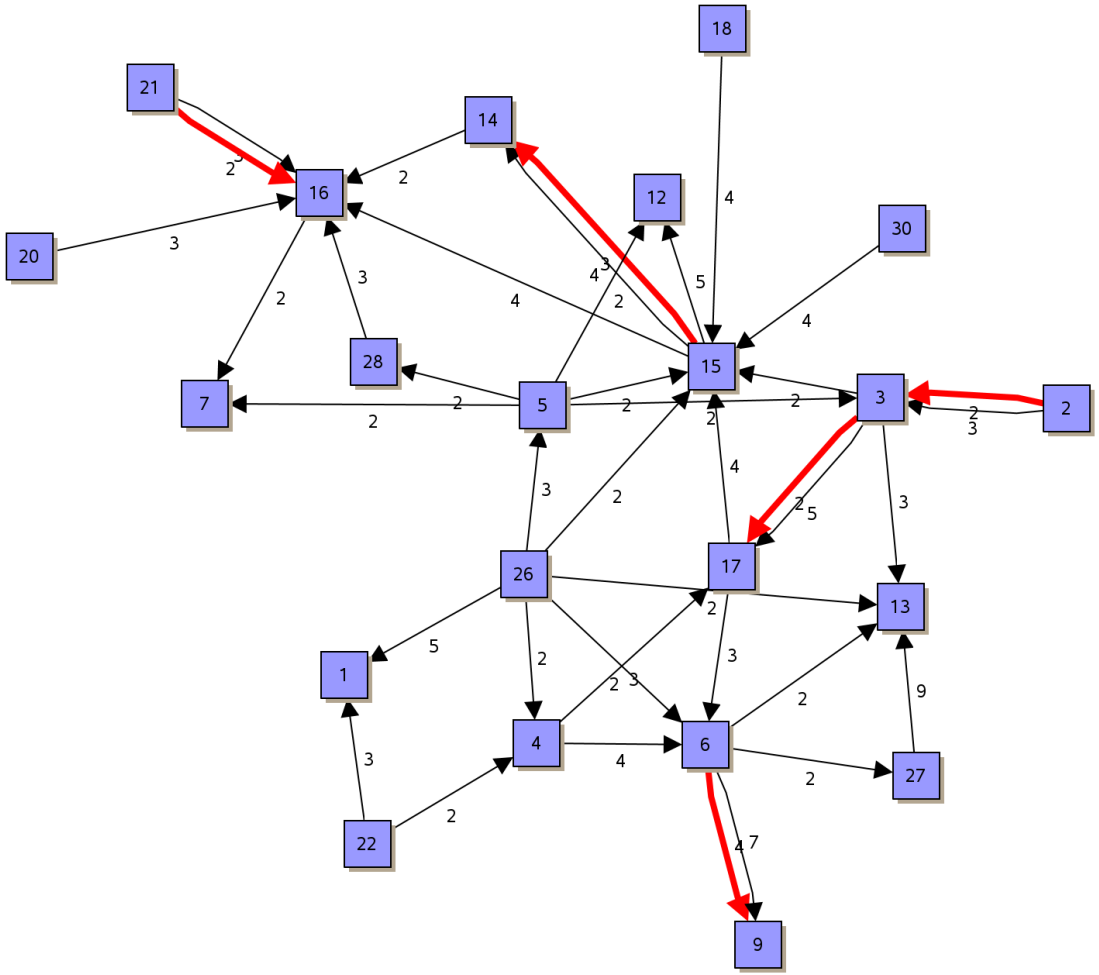
- Betrachte in jedem Schritt den Knoten, bei dem Differenz zwischen Eingangsgrad und Ausgangsgrad maximal ist

- Laufzeit $\mathcal{O}(n \log n + m)$
- A' hat mindestens $1/2|A| + 1/6|V|$ viele Kanten

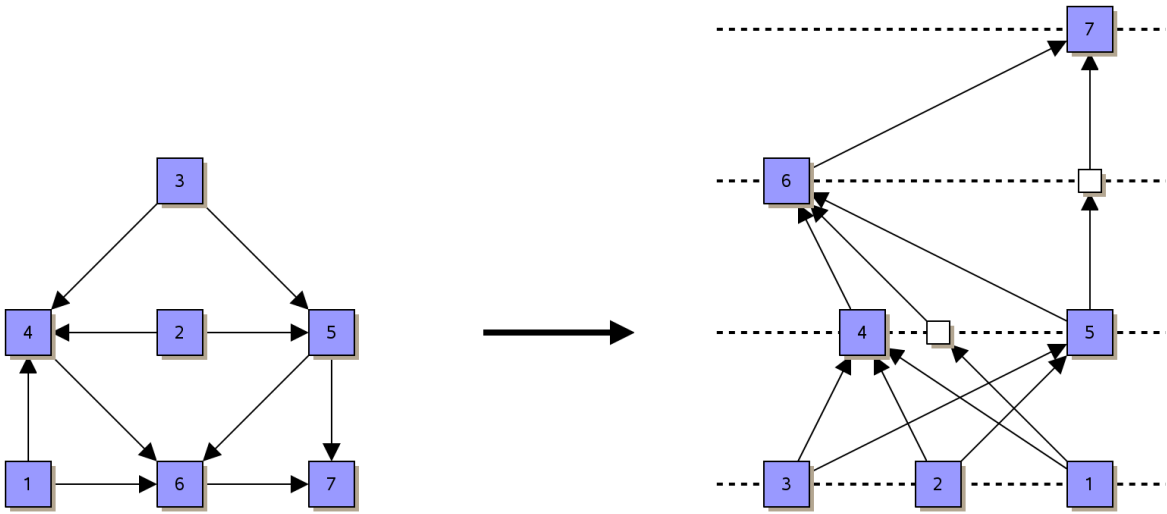
E-Mail-Graph der Fakultät für Informatik



E-Mail-Graph der Fakultät für Informatik



2. Schritt: Lagenzuordnung



Lagenzuordnung

Problemstellung

- Gegeben: azyklischer, gerichteter Graph $D = (V, A)$
- Finde zulässige Partition der Knotenmenge V in Lagen L_y, so dass für all $(u, v) \in A$ gilt $y(u) < y(v)$
- minimiere Gesamthöhe

Weitere Zielfunktion

- minimiere längste Kante
- minimiere Gesamtlänge der Kanten (wenige Dummy-Knoten)

Lagenzuordnung

- Ordne alle Quellen q Layer 1 zu, d.h. $y(q) = 0$
- sei $N^+(u)$ Menge der Knoten v mit $(v, u) \in A$
- setze

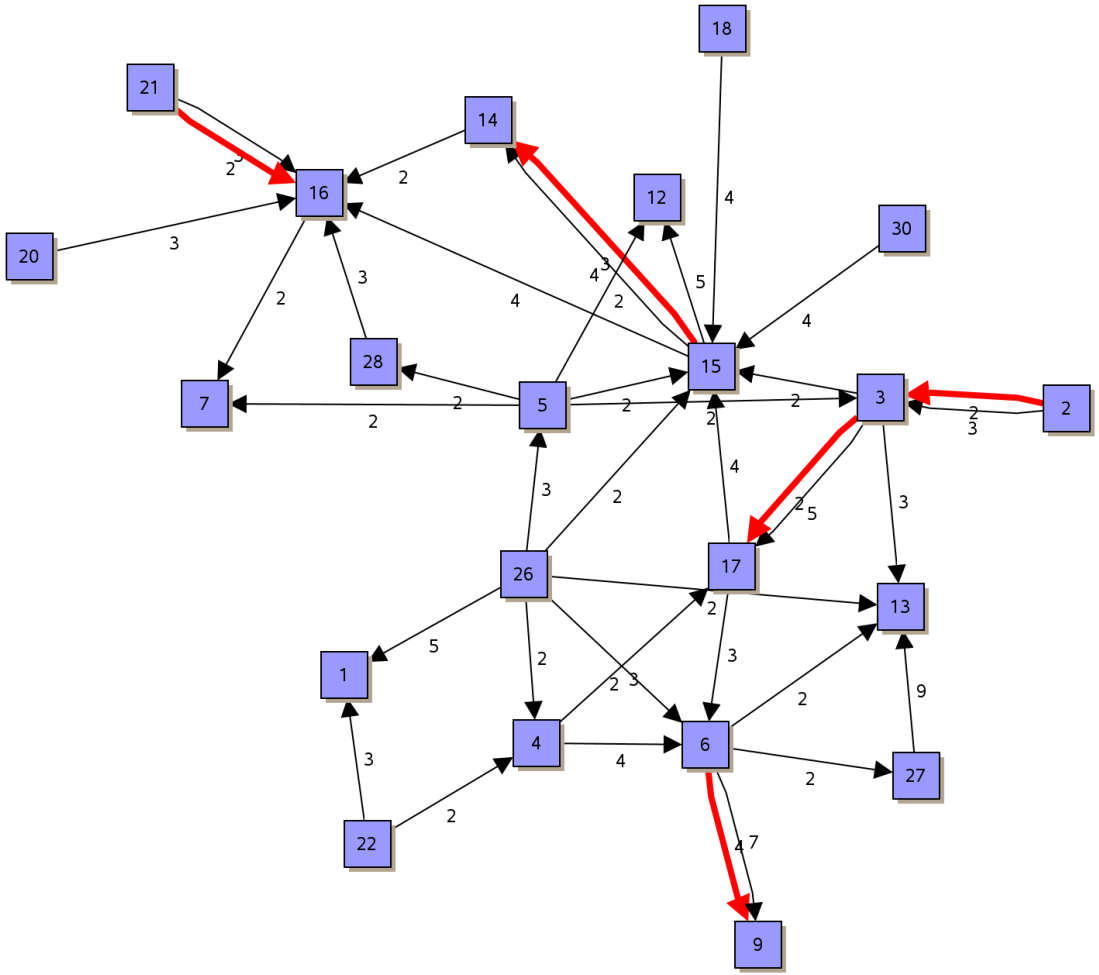
$$y(u) := \max\{i \mid v \in N^+(u) \text{ und } y(v) = i\} + 1$$

- d.h. y-Koordinate ist Länge des Längsten Wegs von einer Quelle zu v

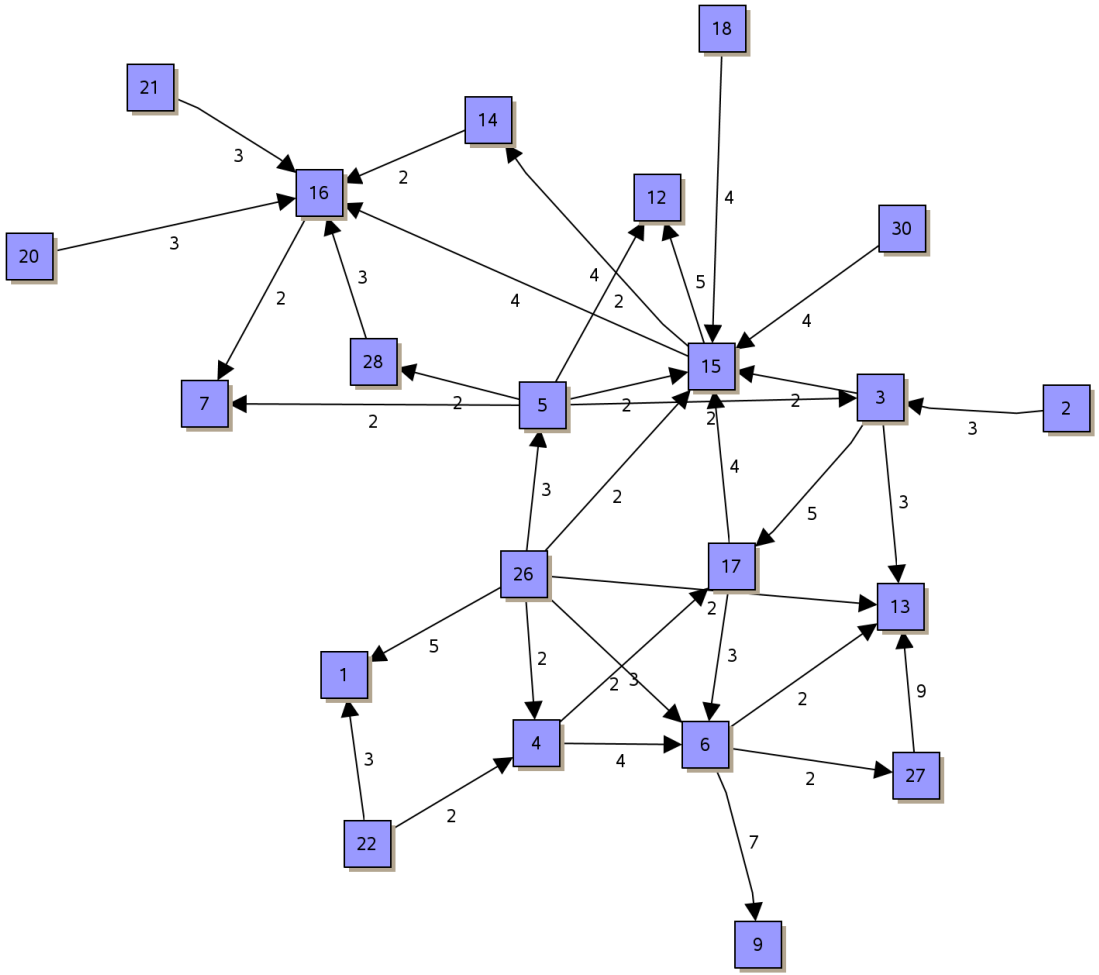
Implementation in Linearzeit

- Entferne transitive Reduktion (topologisches Sortieren)
- Simultane Breitensuche und Lagenzuordnung
- Füge transitive Reduktion wieder hinzu

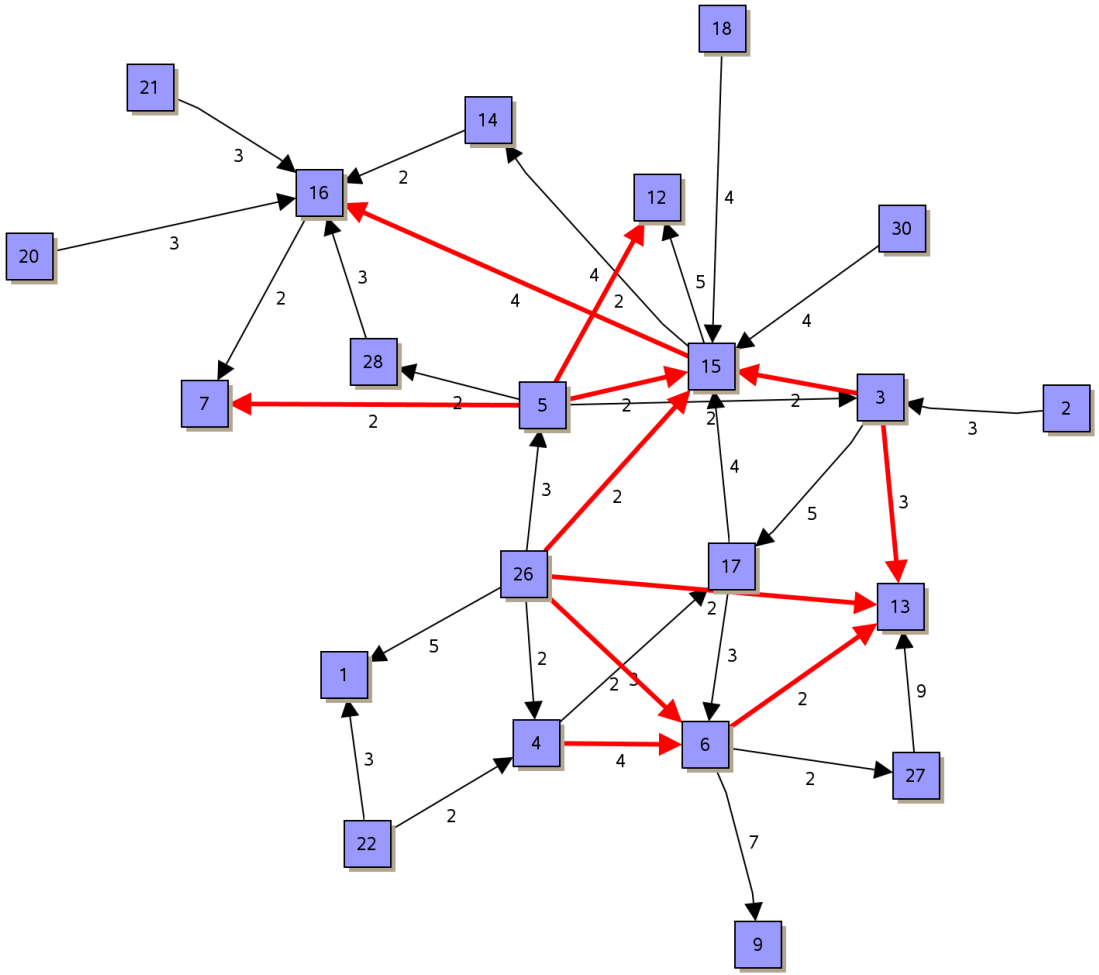
E-Mail-Graph der Fakultät für Informatik



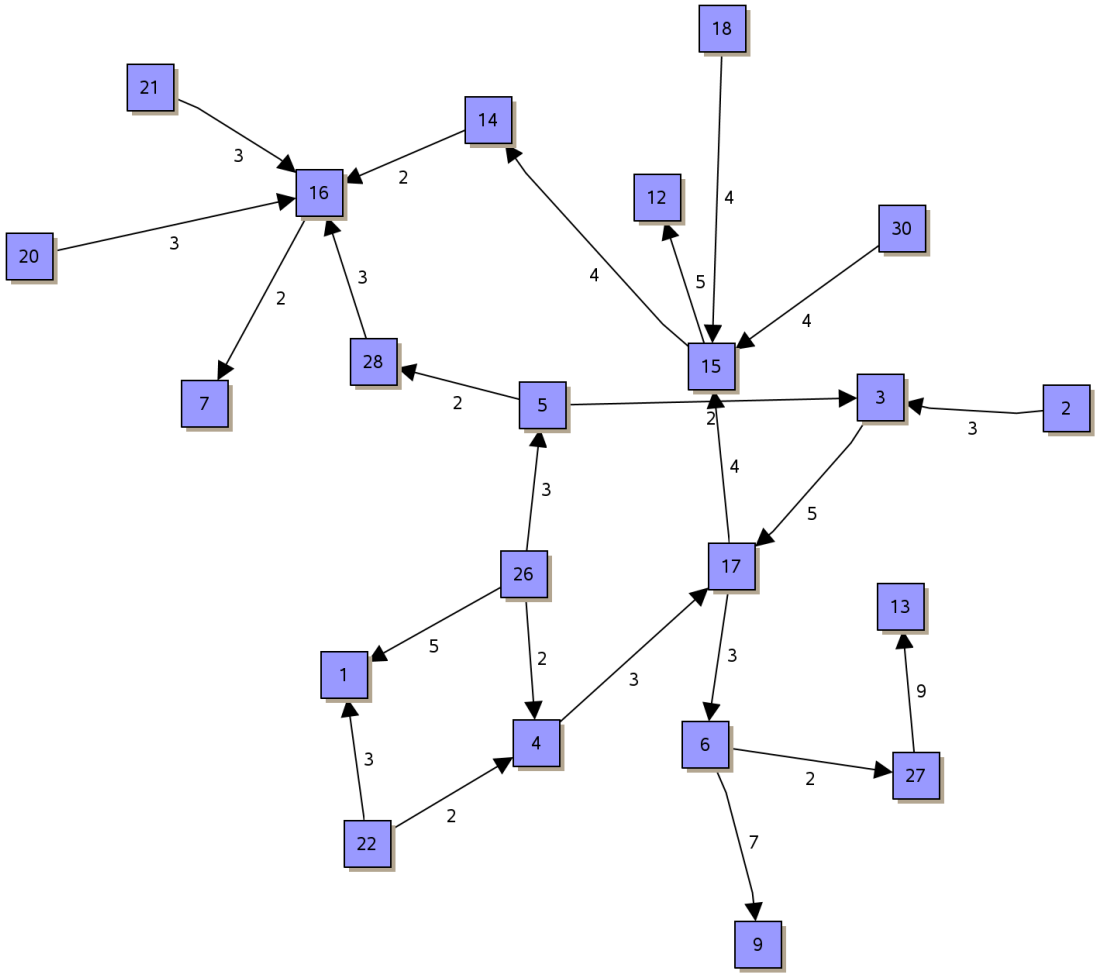
E-Mail-Graph der Fakultät für Informatik



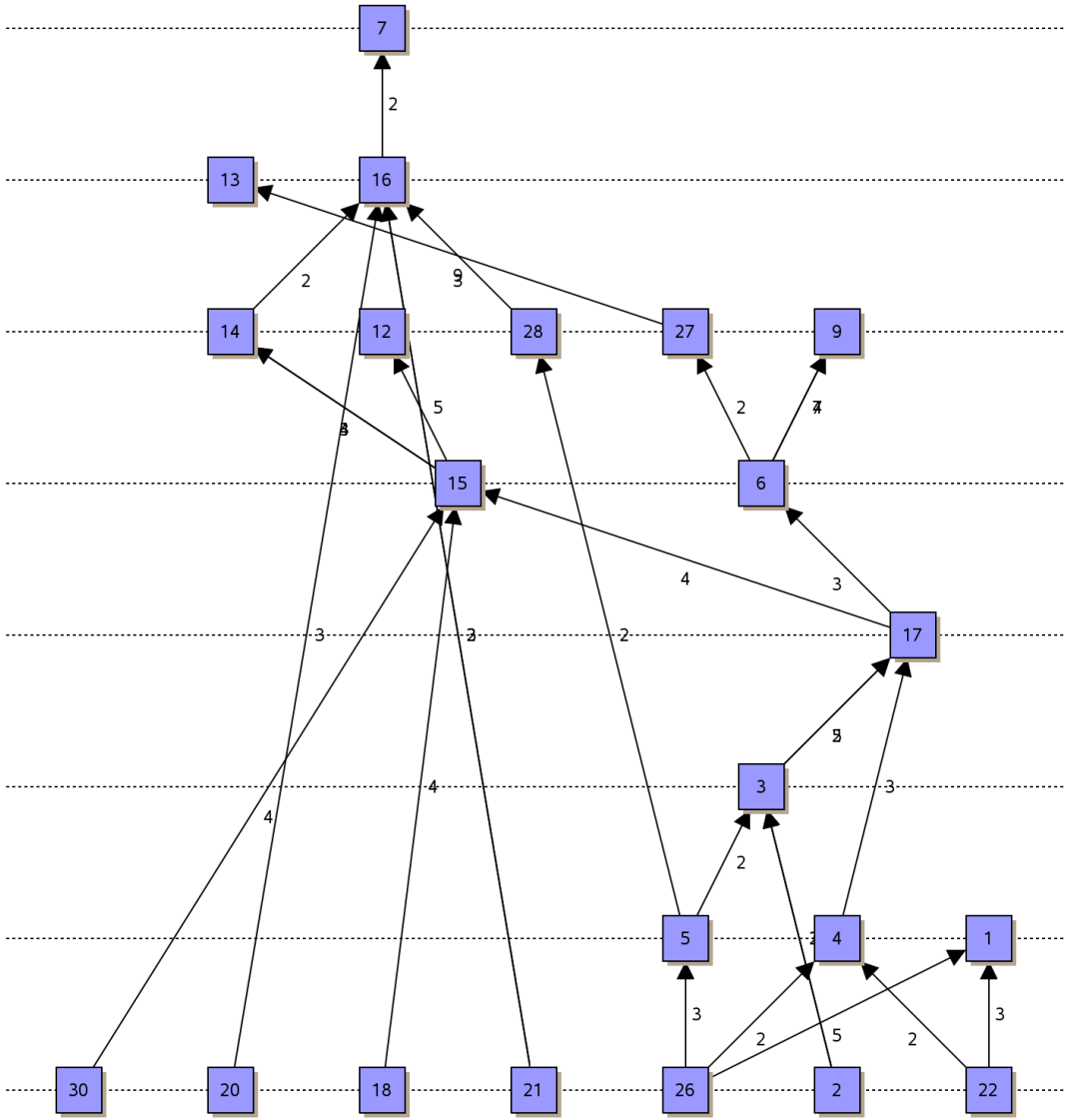
E-Mail-Graph der Fakultät für Informatik



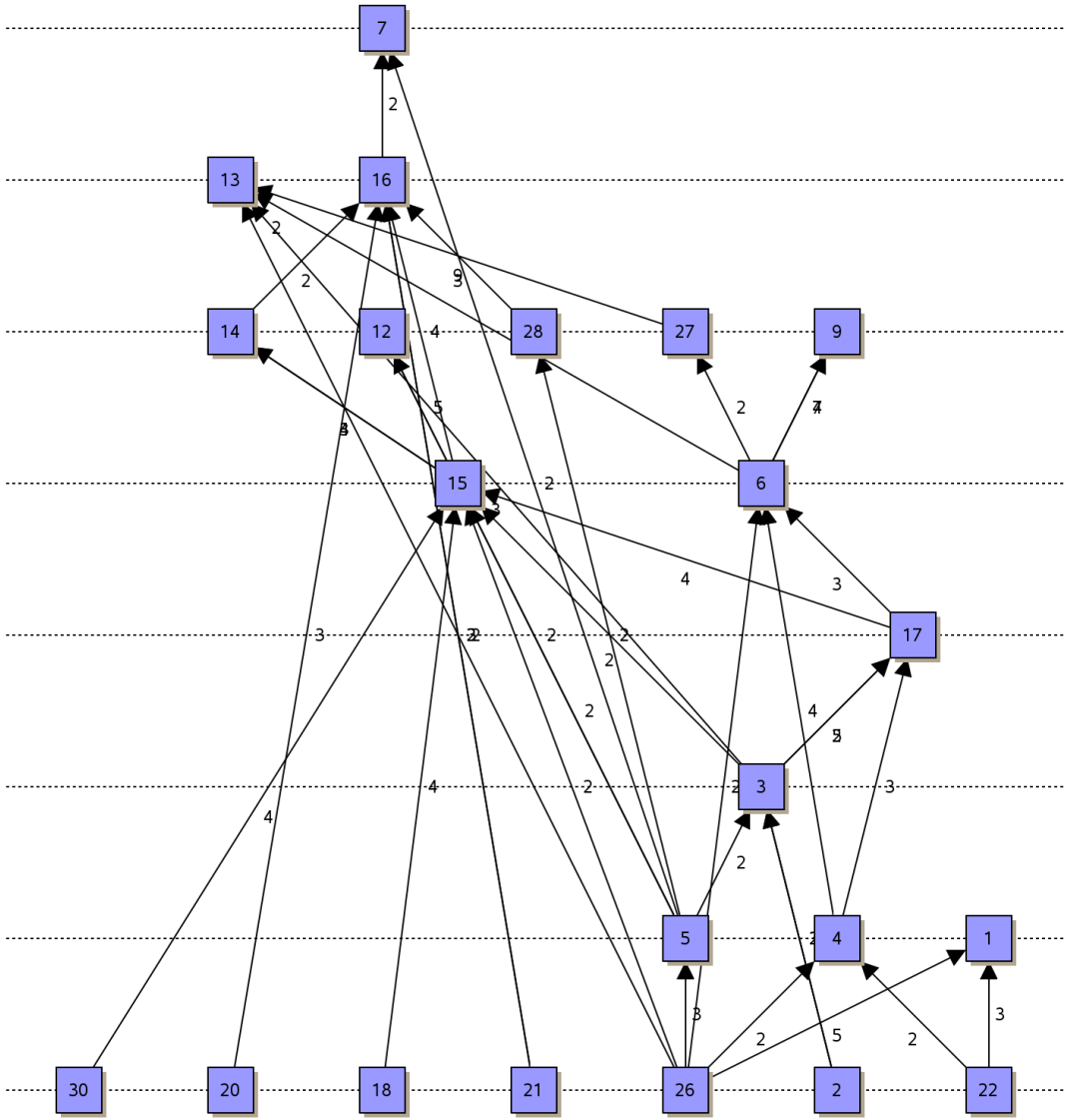
E-Mail-Graph der Fakultät für Informatik



E-Mail-Graph der Fakultät für Informatik



E-Mail-Graph der Fakultät für Informatik



Lagenzuordnung bei vorgegebener Breite

Problem Layer Assignment

- Gegeben: azyklischer, gerichteter Graph $D = (V, A)$ und Breite B
- Finde Partition der Knotenmenge mit minimaler Anzal von Lagen, so dass in jeder Lage höchstens B Elemente sind

- $\mathcal{NP}$-schwer da äquivalent zu Minimum Precedence Constrained Scheduling

Lagenzuordnung bei vorgegebener Breite

Problem Minimum Precedence Constrained Scheduling

- Gegeben: n Jobs mit Bearbeitungsdauer 1 und B Maschinen sowie partielle Ordnung $<$ auf den Jobs
- Finde Schedule, der $<$ berücksichtigt und minimale Bearbeitungsdauer hat

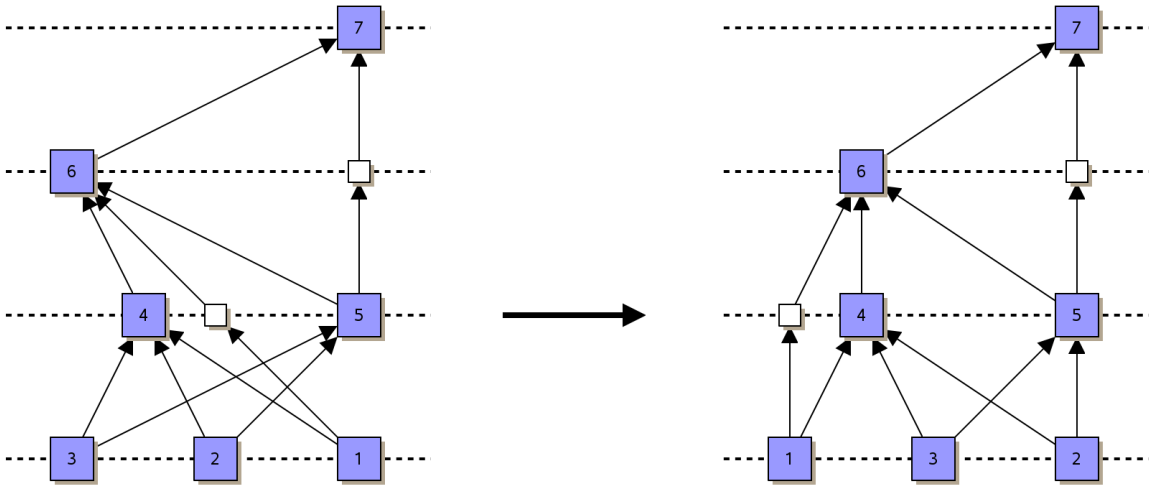
- $\mathcal{NP}$-schwer
- nicht $(4/3 - \varepsilon)$-approximierbar
- approximierbar mit Faktor $2 - 1/B$ bzw. $2 - 2/B$

Lagenzuordnung bei vorgegebener Breite

$(2 - 1/B)$-Approximation

- Verfahren wie bei List-Scheduling:
- Knoten sind in Liste L gespeichert (beliebige Reihenfolge, z.B. topologisch sortiert)
- betrachte Layer in aufsteigender Reihenfolge
- Knoten heißt verfügbar, falls keiner seiner Vorgänger mehr in L ist
- so lange aktuelles Layer nicht voll und verfügbarere Knoten in L existiert, lösche ersten verfügbaren Knoten aus L und ordne ihn aktuellem Layer zu

3. Schritt: Kreuzungsreduktion



Kreuzungsreduktion

Problemstellung

- Gegeben: Graph G, Knoten sind je einem Layer zugeordnet
- Gesucht: Umordnung der Knoten innerhalb der Layer, so dass die Anzahl der Kreuzungen minimiert wird

- Problem ist $\mathcal{NP}$-schwer, sogar für 2 Lagen
- Bipartite Crossing Number (Garey und Johnson, '83)
- kaum Ansätze, die echt über mehrere Layer optimieren

Iterative Kreuzungsreduktion

Iterative Kreuzungsreduktion

- füge Dummy-Knoten für Kanten mit Layerabstand > 1 ein
- betrachte jeweils benachbarte Layer nacheinander
- minimiere Layer L_{i+1} bei gegebener Ordnung der Knoten in Layer L_i

- Beobachtung: Kreuzungszahl hängt nur von der Permutation der Knoten auf den benachbarten Layern ab

Iterative Kreuzungsreduktion

(1) berechne zufällige Permutation für unterstes Layer

(2) betrachte iterativ jeweils benachbare Layer L_i und L_{i+1}

(3) minimiere Anzahl der Kreuzungen durch Umordnen der Knoten in L_{i+1} (L_i fest) $\rightsquigarrow$ Einseitige Kreuzungsminimierung

(4) wiederhole Schritte (2) und (3) in umgekehrter Richtung ausgehend von Layer L_{h-1}

(5) wiederhole Schritte (2)-(4) bis keine Verbesserung mehr erzielt wird

(6) wiederhole Schritte (1)-(4) mit unterschiedlichen initialen Permutationen

Einseitige Kreuzungsreduktion

Einseitige Kreuzungsminimierung

- Gegeben: Graph G mit Partition L_1, L_2 der Kanten und gegebener Ordnung (Permutation) π_1 der Knoten in L_1
- Finde Knotenordnung π_2 auf L_2, so dass die Anzahl der Kantenpaare, die sich kreuzen, minimiert wird

- Problem ist $\mathcal{NP}$-schwer (Eades und Whitesides, 1994)

Einseitige Kreuzungsreduktion

Heuristiken

- Baryzentrisch
- Median
- Greedy Switch
- uvm.

Exakt

- ILP

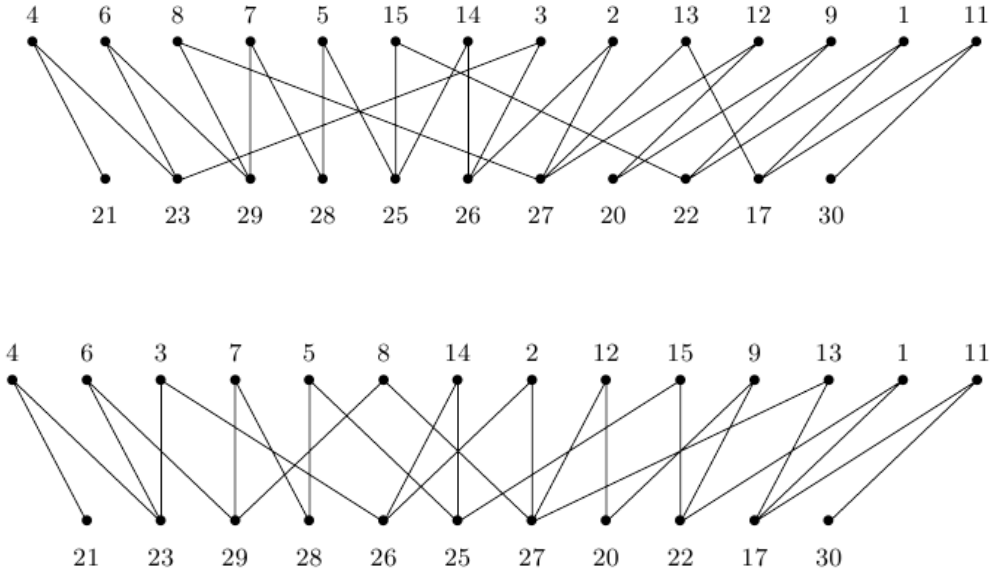


Abb. aus *Drawing Graphs, Kaufmann und Wagner*

Bemerkung: Heuristiken funktionieren besser für dichte Graphen

Einseitige Kreuzungsreduktion

Baryzenter-Heuristik (Sugiyama et al., 1981)

- Intuition: wenige Kreuzungen, wenn Knoten nah bei Nachbarn
- Baryzenter von u ist durchschnittliche x-Koordinate der Nachbarn $N(u)$ in Layer L_1

$$\text{bary}(u) = \frac{1}{\deg(u)} \sum_{v \in N(u)} x_1(v)$$

- setze $x_2(u) = \text{bary}(u)$
- bei gleichen Werten werden Knoten um einen Wert δ versetzt
- sortiere Knoten nach x-Koordinate, um Ordnung der Knoten zu erhalten
- Laufzeit $\mathcal{O}(|V_1| + |V_2| \log |V_2|)$

Einseitige Kreuzungsreduktion

Baryzenter-Heuristik (Sugiyama et al., 1981)

- geringer Implementationsaufwand
- schnell
- relativ gute Ergebnisse
- optimal, falls keine Kreuzung benötigt wird
- $\mathcal{O}(\sqrt{n})$-Approximation
- es muss keine Kreuzungsmatrix berechnet werden

Einseitige Kreuzungsreduktion

Median-Heuristik (Eades und Wormald, 1994)

- x-Koordinate von u wird auf Median der x-Koordinaten der Nachbarn von u in L_1 gesetzt
- seien $v_1, \ldots, v_k$ Nachbarn von u mit $\pi_1(v_1) < \pi_1(v_2) < \cdots < \pi_1(v_k)$

$$\text{med}(u) = \pi_1(v_{\lceil k/2 \rceil})$$

- $\text{med}(u) = 0$ falls $N(u) = \emptyset$
- verschiebe Knoten um δ geeignet, falls $\text{med}(u) = \text{med}(v)$

Einseitige Kreuzungsreduktion

Median-Heuristik (Eades und Wormald, 1994)

- geringer Implementationsaufwand
- schnell
- relativ gute Ergebnisse
- es muss keine Kreuzungsmatrix berechnet werden
- Faktor-3-Approximation

Einseitige Kreuzungsreduktion

Greedy-Switch

- vertausche iterativ jeweils benachbarte Knoten, falls dadurch weniger Kreuzungen induziert werden
- Laufzeit $\mathcal{O}(|V_2|)$ pro Iteration und maximal $|V_2|$ Iterationen
- als Post-Processing für andere Heuristiken

Optimale Einseitige Kreuzungsreduktion

ILP-Modellierung (Jünger und Mutzel, 1997)

- Konstante c_{ij} Anzahl von Kreuzungen zwischen Kanten, die zu i oder j inzident sind, falls $\pi_2(v_i) < \pi_2(v_j)$
- Variable $1 \leq i < j \leq n_2$

$$x_{ij} = \begin{cases} 1 & \text{falls } \pi_2(v_i) < \pi_2(v_j) \\ 0 & \text{sonst} \end{cases}$$

- Anzahl Kreuzungen für feste Permutation π_2

$$\text{cross}(\pi_2) = \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} (c_{ij} - c_{ji}) x_{ij} + \underbrace{\sum_{j=i+1}^{n_2} c_{ji}}_{\text{konstant}}$$

Optimale Einseitige Kreuzungsreduktion

ILP-Modellierung (Jünger und Mutzel, 1997)

- Minimiere Anzahl der Kreuzungen:

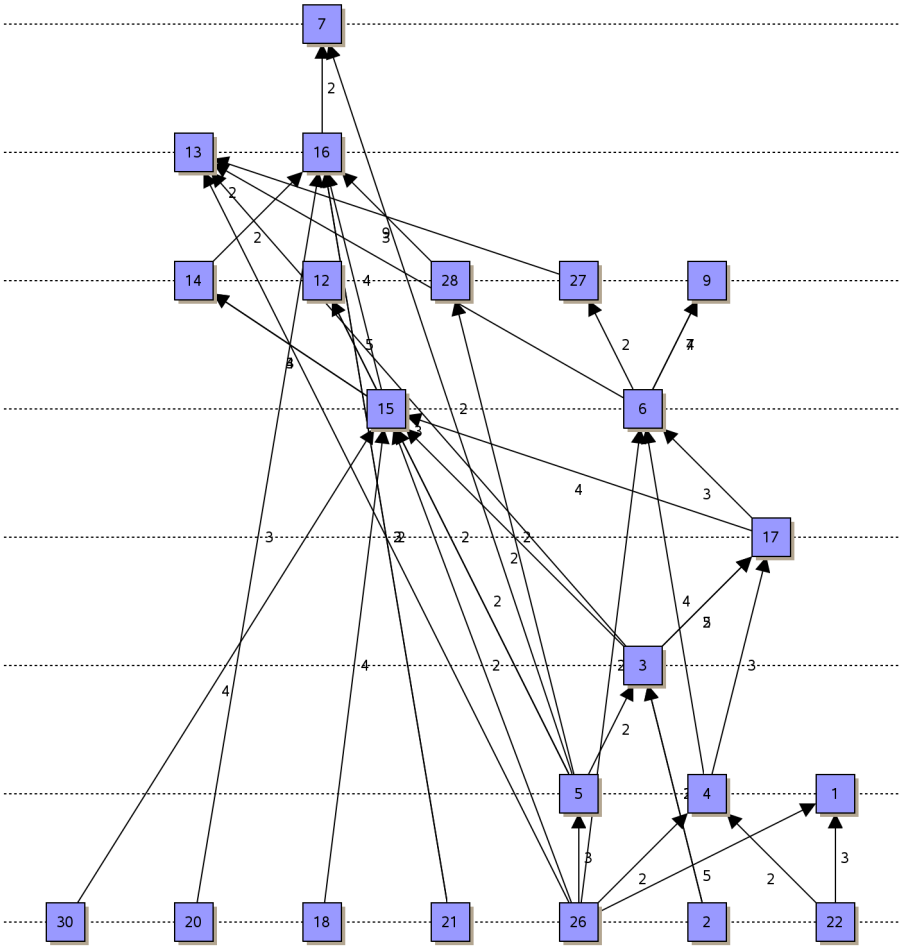
$$\min \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} (c_{ij} - c_{ji}) x_{ij}$$

- Nebenbedingungen:

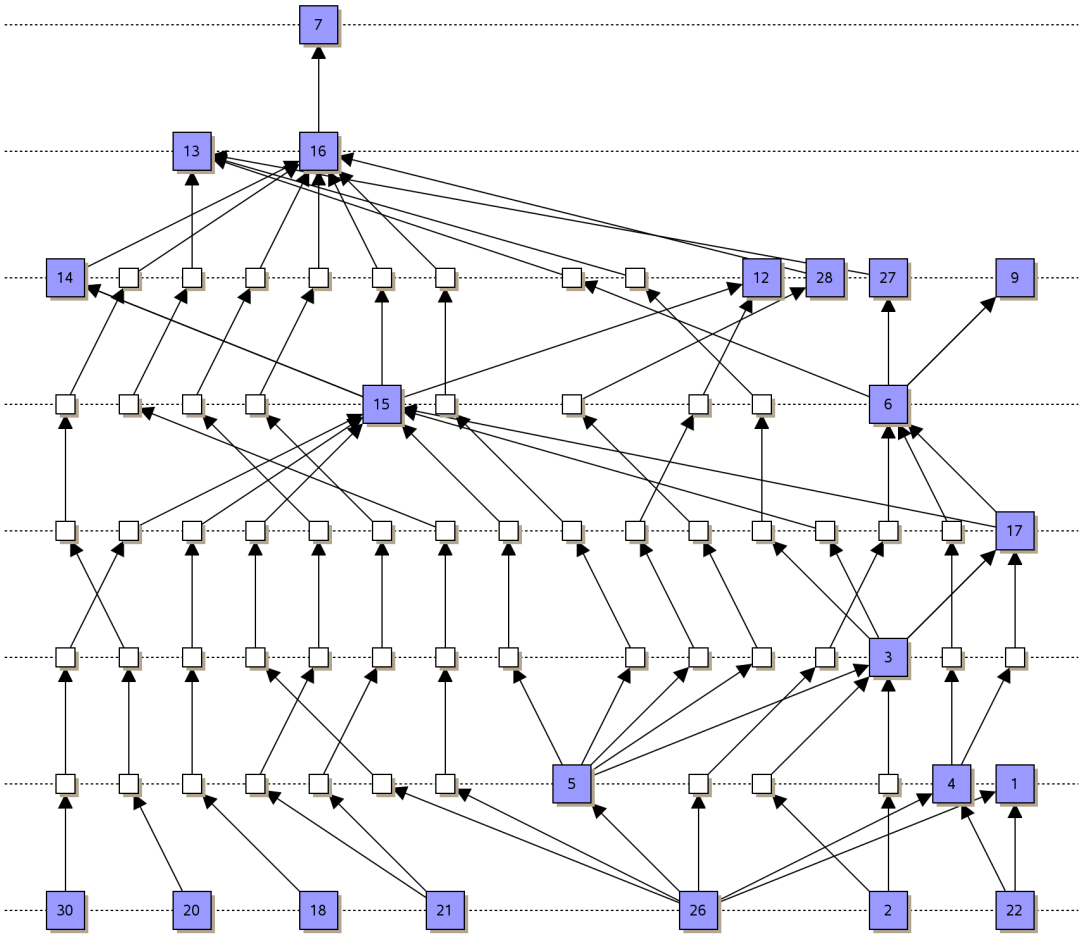
$$0 \leq x_{ij} + x_{jk} - x_{ik} \leq 1 \qquad 1 \leq i < j < k \leq n_2$$

- Implementierung mit Branch-and-Cut bei wenigen Knoten pro Layer relativ schnell

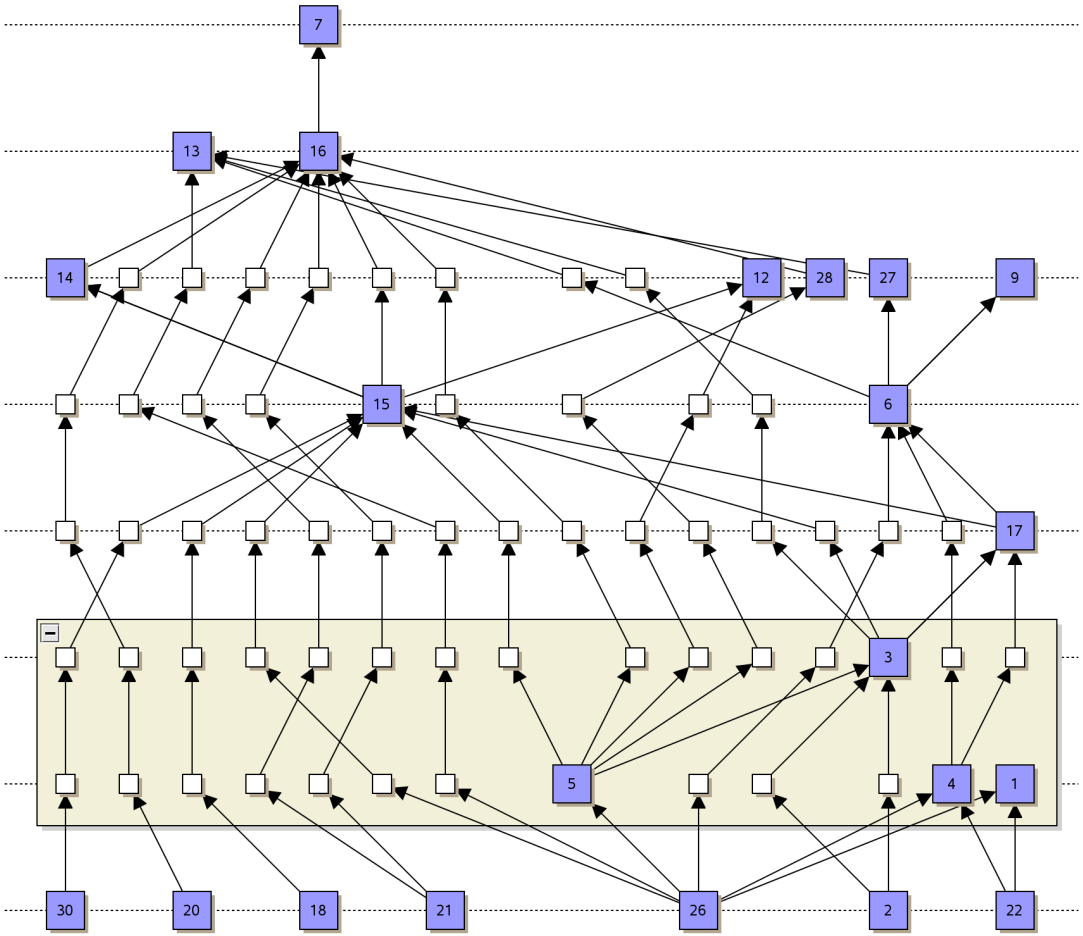
E-Mail-Graph der Fakultät für Informatik



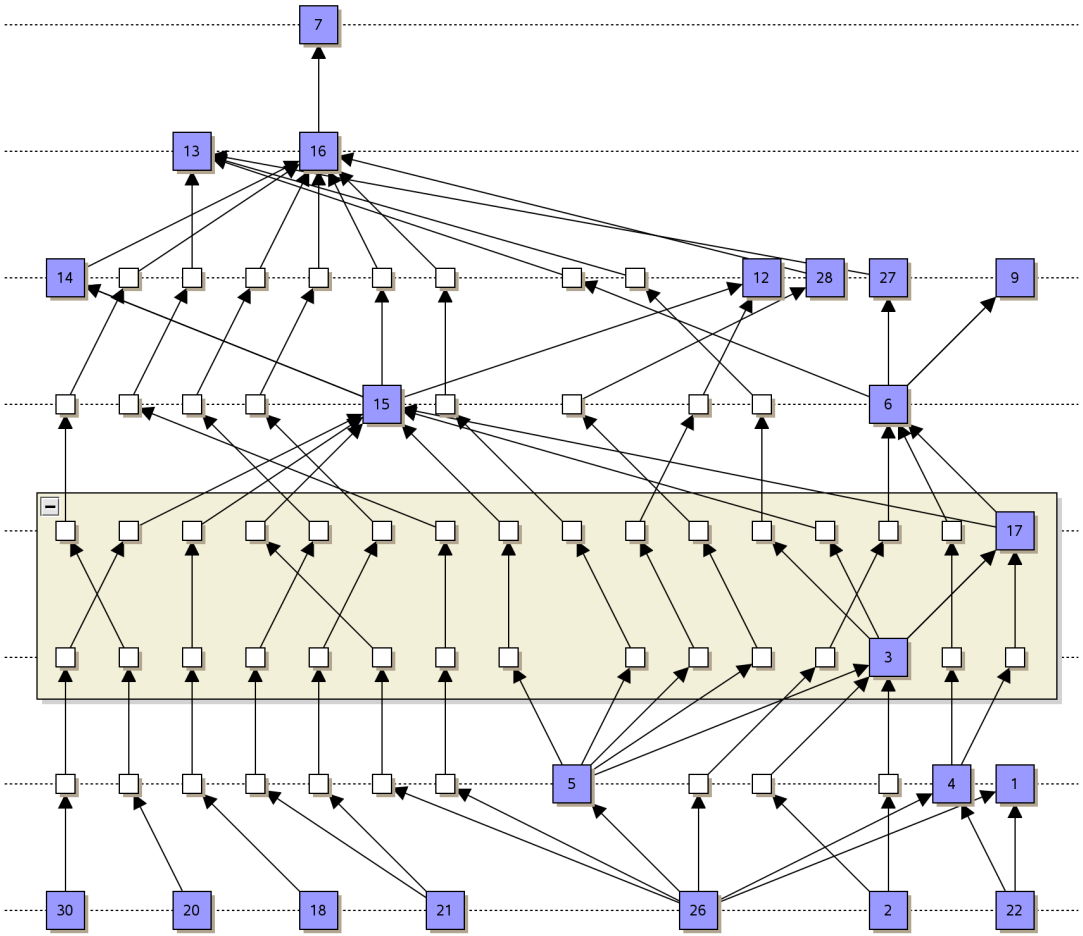
E-Mail-Graph der Fakultät für Informatik



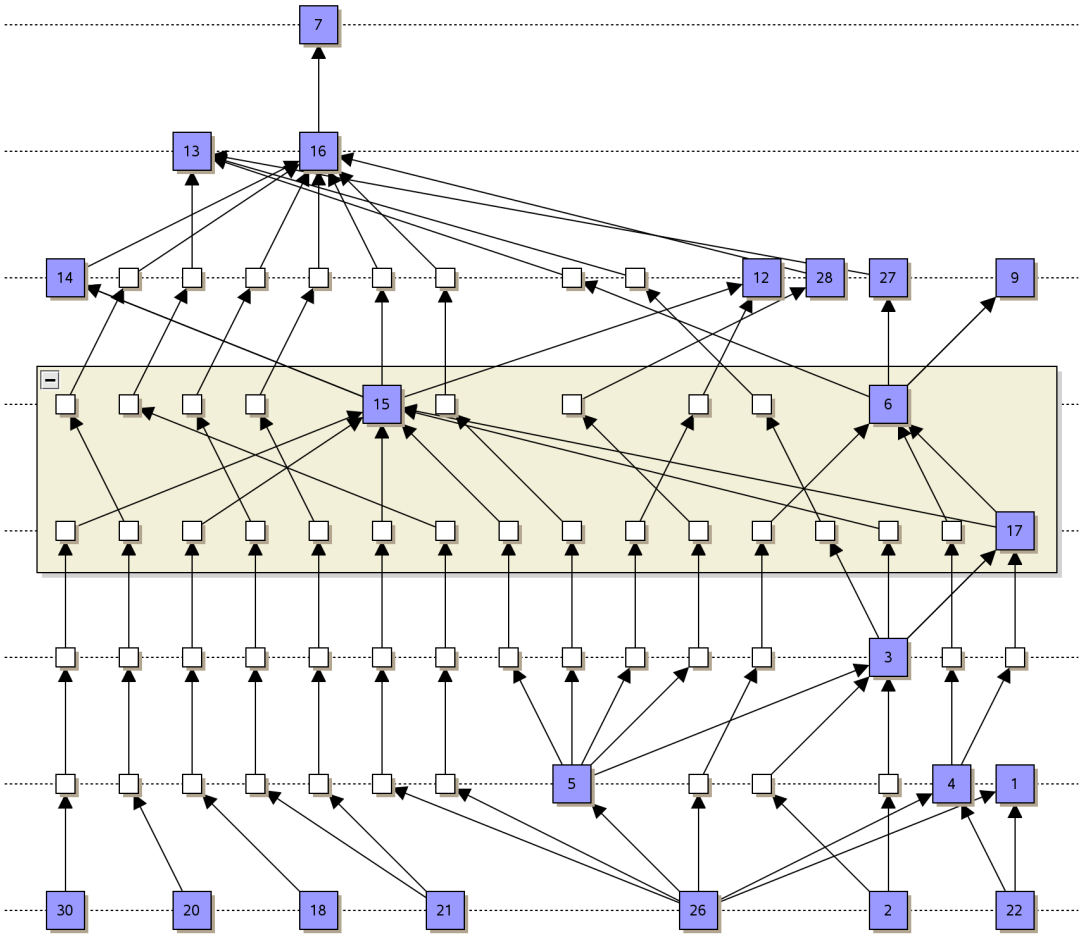
E-Mail-Graph der Fakultät für Informatik



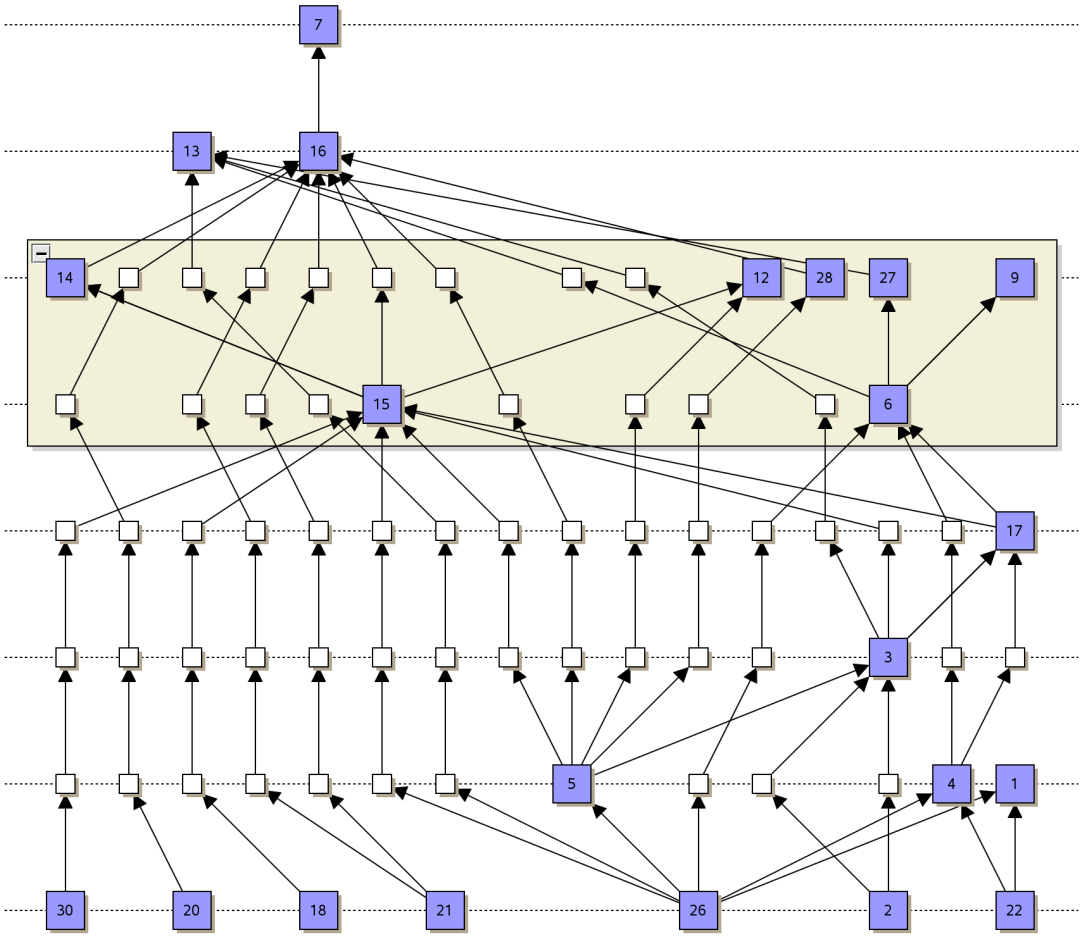
E-Mail-Graph der Fakultät für Informatik



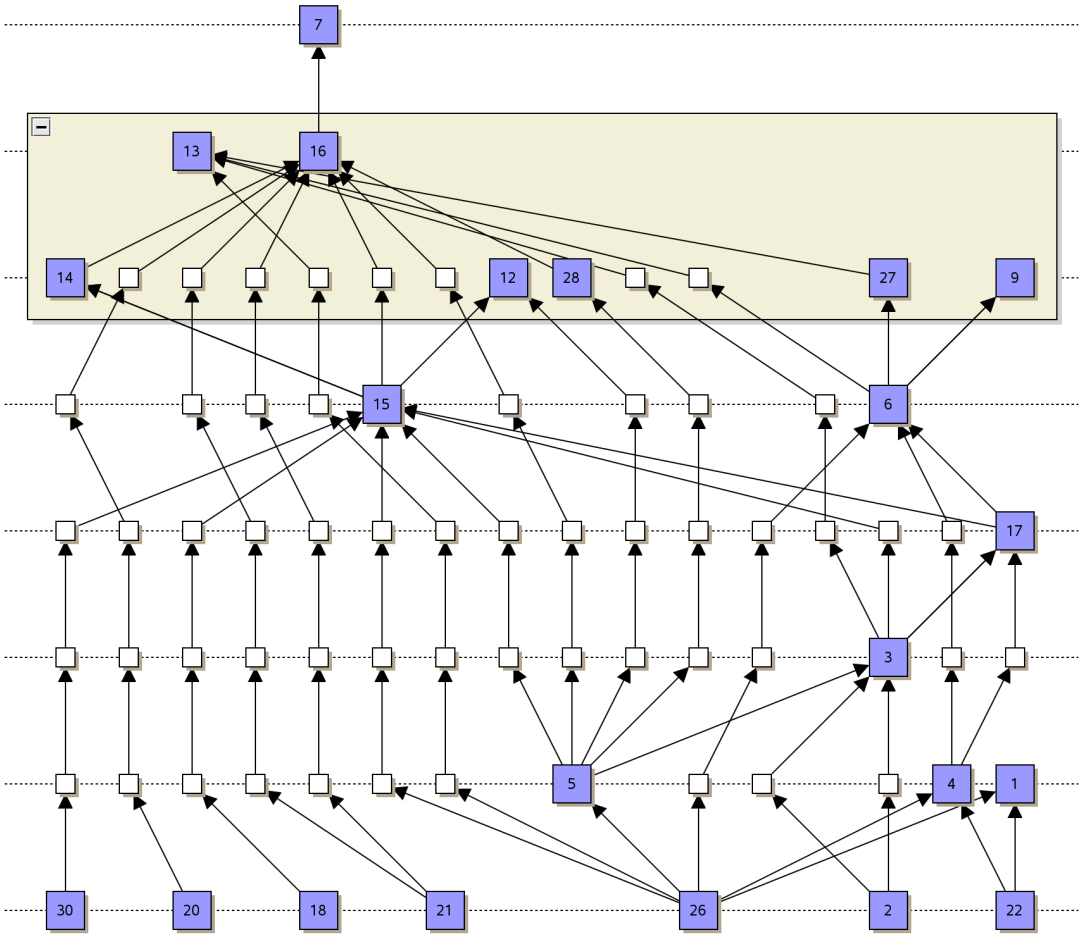
E-Mail-Graph der Fakultät für Informatik



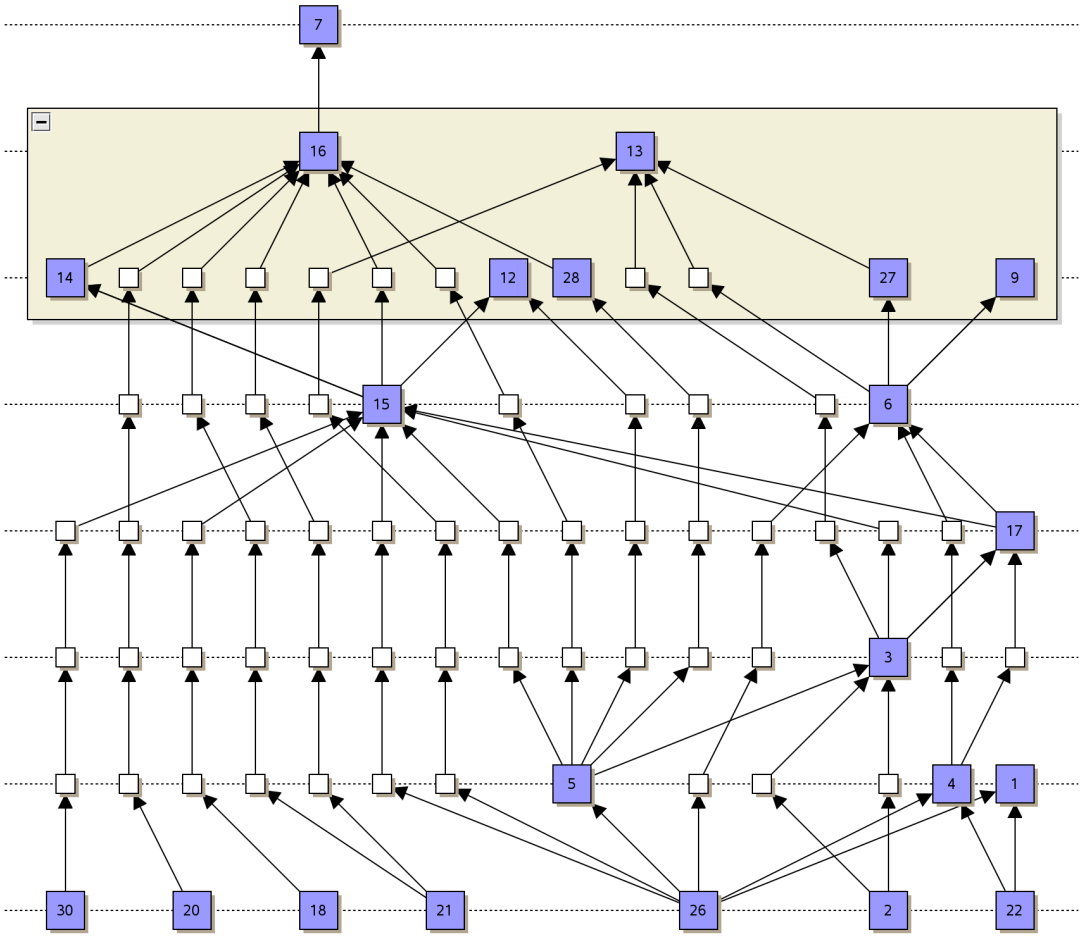
E-Mail-Graph der Fakultät für Informatik



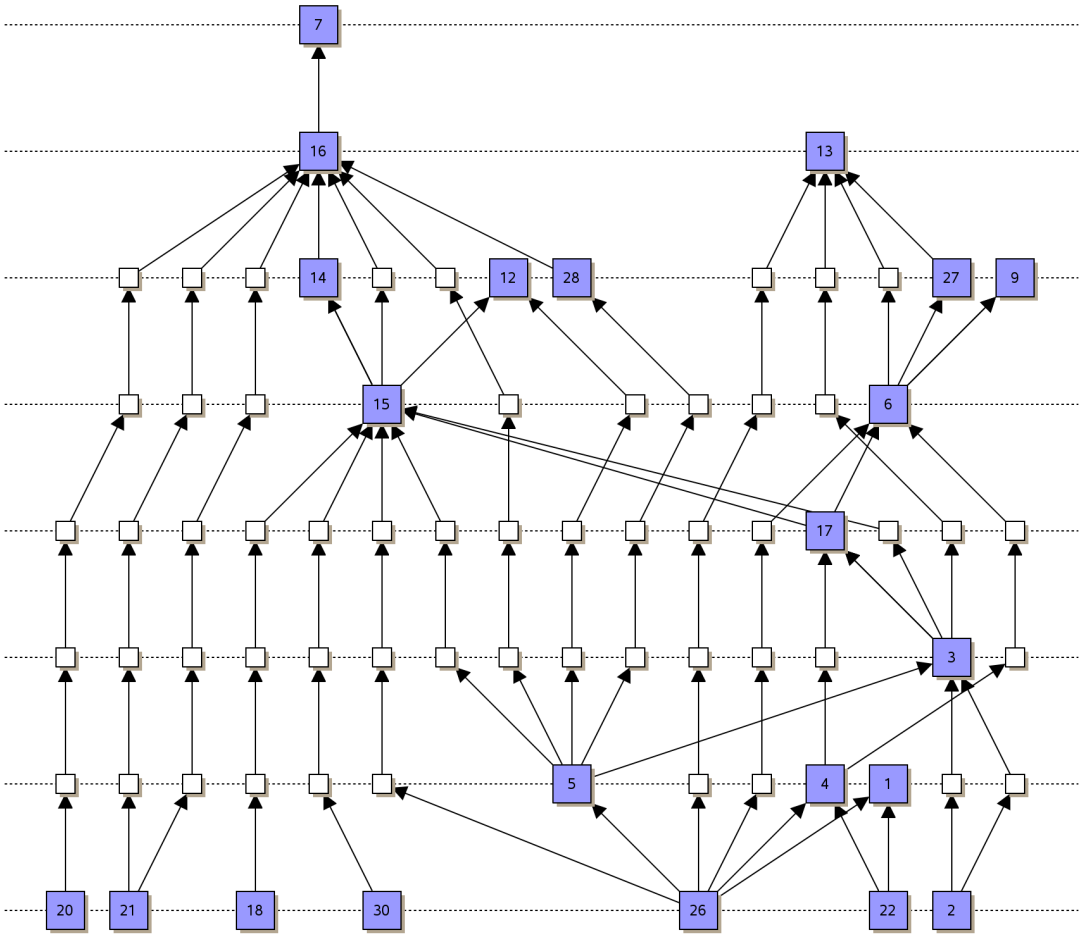
E-Mail-Graph der Fakultät für Informatik



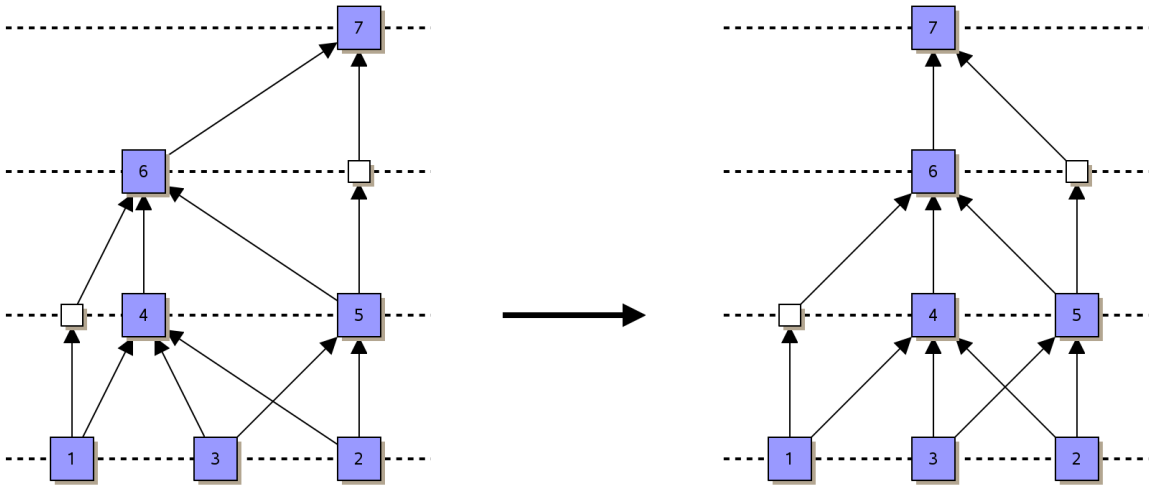
E-Mail-Graph der Fakultät für Informatik



E-Mail-Graph der Fakultät für Informatik



4. Schritt: Knotenpositionierung



Knotenpositionierung

Ziel

≫ minimiere Abweichung der Kanten-Pfade von gerader Linie

Exakt

≫ Quadratisches Programm

Heuristisch

≫ iterative Heuristiken

Knotenpositionierung

Quadratisches Programm

- Betrachte Kanten-Pfad $p_e = (v_1, \ldots, v_k)$ zu Kante e und Dummy-Knoten v_i
- x-Koordinate von v_i bei gerader Kante (gleicher Layerabstand):

$$\overline{x(v_i)} = \frac{i-1}{k-1}\left(x(v_k) - x(v_1)\right)$$

- definiere Abweichung von gerader Linie

$$\mathrm{dev}(p_e) := \sum_{i=2}^{k-1} \left(x(v_i) - \overline{x(v_i)}\right)^2$$

Knotenpositionierung

Quadratisches Programm

- Zielfunktion:

$$\min \sum_{e \in E} \mathrm{dev}(p_e)$$

- Nebenbedingungen: für alle Knoten v und alle Knoten w im gleichen Layer mit w rechts von v

$$x(w) - x(v) \geq \rho(w, v)$$

- $\rho(w, v)$ ist minimaler horizontaler Abstand zwischen den Knoten
- Problem: quadratisches Programm und potentiell exponentielle Breite

Knotenpositionierung

Iterative Heuristik

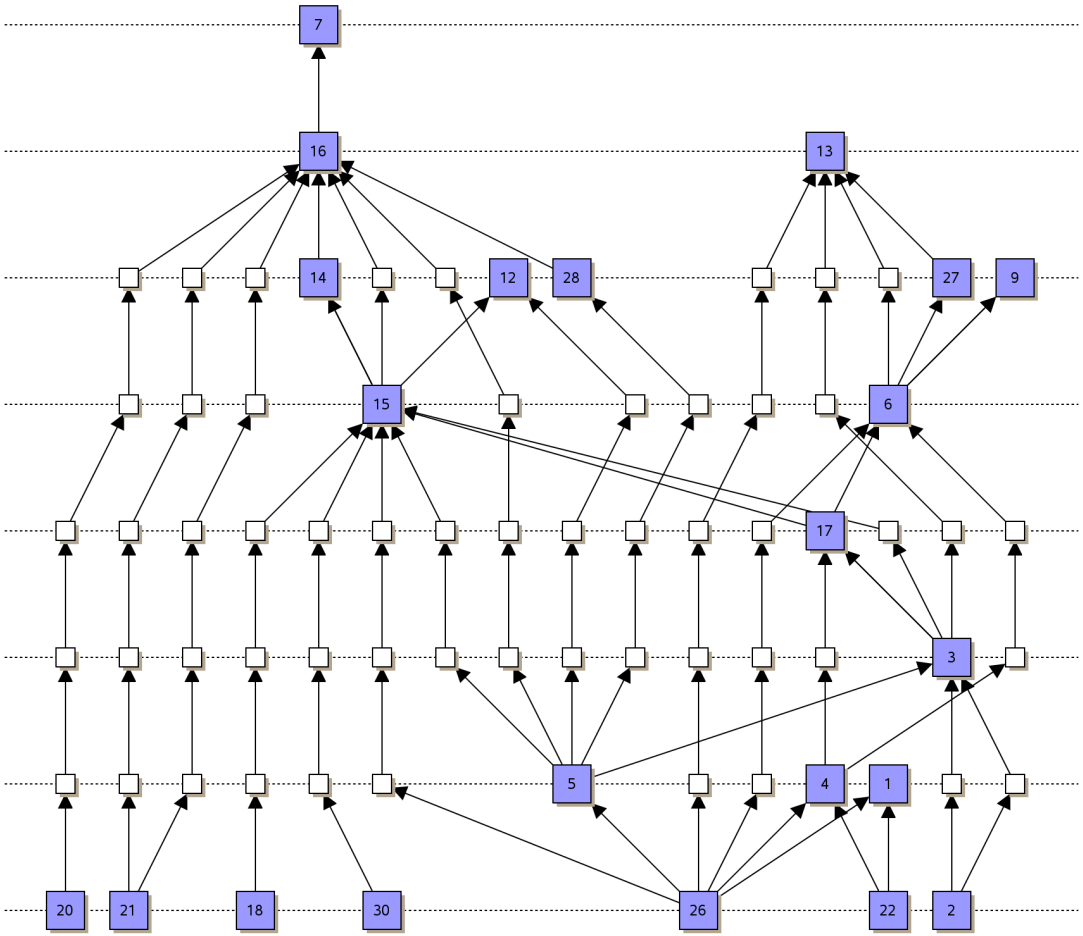
- ≫ berechne Initial-Layout
- ≫ führe die folgenden Schritte so lange aus bis Abbruchbedingung erfüllt ist

(1) Positioniere die Knoten

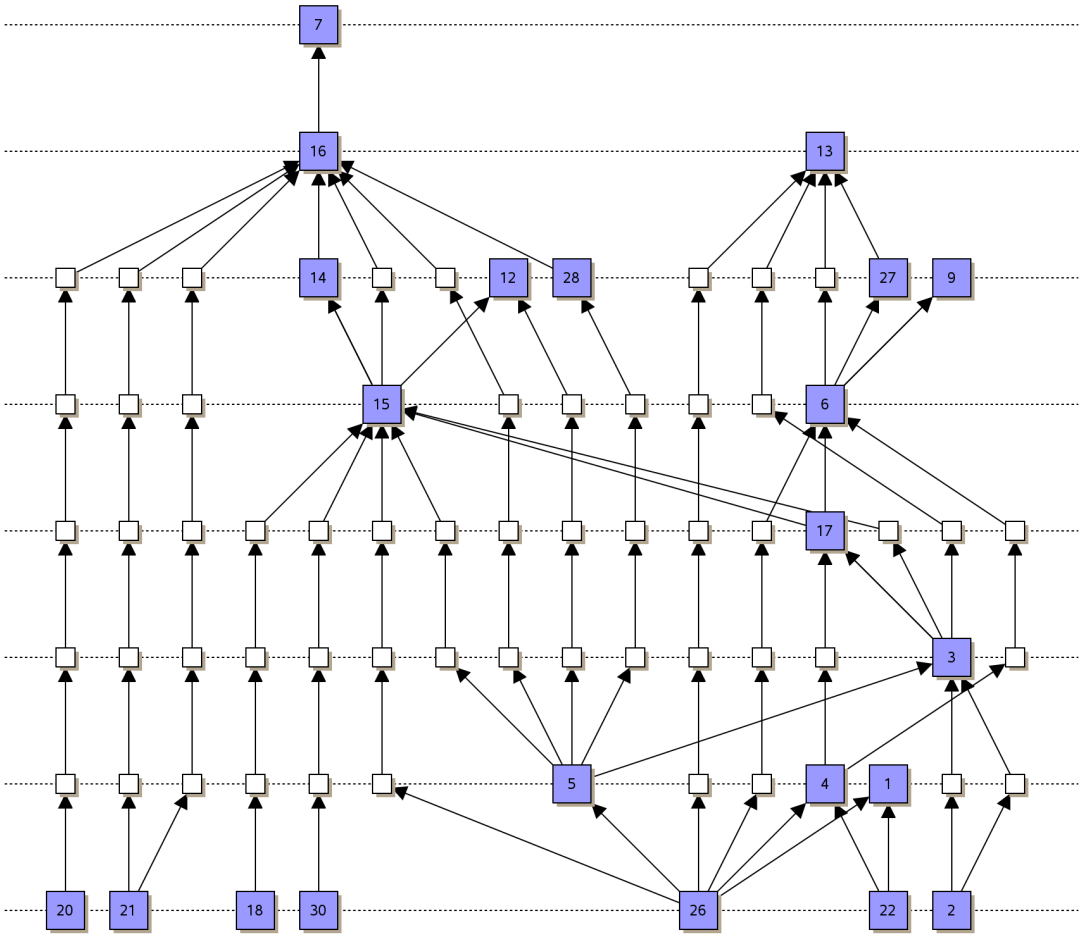
(2) Ziehe Kanten gerade

(3) Kompaktifiziere Layout in x-Richtung

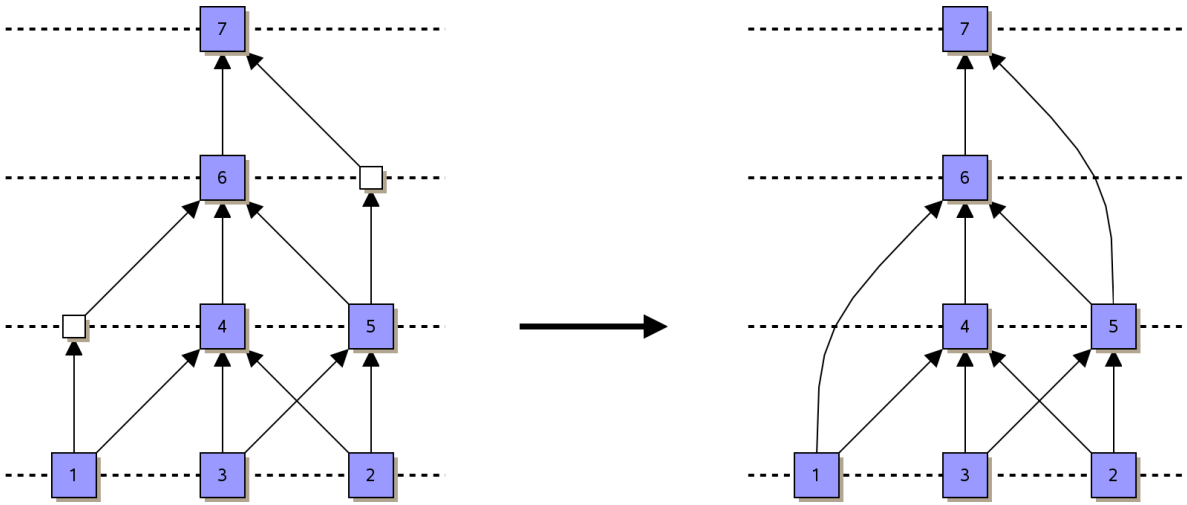
E-Mail-Graph der Fakultät für Informatik



E-Mail-Graph der Fakultät für Informatik



5. Schritt: Kanten zeichnen



Kanten zeichnen

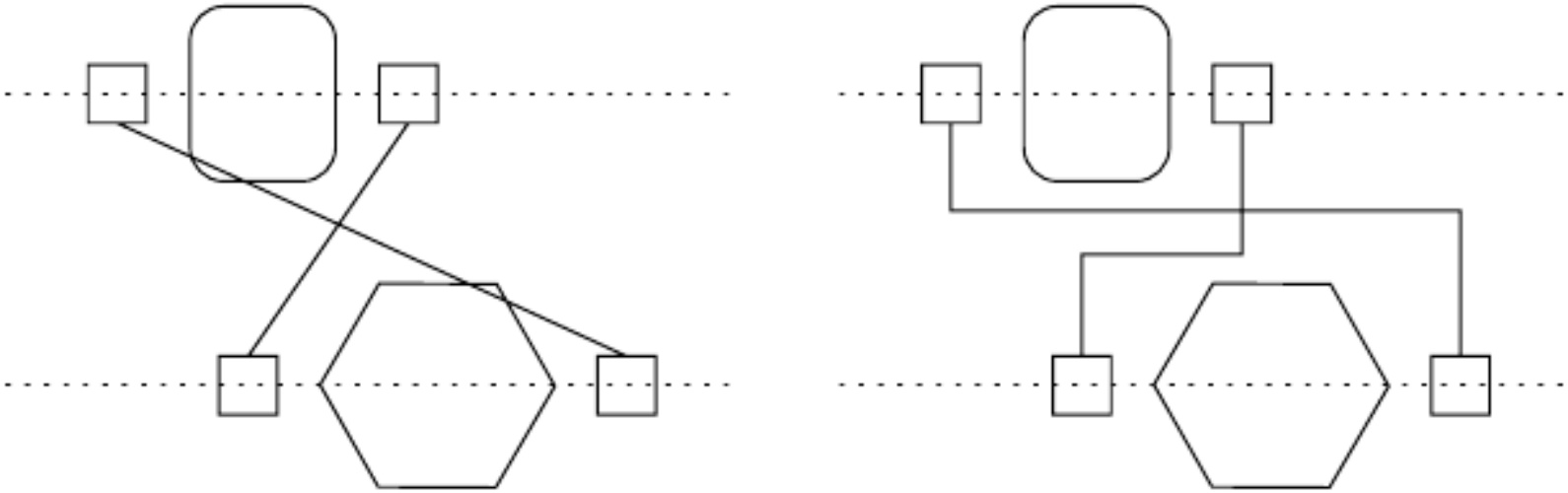


Abb. aus *Drawing Graphs, Kaufmann und Wagner*

Kanten zeichnen

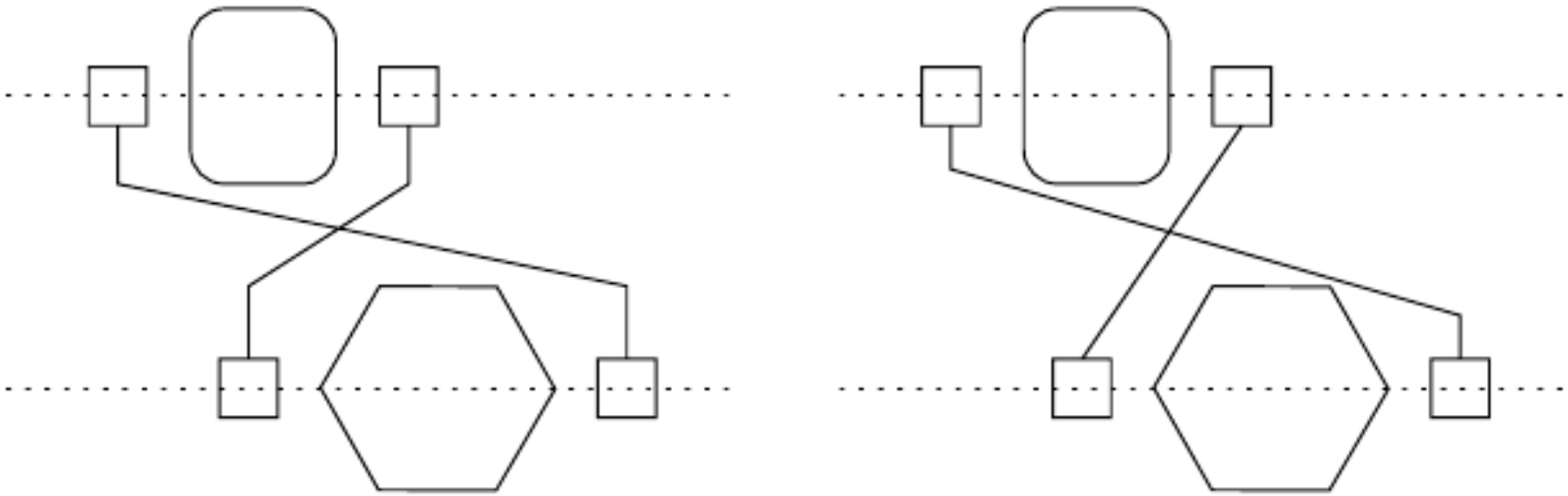


Abb. aus *Drawing Graphs, Kaufmann und Wagner*

Kanten zeichnen

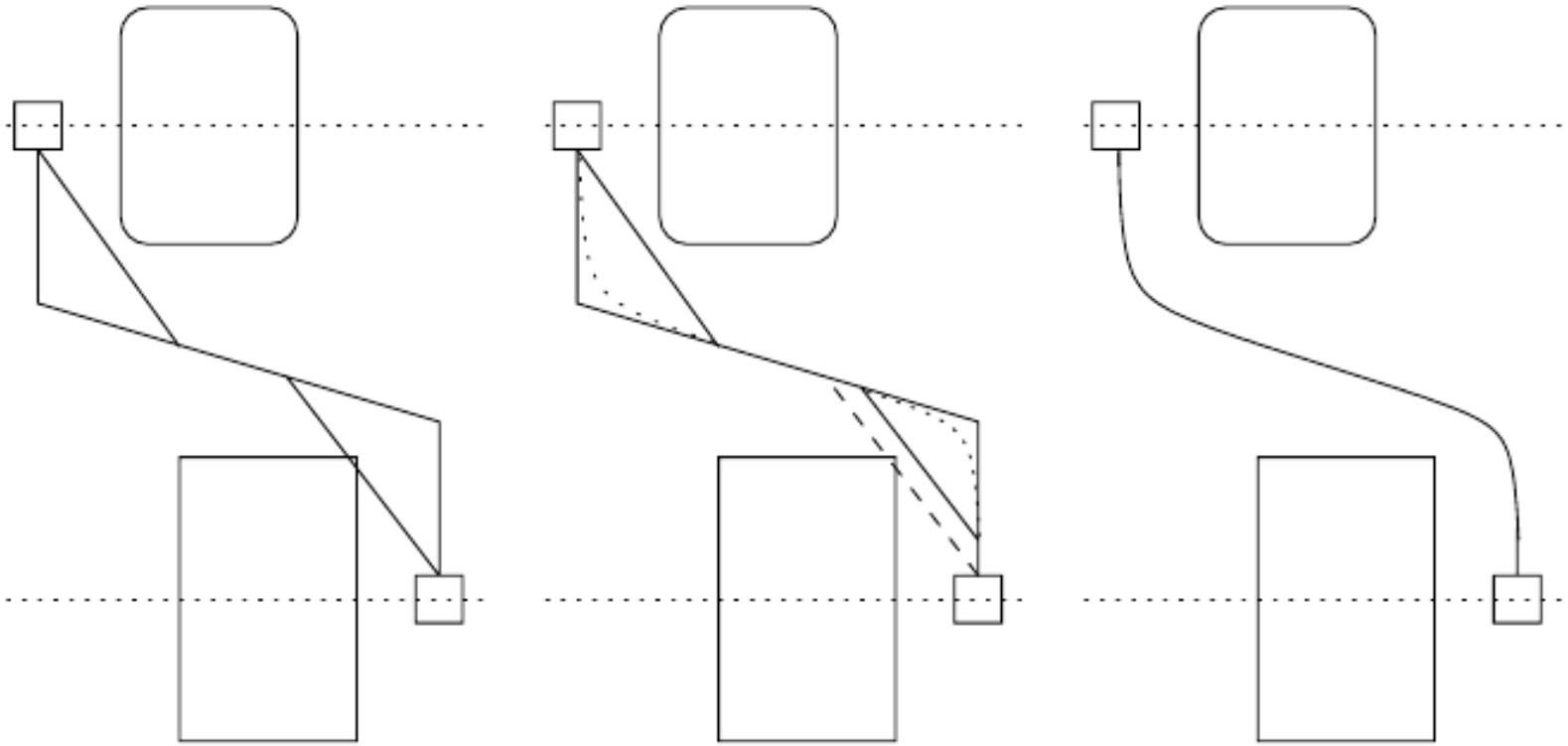
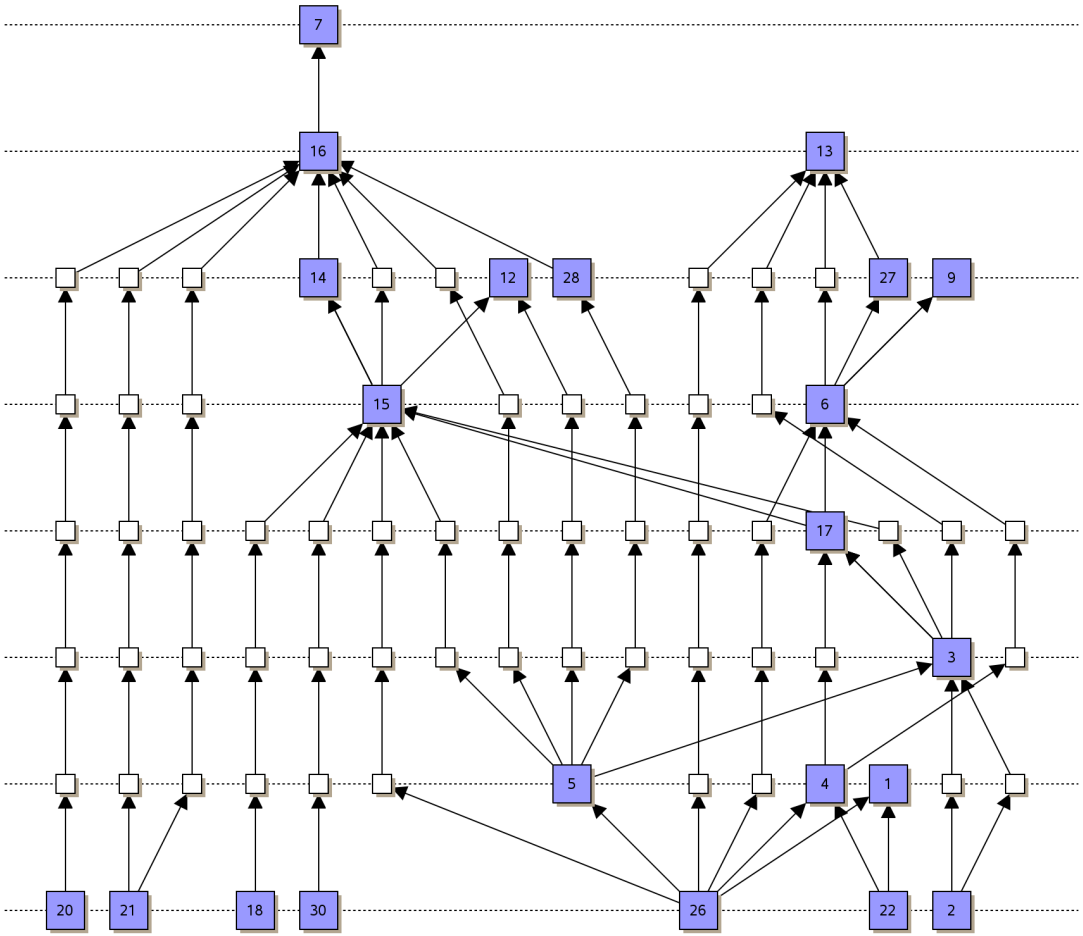
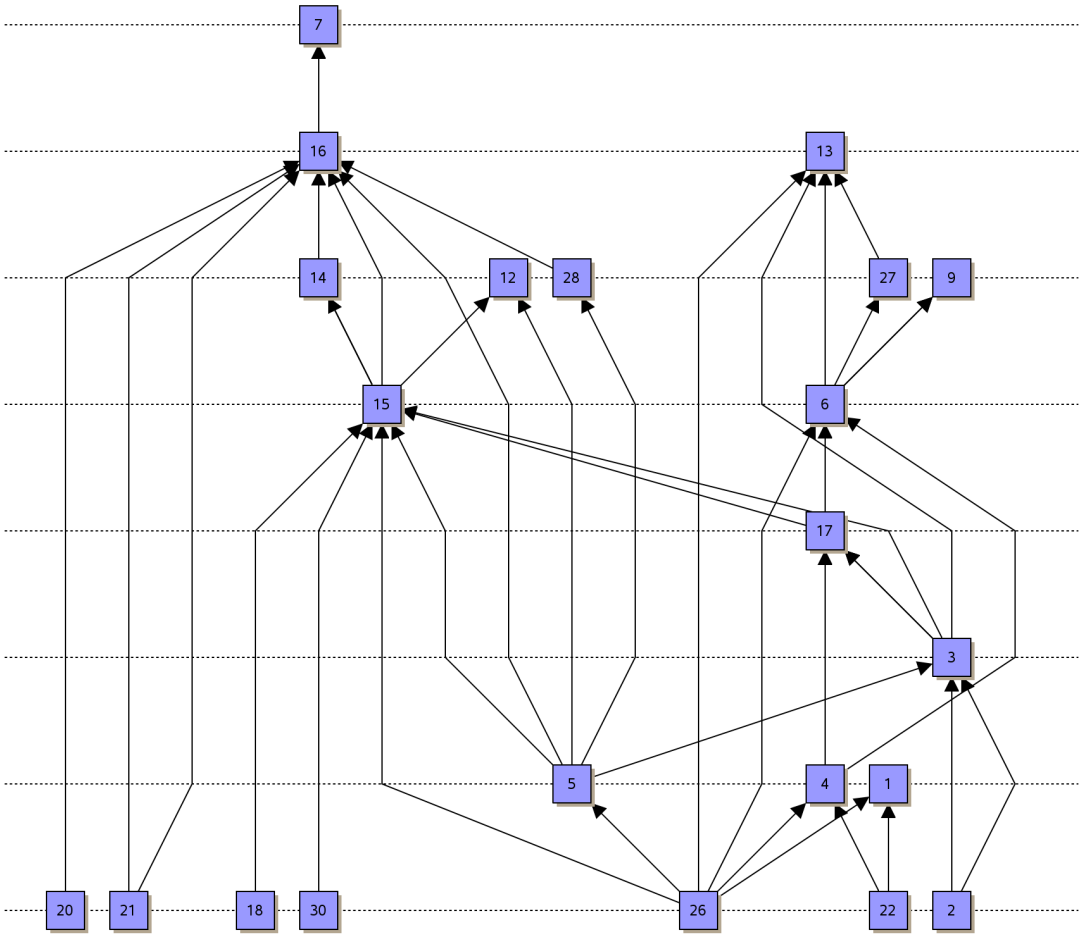


Abb. aus *Drawing Graphs, Kaufmann und Wagner*

E-Mail-Graph der Fakultät für Informatik



E-Mail-Graph der Fakultät für Informatik



E-Mail-Graph der Fakultät für Informatik