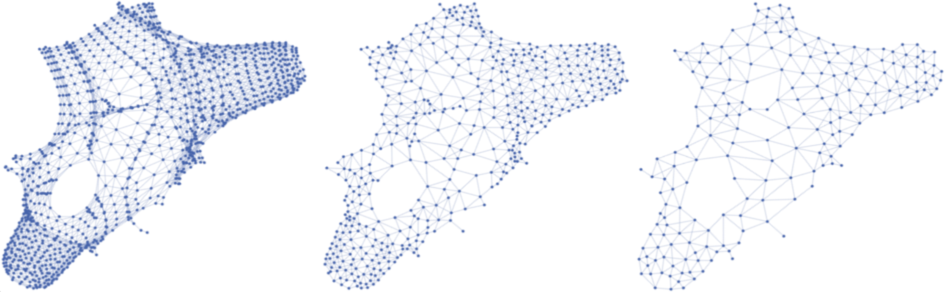
Karlsruhe Institute of Technology

Algorithmen zur Visualisierung von Graphen

Lagenlayouts II

Marcus Krug | WS 2011/12

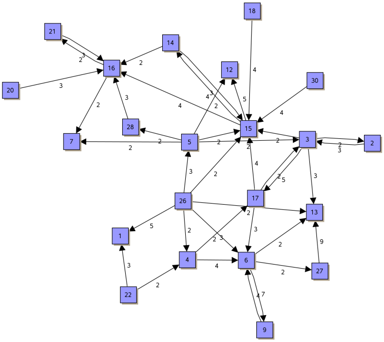
INSTITUTE OF THEORETICAL INFORMATICS
KARLSRUHE INSTITUTE OF TECHNOLOGY (KIT)

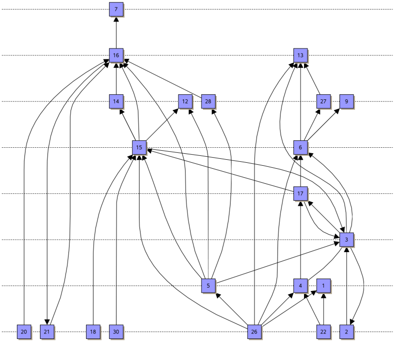


Lagenlayouts

Problemstellung

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





Lagenlayouts

Problemstellung

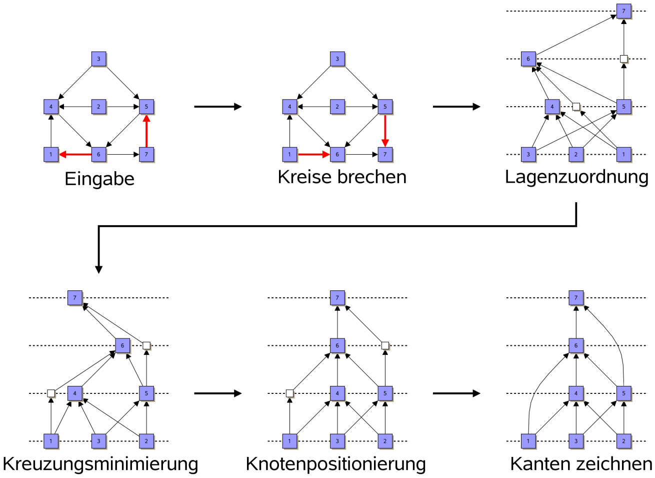
- Gegeben: gerichteter Graph $D = (V, A)$
- Gesucht: Zeichnung von D, die Hierarchie möglichst gut wiedergibt

Desiderata

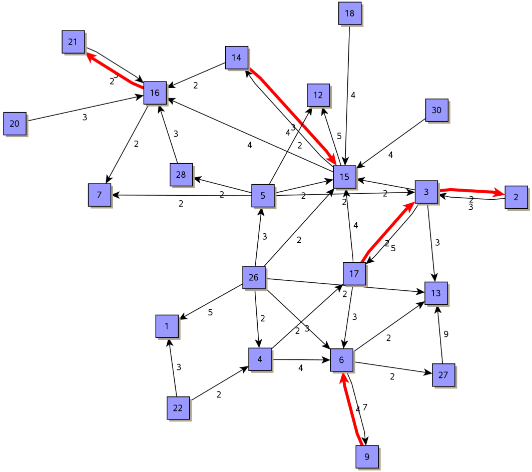
- möglichst viele Kanten aufwärtsgerichtet
- Kanten möglichst geradlinig und kurz
- Zuordnung der Knoten auf (wenige) horizontale Linien
- möglichst wenige Kantenkreuzungen
- Knoten gleichmäßig verteilt

! Kriterien widersprechen sich

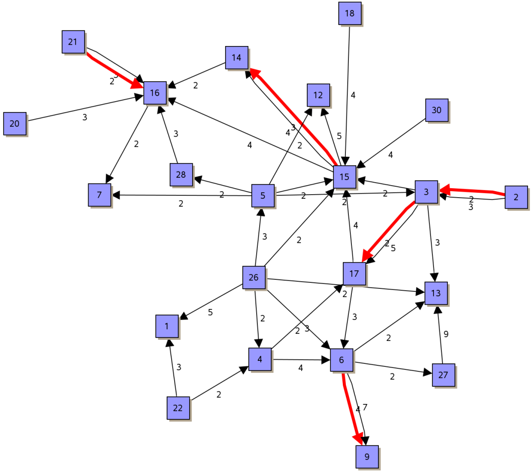
Klassisches Vorgehen (Sugiyama)



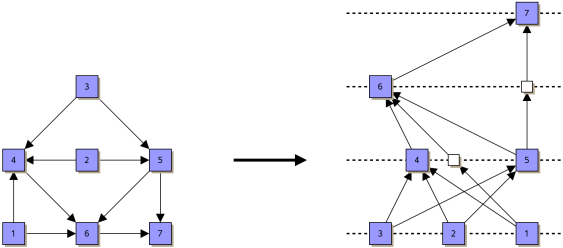
E-Mail-Graph der Fakultät für Informatik



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

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)

ungerichteter Fall?

Lagenzuordnung

- Ordne alle Quellen q Layer 1 zu, d.h. $y(q) = 0$

Lagenzuordnung

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

Lagenzuordnung

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

$$y(u) := \max_{v \in N^{\leftarrow}(u)} \{y(v)\} + 1$$

Lagenzuordnung

- Ordne alle Quellen q Layer 1 zu, d.h. $y(q) = 0$
- sei $N^{\leftarrow}(u)$ Menge der Knoten v mit $(v, u) \in A$
- setze
$$y(u) := \max_{v \in N^{\leftarrow}(u)} \{y(v)\} + 1$$
- d.h. y-Koordinate ist Länge des Längsten Wegs von einer Quelle zu v

Lagenzuordnung

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

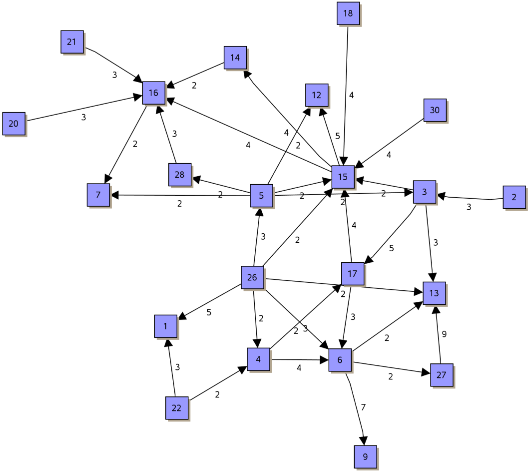
$$y(u) := \max_{v \in N^{\leftarrow}(u)} \{y(v)\} + 1$$

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

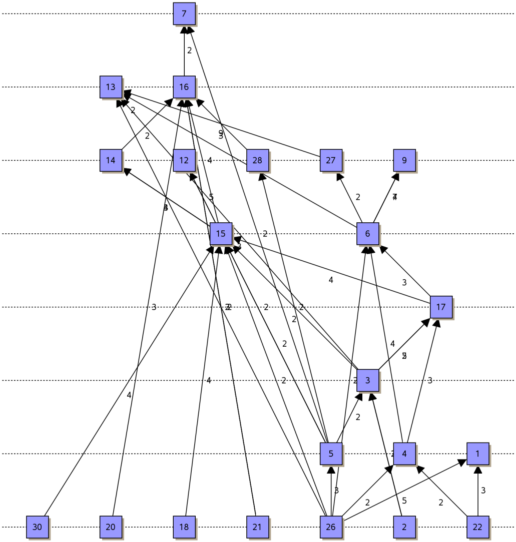
Implementation in Linearzeit

- topologisch sortieren in $\mathcal{O}(n + m)$ Wie?

E-Mail-Graph der Fakultät für Informatik



E-Mail-Graph der Fakultät für Informatik



Minimierung der Kantenlängen

Ganzzahliges lineares Programm

$$\min \sum_{(u,v)\in A} w(u,v) \cdot (y(v) - y(u))$$

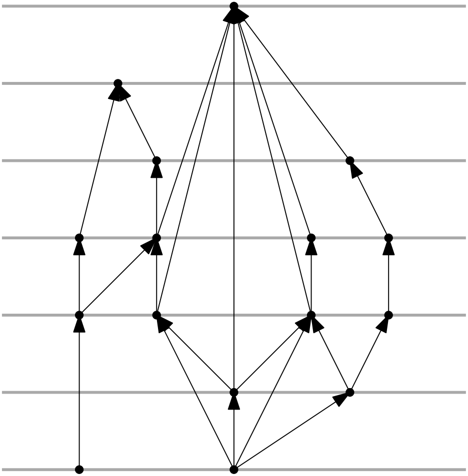
wobei für alle $(u,v) \in A$ gilt

$$y(v) - y(u) \geq \delta(u,v)$$

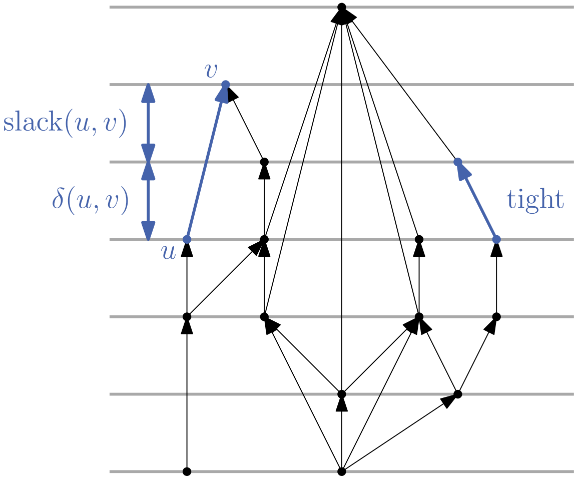
$w(u,v)$ Gewichtung von $(u,v) \in A$
$\delta(u,v)$ Minimallänge von $(u,v) \in A$

Total unimodular $\Rightarrow$ Relaxierung besitzt optimale ganzzahlige Lösung

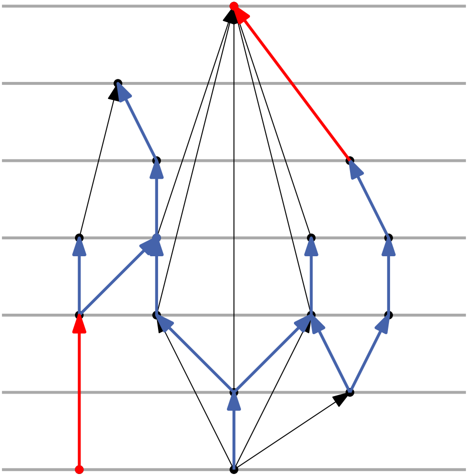
Optimierung mit Netzwerk-Simplex Methode



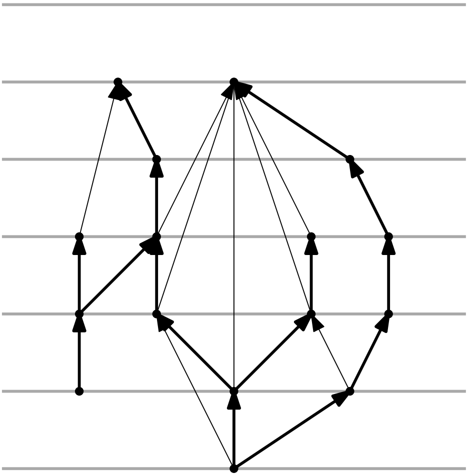
Optimierung mit Netzwerk-Simplex Methode



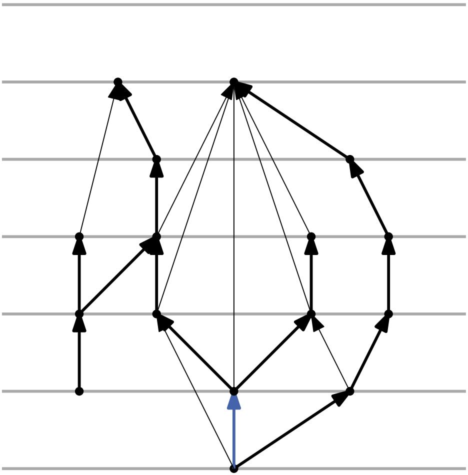
Optimierung mit Netzwerk-Simplex Methode



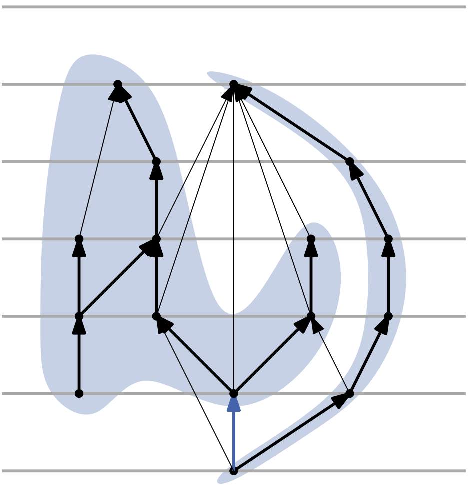
Optimierung mit Netzwerk-Simplex Methode



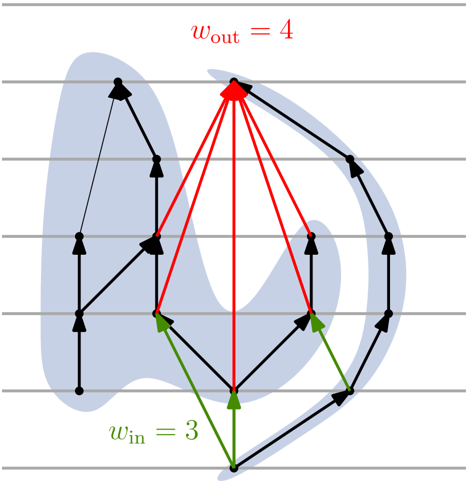
Optimierung mit Netzwerk-Simplex Methode



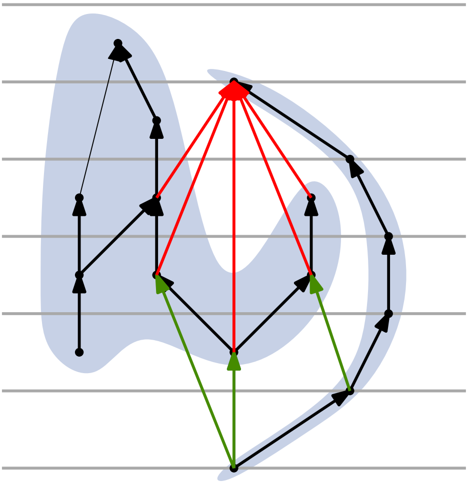
Optimierung mit Netzwerk-Simplex Methode



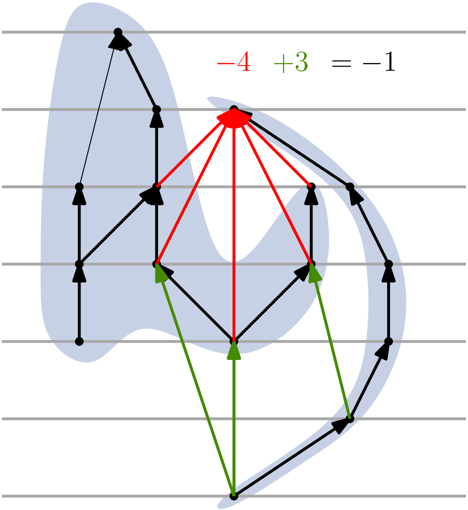
Optimierung mit Netzwerk-Simplex Methode



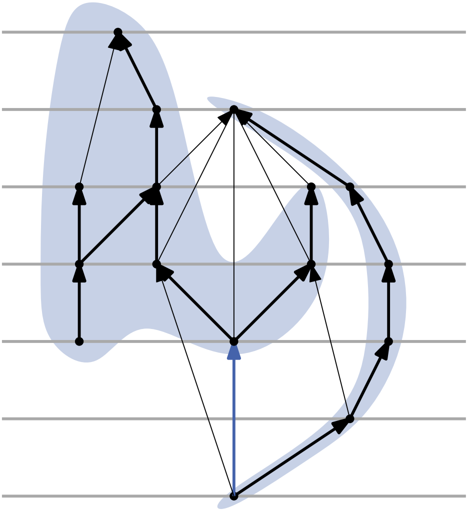
Optimierung mit Netzwerk-Simplex Methode



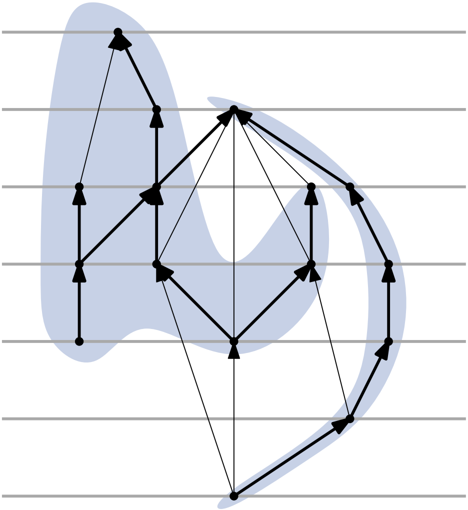
Optimierung mit Netzwerk-Simplex Methode



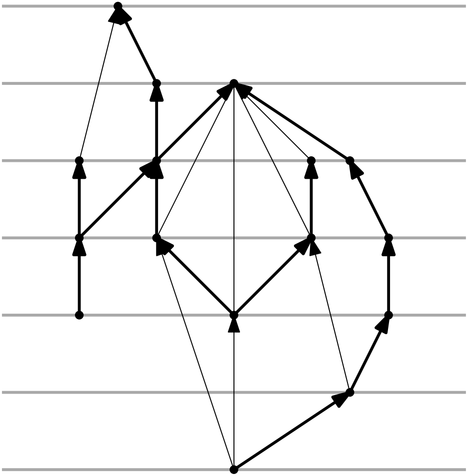
Optimierung mit Netzwerk-Simplex Methode



Optimierung mit Netzwerk-Simplex Methode



Optimierung mit Netzwerk-Simplex Methode



Implementierung

Algorithmus

- Bestimme Initiallösung, Spannbaum, tight
- solange Verbesserung möglich
  - suche Schnittkante e mit $w_{\text{out}} - w_{\text{in}} > 0$
  - suche Ersatzkante f mit minimalem Slack
  - ersetze e durch f

Implementierung

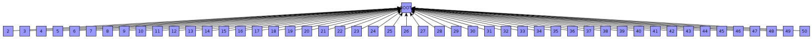
Algorithmus

- Bestimme Initiallösung, Spannbaum, tight
- solange Verbesserung möglich
  - suche Schnittkante e mit $w_{\text{out}} - w_{\text{in}} > 0$
  - suche Ersatzkante f mit minimalem Slack
  - ersetze e durch f

Bemerkung

- Laufzeit unklar, aber in der Praxis effizient

Limitationen



Lagenzuordnung bei vorgegebener Breite

Problem LAYER ASSIGNMENT

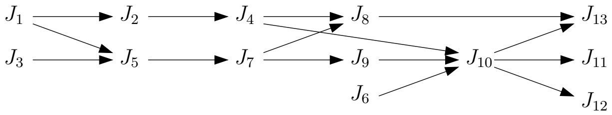
- Gegeben: azyklischer, gerichteter Graph $D = (V, A)$ und Breite B
- Finde Partition der Knotenmenge mit minimaler Anzal von Lagen, so dass in jeder Lage höchstens B Elemente sind

- $\mathcal{NP}$-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 mit minimaler Bearbeitunszeit, der $<$ berücksichtigt



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 mit minimaler Bearbeitunszeit, der $<$ berücksichtigt

- $\mathcal{NP}$-schwer

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 mit minimaler Bearbeitunszeit, der $<$ berücksichtigt

- $\mathcal{NP}$-schwer
- nicht $(4/3 - \varepsilon)$-approximierbar

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 mit minimaler Bearbeitunszeit, der $<$ berücksichtigt

- $\mathcal{NP}$-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:

Lagenzuordnung bei vorgegebener Breite

$(2 - 1/B)$-Approximation

- Verfahren wie bei List-Scheduling:
- Knoten sind in Liste L gespeichert (beliebige Reihenfolge)

Lagenzuordnung bei vorgegebener Breite

$(2 - 1/B)$-Approximation

- Verfahren wie bei List-Scheduling:
- Knoten sind in Liste L gespeichert (beliebige Reihenfolge)
- betrachte Layer in aufsteigender Reihenfolge

Lagenzuordnung bei vorgegebener Breite

$(2 - 1/B)$-Approximation

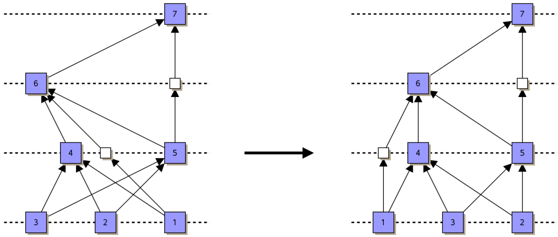
- Verfahren wie bei List-Scheduling:
- Knoten sind in Liste L gespeichert (beliebige Reihenfolge)
- betrachte Layer in aufsteigender Reihenfolge
- Knoten heißt verfügbar, falls alle Vorgänger kleineren Layern zugeordnet sind

Lagenzuordnung bei vorgegebener Breite

$(2 - 1/B)$-Approximation

- Verfahren wie bei List-Scheduling:
- Knoten sind in Liste L gespeichert (beliebige Reihenfolge)
- betrachte Layer in aufsteigender Reihenfolge
- Knoten heißt verfügbar, falls alle Vorgänger kleineren Layern zugeordnet sind
- 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

Iterative Kreuzungsreduktion

Iterative Kreuzungsreduktion

- füge Dummy-Knoten für Kanten mit Layerabstand > 1 ein
- betrachte jeweils benachbarte Layer nacheinander

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

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

Iterative Kreuzungsreduktion

(1) berechne zufällige Permutation für unterstes Layer

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

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

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}

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) $\leadsto$ 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

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

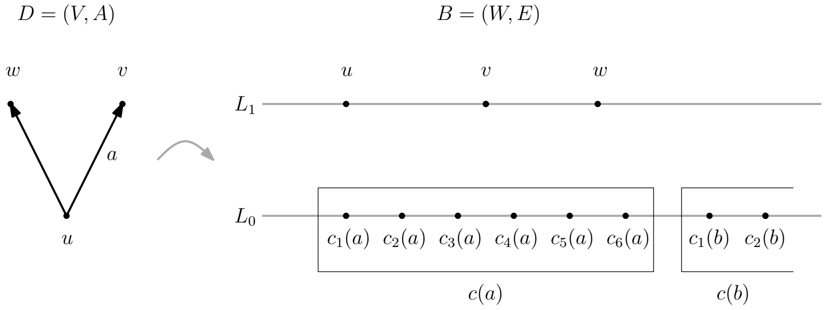
Einseitige Kreuzungsreduktion

Einseitige Kreuzungsminimierung

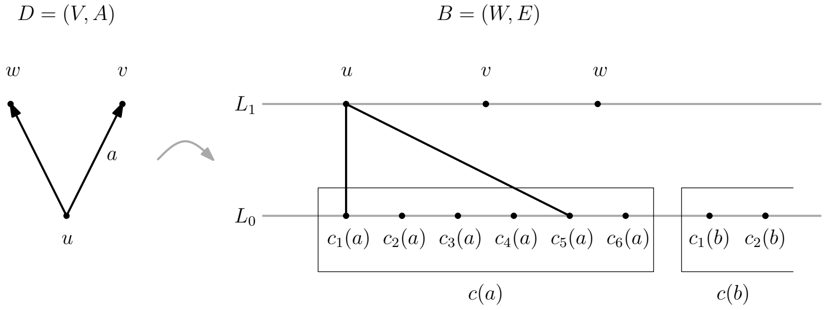
- Gegeben: Graph G mit Partition L_1, L_2 der Kanten und gegebener Ordnung (Permutation) π_1 der Knoten in L_1
- Finde Knotenordnung π_2 auf L_2, so dass die Anzahl der Kantenpaare, die sich kreuzen, minimiert wird

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

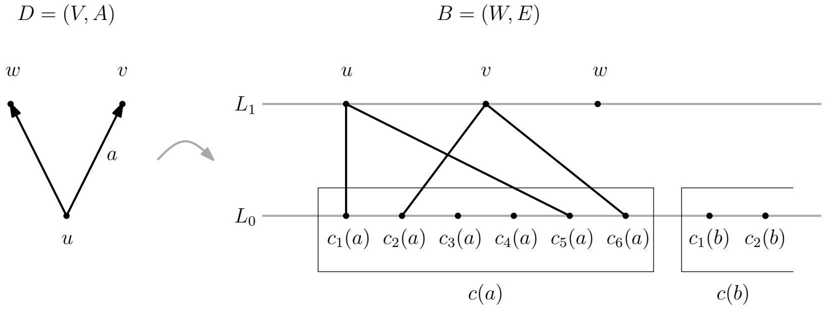
Reduktion von Feedback Arc Set (FAS)



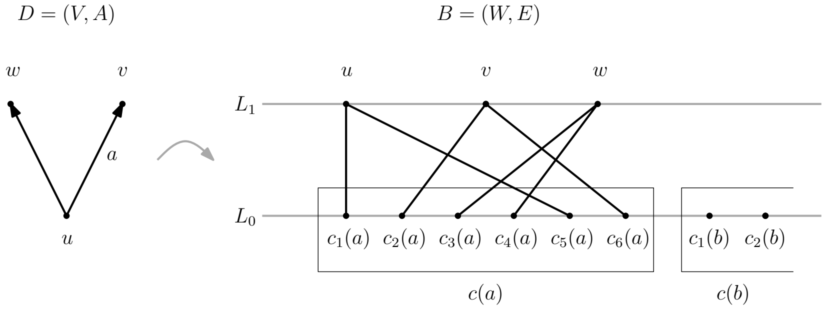
Reduktion von Feedback Arc Set (FAS)



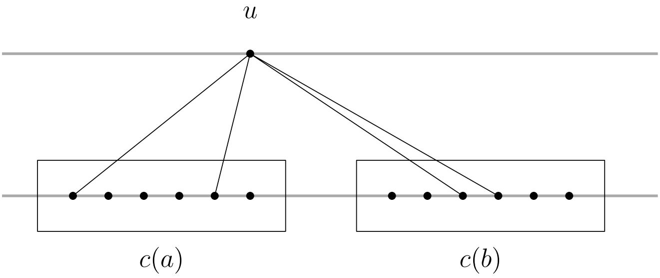
Reduktion von Feedback Arc Set (FAS)



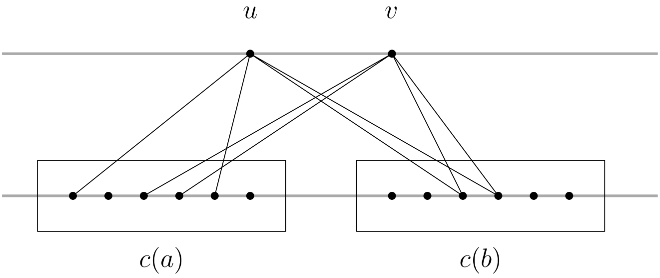
Reduktion von Feedback Arc Set (FAS)



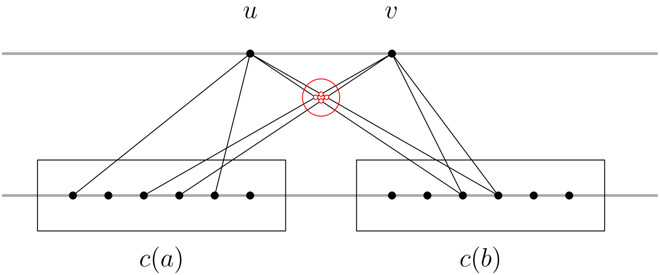
Reduktion von Feedback Arc Set (FAS)



Reduktion von Feedback Arc Set (FAS)



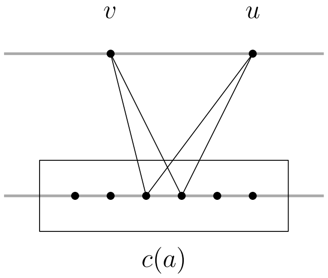
Reduktion von Feedback Arc Set (FAS)



$$4 \cdot \binom{n}{2} \cdot \binom{m}{2}$$

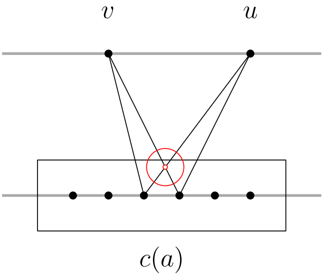
Reduktion von Feedback Arc Set (FAS)

$4 \cdot \binom{n}{2} \cdot \binom{m}{2}$



Reduktion von Feedback Arc Set (FAS)

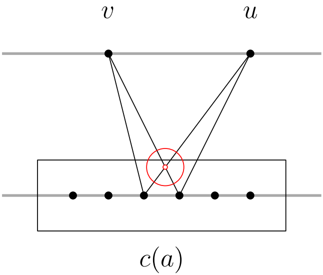
$4 \cdot \binom{n}{2} \cdot \binom{m}{2}$

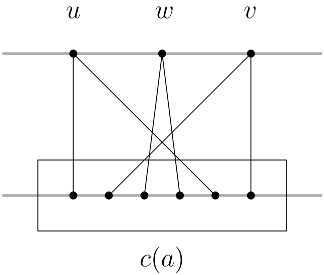


$+m \cdot \binom{n-2}{2}$

Reduktion von Feedback Arc Set (FAS)

$4 \cdot \binom{n}{2} \cdot \binom{m}{2}$

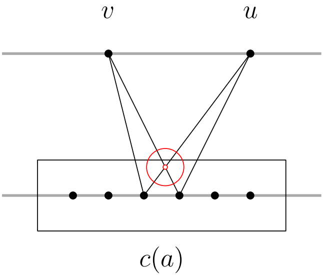




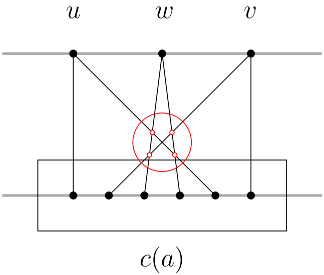
$+m \cdot \binom{n-2}{2}$

Reduktion von Feedback Arc Set (FAS)

$4 \cdot \binom{n}{2} \cdot \binom{m}{2}$



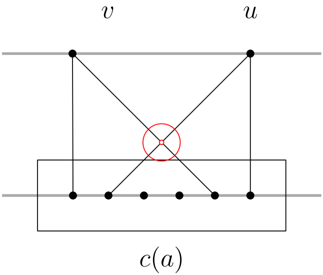
$+m \cdot \binom{n-2}{2}$



$+4 \cdot m \cdot (n-2)$

Reduktion von Feedback Arc Set (FAS)

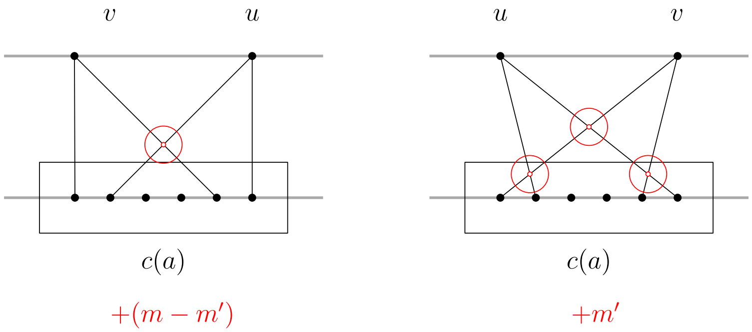
$$4 \cdot \binom{n}{2} \cdot \binom{m}{2} + m \cdot \binom{n-2}{2} + 4 \cdot m \cdot (n-2) =: M$$



$+(m - m')$

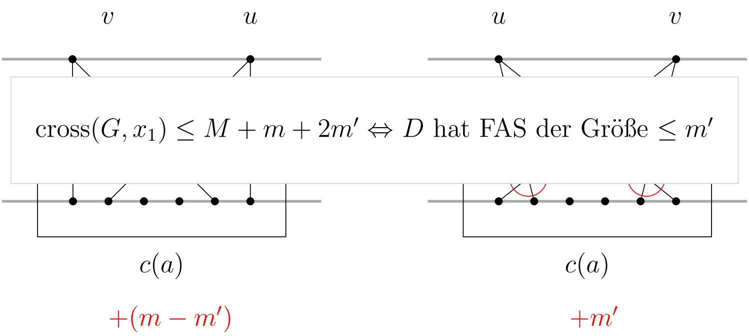
Reduktion von Feedback Arc Set (FAS)

$$4 \cdot \binom{n}{2} \cdot \binom{m}{2} + m \cdot \binom{n-2}{2} + 4 \cdot m \cdot (n-2) =: M$$



Reduktion von Feedback Arc Set (FAS)

$$4 \cdot \binom{n}{2} \cdot \binom{m}{2} + m \cdot \binom{n-2}{2} + 4 \cdot m \cdot (n-2) =: M$$



$$\text{cross}(G, x_1) \leq M + m + 2m' \Leftrightarrow D \text{ hat FAS der Größe} \leq m'$$

Einseitige Kreuzungsreduktion

Heuristiken

- Baryzentrisch
- Median
- Greedy Switch
- uvm.

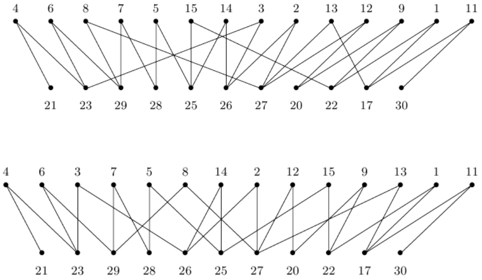


Abb. aus *Drawing Graphs, Kaufmann und Wagner*

Exakt

- ILP

Bemerkung: Heuristiken funktionieren besser für dichte Graphen

Einseitige Kreuzungsreduktion

Baryzenter-Heuristik (Sugiyama et al., 1981)

- Intuition: wenige Kreuzungen, wenn Knoten nah bei Nachbarn

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

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

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

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

- setze $x_2(u) = \mathrm{bary}(u)$

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

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

- setze $x_2(u) = \mathrm{bary}(u)$
- bei gleichen Werten werden Knoten um einen Wert δ versetzt

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

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

- setze $x_2(u) = \mathrm{bary}(u)$
- bei gleichen Werten werden Knoten um einen Wert δ versetzt
- sortiere Knoten nach x-Koordinate, um Ordnung der Knoten zu erhalten

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

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

- setze $x_2(u) = \mathrm{bary}(u)$
- bei gleichen Werten werden Knoten um einen Wert δ versetzt
- sortiere Knoten nach x-Koordinate, um Ordnung der Knoten zu erhalten
- Laufzeit $\mathcal{O}(|E_{12}| + |V_2| \log |V_2|)$

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 $\pi_1(v_1) < \pi_1(v_2) < \cdots < \pi_1(v_k)$

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

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

Einseitige Kreuzungsreduktion

Median-Heuristik (Eades und Wormald, 1994)

- geringer Implementationsaufwand
- schnell
- relativ gute Ergebnisse
- ebenfalls keine Kreuzung, falls möglich
- 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)

- Betrachte Variablen δ^1_{ij} $(1 \leq i < j \leq n_1)$

$$\delta^1_{ij} = \begin{cases} 1 & \text{falls } \pi_1(v_i) < \pi_1(v_j) \\ 0 & \text{sonst} \end{cases}$$

- und δ^2_{ij} $(1 \leq i < j \leq n_2)$

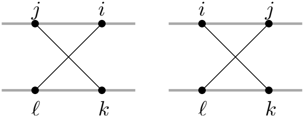
$$\delta^2_{ij} = \begin{cases} 1 & \text{falls } \pi_2(v_i) < \pi_2(v_j) \\ 0 & \text{sonst} \end{cases}$$

- $N(i)$ L_1-Nachbarn von $i \in L_2$

Optimale Einseitige Kreuzungsreduktion

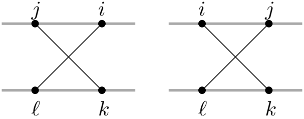
$\text{cross}(\pi_2) =$

$$\underbrace{\delta^1_{k\ell} \cdot \delta^2_{ji} + \delta^1_{\ell k} \cdot \delta^2_{ij}}$$



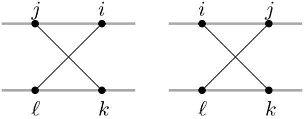
Optimale Einseitige Kreuzungsreduktion

$$\text{cross}(\pi_2) = \sum_{k \in N(i)} \sum_{\ell \in N(j)} \underbrace{\delta^1_{k\ell} \cdot \delta^2_{ji} + \delta^1_{\ell k} \cdot \delta^2_{ij}}$$



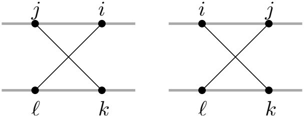
Optimale Einseitige Kreuzungsreduktion

$$\text{cross}(\pi_2) = \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} \sum_{k \in N(i)} \sum_{\ell \in N(j)} \underbrace{\delta^1_{k\ell} \cdot \delta^2_{ji} + \delta^1_{\ell k} \cdot \delta^2_{ij}}$$



Optimale Einseitige Kreuzungsreduktion

$$\mathrm{cross}(\pi_2) = \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} \sum_{k \in N(i)} \sum_{\ell \in N(j)} \underbrace{\delta^1_{k\ell} \cdot \delta^2_{ji} + \delta^1_{\ell k} \cdot \delta^2_{ij}}$$

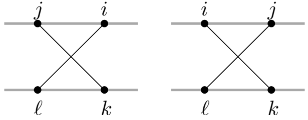


$$= \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} \left(c_{ij}\delta^2_{ij} + c_{ji}(1 - \delta^2_{ij}) \right)$$

mit $c_{ij} = \sum_{k \in N(i)} \sum_{\ell \in N(j)} \delta^1_{\ell k}$: # Kreuzungen, falls $\pi_2(i) < \pi_2(j)$

Optimale Einseitige Kreuzungsreduktion

$$\mathrm{cross}(\pi_2) = \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} \sum_{k \in N(i)} \sum_{\ell \in N(j)} \underbrace{\delta^1_{k\ell} \cdot \delta^2_{ji} + \delta^1_{\ell k} \cdot \delta^2_{ij}}$$



$$= \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} \left(c_{ij}\delta^2_{ij} + c_{ji}(1 - \delta^2_{ij}) \right)$$

mit $c_{ij} = \sum_{k \in N(i)} \sum_{\ell \in N(j)} \delta^1_{\ell k}$: # Kreuzungen, falls $\pi_2(i) < \pi_2(j)$

$$= \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} (c_{ij} - c_{ji})\delta^2_{ij} + \overbrace{\sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} c_{ji}}^{\text{konstant}}$$

Optimale Einseitige Kreuzungsreduktion

ILP-Modellierung (Jünger und Mutzel, 1997)

- Minimiere Anzahl der Kreuzungen:

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

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:

$$\begin{aligned} 0 \le x_{ij} + x_{jk} - x_{ik} \le 1 & \qquad 1 \le i < j < k \le n_2 \\ 0 \le x_{ij} \le 1 & \qquad 1 \le i < j < k \le n_2 \\ x_{ij} \in \mathbb{Z} & \qquad 1 \le i < j < k \le n_2 \end{aligned}$$

Optimale Einseitige Kreuzungsreduktion

ILP-Modellierung (Jünger und Mutzel, 1997)

- Minimiere Anzahl der Kreuzungen:

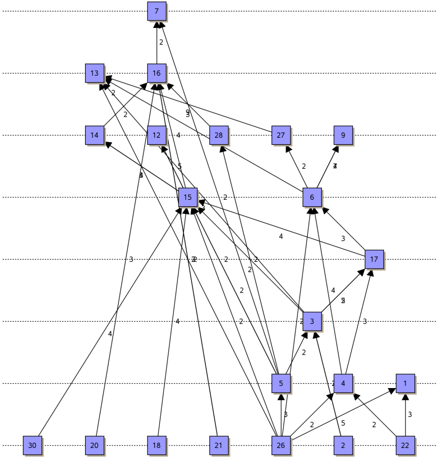
$$\min \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} (c_{ij} - c_{ji}) x_{ij}$$

- Nebenbedingungen:

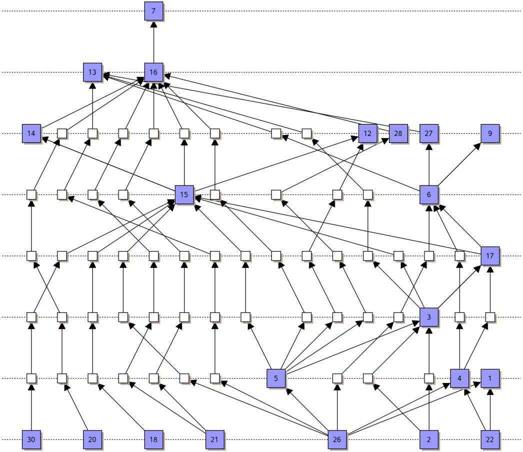
$$\begin{aligned} 0 \leq x_{ij} + x_{jk} - x_{ik} \leq 1 & \qquad 1 \leq i < j < k \leq n_2 \\ 0 \leq x_{ij} \leq 1 & \qquad 1 \leq i < j < k \leq n_2 \\ x_{ij} \in \mathbb{Z} & \qquad 1 \leq i < j < k \leq n_2 \end{aligned}$$

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

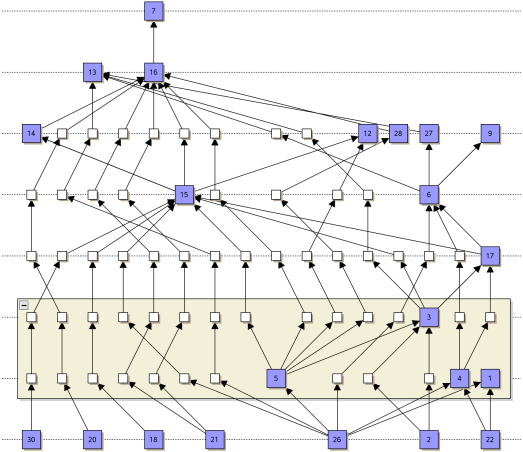
E-Mail-Graph der Fakultät für Informatik



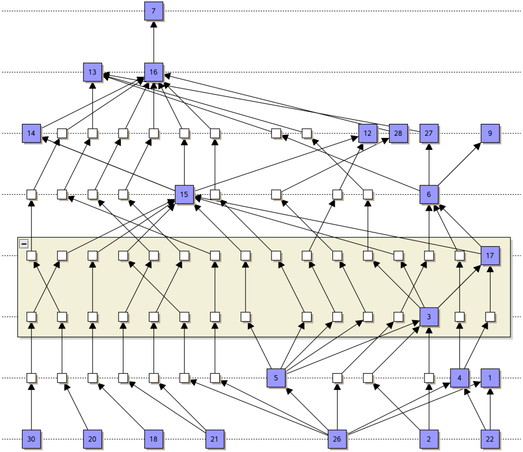
E-Mail-Graph der Fakultät für Informatik



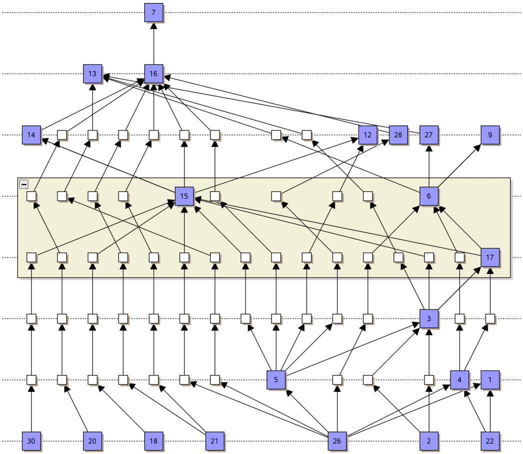
E-Mail-Graph der Fakultät für Informatik



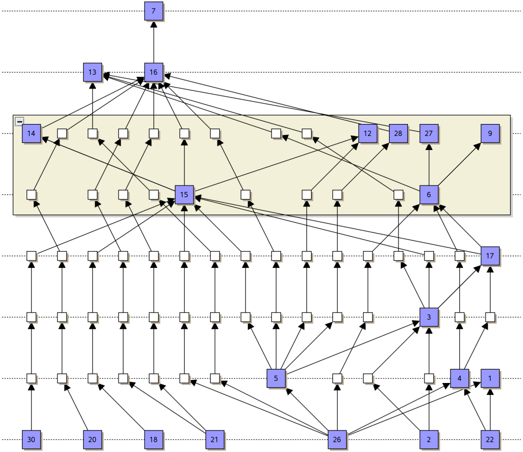
E-Mail-Graph der Fakultät für Informatik



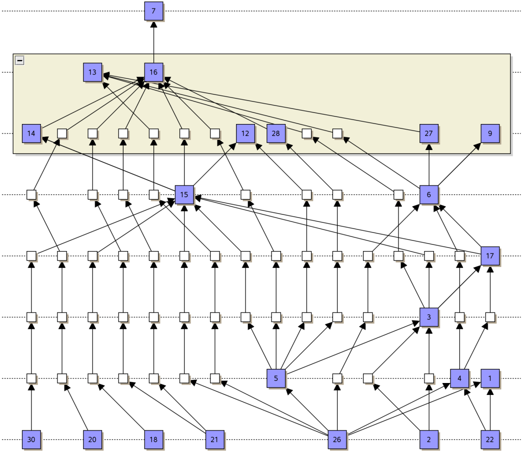
E-Mail-Graph der Fakultät für Informatik



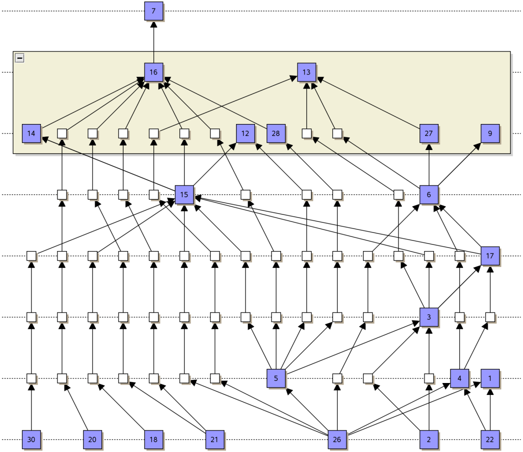
E-Mail-Graph der Fakultät für Informatik



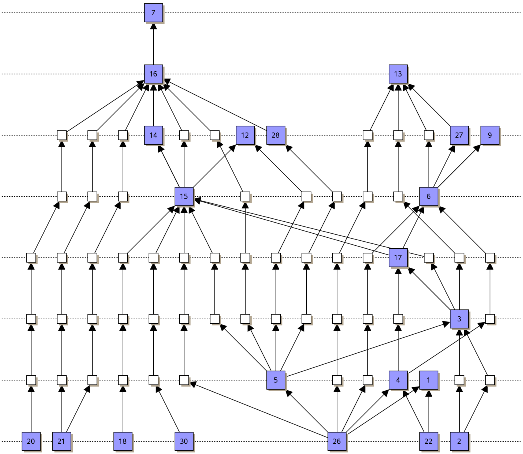
E-Mail-Graph der Fakultät für Informatik



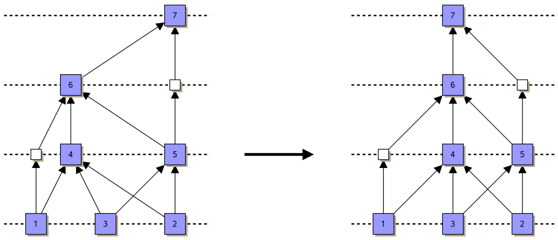
E-Mail-Graph der Fakultät für Informatik



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):

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

- 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:

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

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

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

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

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 sehr breit

Knotenpositionierung

Quadratisches Programm

- Zielfunktion:

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

- 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 sehr breit
- evtl. weitere Constraints für Breite

Algorithmus von Gansner et al. (1993)

Zielstellung: möglichst vertikale, gerade Kanten

$$\min \sum_{(u,v)\in A} \Omega(u,v) \cdot \omega(u,v) \cdot |x_u - x_v|$$

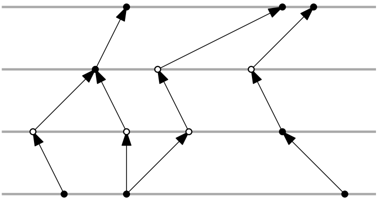
so dass für $u, v \in V$ mit u links von v gilt

$$x_v - x_u \geq \rho(u,v)$$

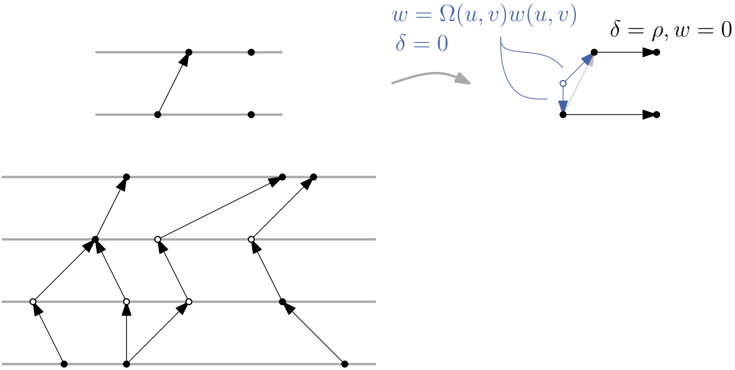
$\Omega(u,v)$ reflektiert Art der Kante, z.B.:

- Regular-Regular $\Omega(u,v) = 1$
- Regular-Dummy $\Omega(u,v) = 2$
- Dummy-Dummy $\Omega(u,v) = 8$

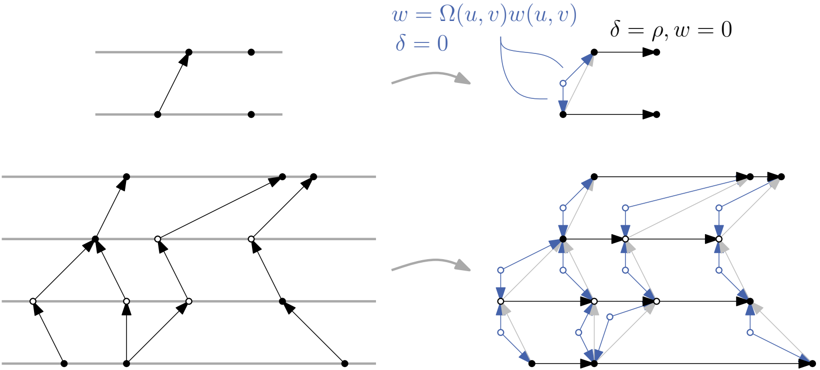
Reduktion auf Lagenzuweisung



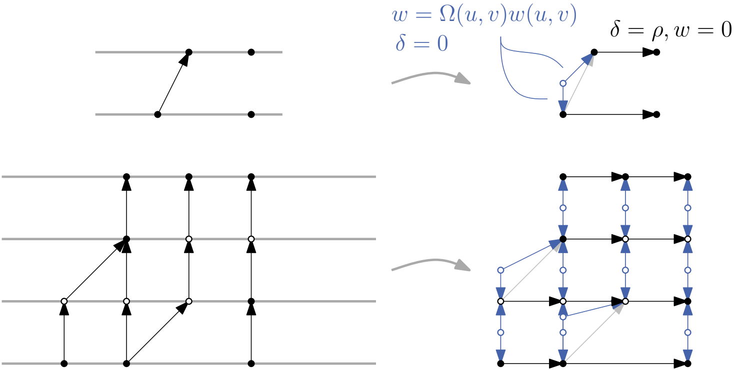
Reduktion auf Lagenzuweisung



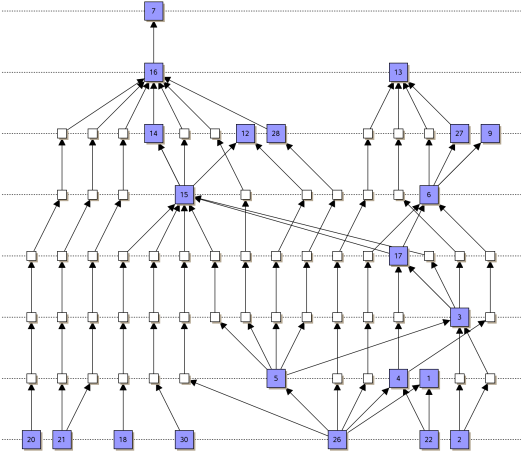
Reduktion auf Lagenzuweisung



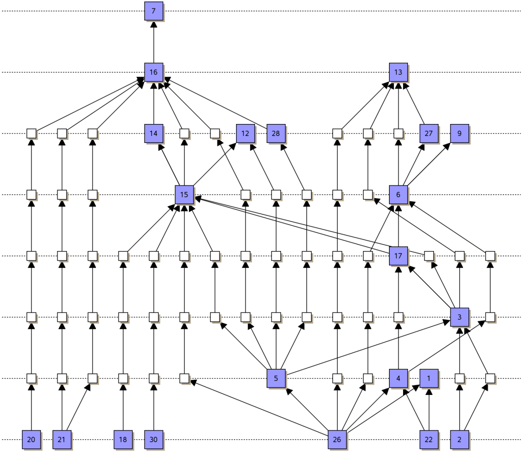
Reduktion auf Lagenzuweisung



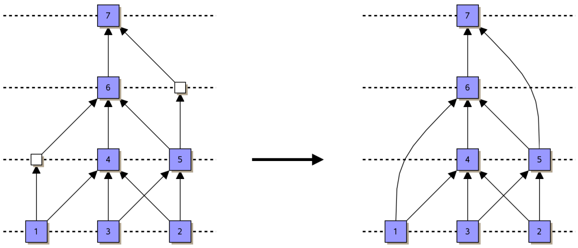
E-Mail-Graph der Fakultät für Informatik



E-Mail-Graph der Fakultät für Informatik



5. Schritt: Kanten zeichnen



Kanten zeichnen

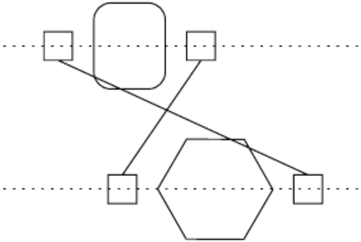


Abb. aus *Drawing Graphs, Kaufmann und Wagner*

Kanten zeichnen

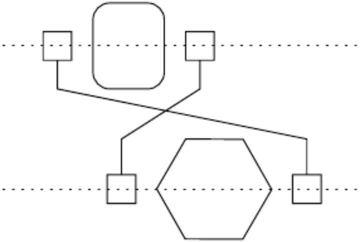


Abb. aus *Drawing Graphs, Kaufmann und Wagner*

Kanten zeichnen

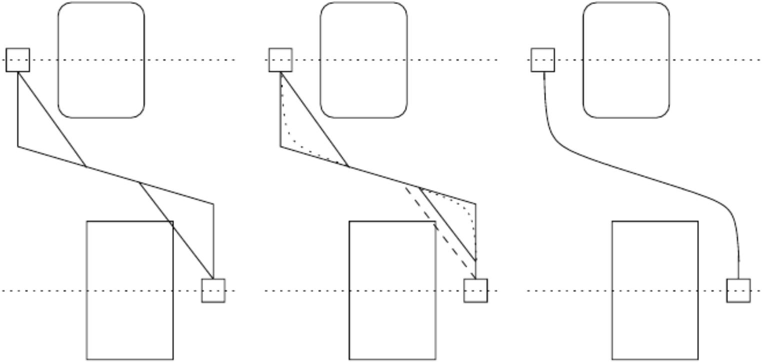
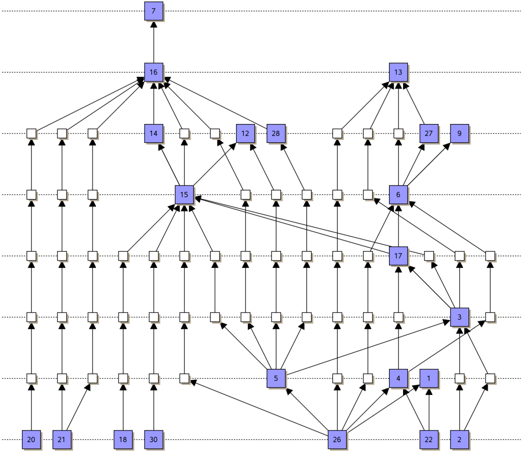
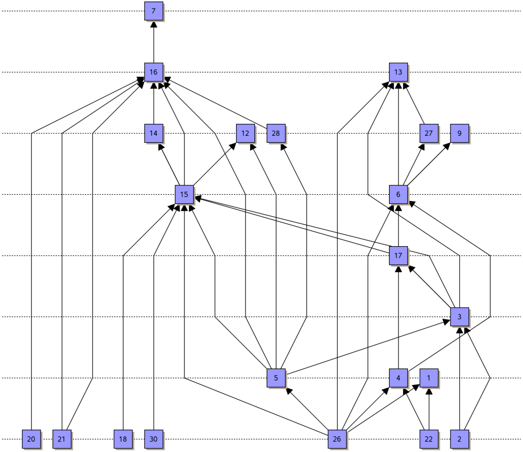


Abb. aus *Drawing Graphs, Kaufmann und Wagner*

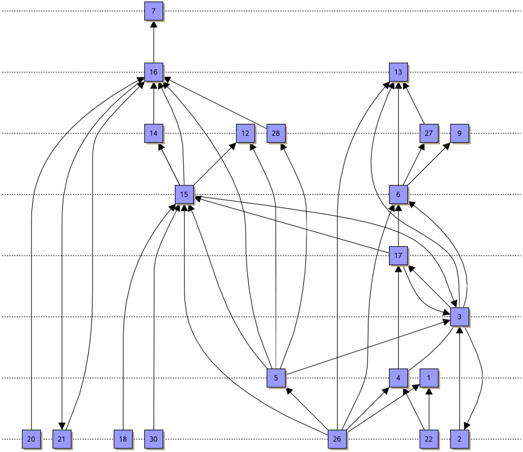
E-Mail-Graph der Fakultät für Informatik



E-Mail-Graph der Fakultät für Informatik



E-Mail-Graph der Fakultät für Informatik



Zusammenfassung

- Framework zur Visualisierung von gerichteten Graphen
- Reduktion der Komplexität durch unabhängige Optimierung verschiedener Desiderata

\+ Modularisierung, überschaubare Optimierungsprobleme

– Einschränkung der Lösungen

Demo

Exkurs: Kreuzungszahl

Problem Kreuzungsminimierung

- Gegeben: Graph $G = (V, E)$
- Gesucht: Zeichnung von G mit minimaler Anzahl von Kreuzungen
- Varianten: topologisch, geradlinig

Exkurs: Kreuzungszahl

Problem Kreuzungsminimierung

- Gegeben: Graph $G = (V, E)$
- Gesucht: Zeichnung von G mit minimaler Anzahl von Kreuzungen
- Varianten: topologisch, geradlinig

Definition (Kreuzungszahl)

Die Kreuzungszahl $cr(G)$ von $G = (V, E)$ bezeichnet die minimale Anzahl von Kreuzungen, die man in einer (topologischen) Zeichnung von G benötigt.

Komplexität

- Crossing-Number ist NP-schwer, selbst für kubische Graphen [Garey & Johnson, ’83]

Komplexität

- Crossing-Number ist NP-schwer, selbst für kubische Graphen [Garey & Johnson, '83]
- Kreuzungszahl von K_n ist unbekannt für fast alle n

Komplexität

- Crossing-Number ist NP-schwer, selbst für kubische Graphen [Garey & Johnson, '83]
- Kreuzungszahl von K_n ist unbekannt für fast alle n
- Crossing-Number ist FPT mit Parameter Kreuzungszahl

Komplexität

- Crossing-Number ist NP-schwer, selbst für kubische Graphen [Garey & Johnson, '83]
- Kreuzungszahl von K_n ist unbekannt für fast alle n
- Crossing-Number ist FPT mit Parameter Kreuzungszahl
- $m \geq 7.5n$

$$\Rightarrow cr(G) \geq \frac{m^3}{33.75n^2}$$

Heuristik

Idee

- berechne möglichst großen planaren Teilgraphen
- füge restliche Kanten iterativ hinzu

Heuristik

Idee

- berechne möglichst großen planaren Teilgraphen
- füge restliche Kanten iterativ hinzu

Komplexität

- maximalen planaren Teilgraphen finden ist NP-schwer

Heuristik

Idee

- berechne möglichst großen planaren Teilgraphen
- füge restliche Kanten iterativ hinzu

Komplexität

- maximalen planaren Teilgraphen finden ist NP-schwer
- Heuristik (z.B. iteratives Einfügen)

Heuristik

Idee

- berechne möglichst großen planaren Teilgraphen
- füge restliche Kanten iterativ hinzu

Komplexität

- maximalen planaren Teilgraphen finden ist NP-schwer
- Heuristik (z.B. iteratives Einfügen)
- Optimales Einfügen in einer Kante bei gegebener Einbettung entspricht Küzeste-Wege Problem

Heuristik

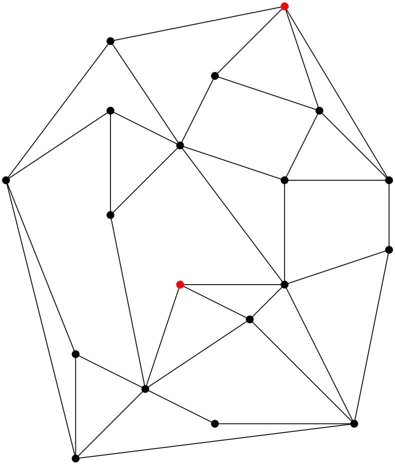
Idee

- berechne möglichst großen planaren Teilgraphen
- füge restliche Kanten iterativ hinzu

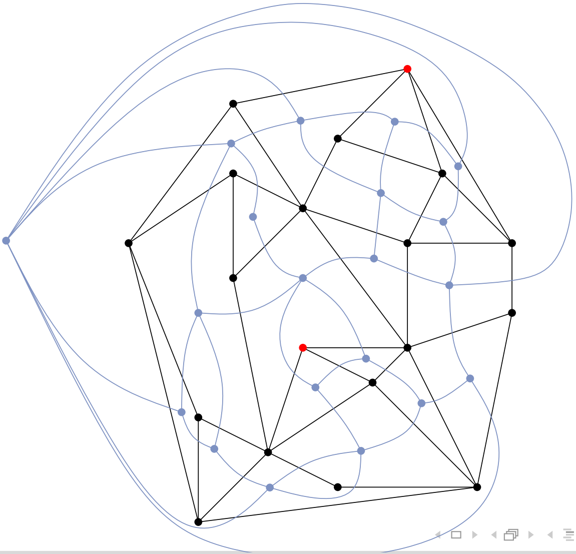
Komplexität

- maximalen planaren Teilgraphen finden ist NP-schwer
- Heuristik (z.B. iteratives Einfügen)
- Optimales Einfügen in einer Kante bei gegebener Einbettung entspricht Küzeste-Wege Problem
- Optimierung über alle Einbettungen mit SPQR-Baum (Gutwenger et al., 2001)

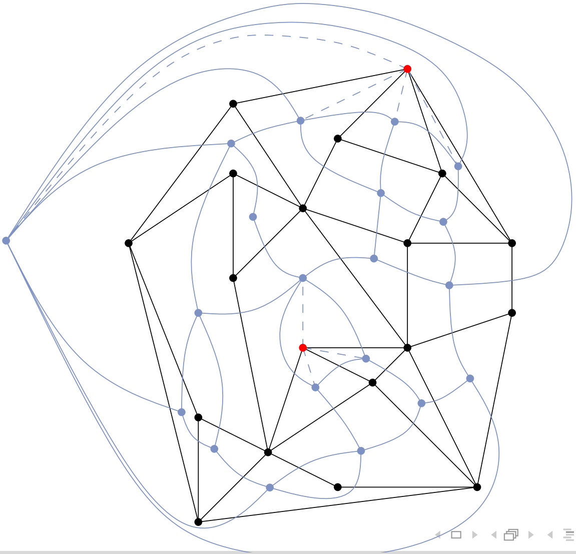
Einfüge-Pfad



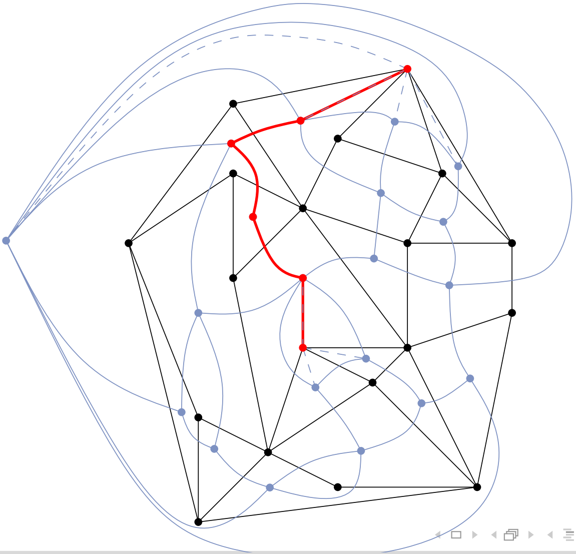
Einfüge-Pfad



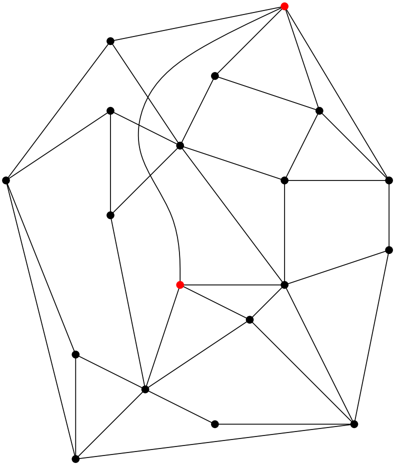
Einfüge-Pfad



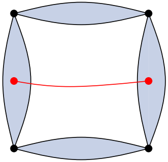
Einfüge-Pfad



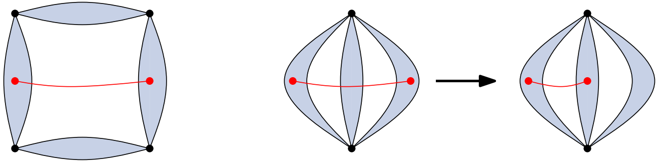
Einfüge-Pfad



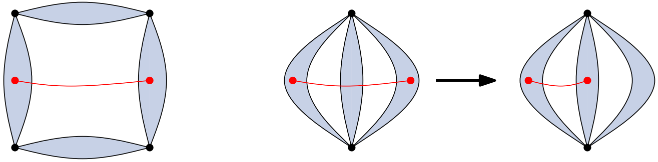
SPQR-Baum

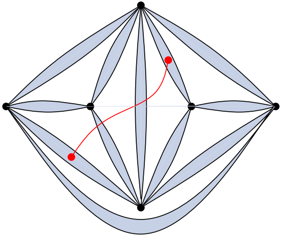


SPQR-Baum

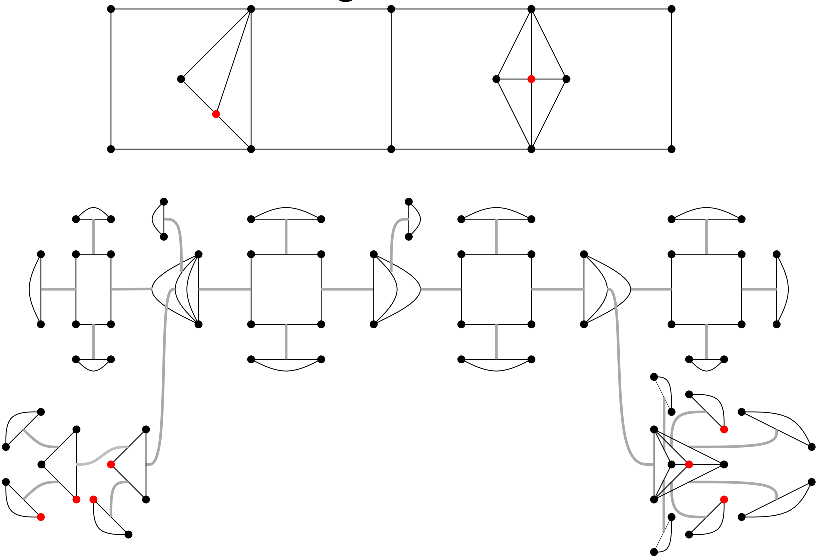


SPQR-Baum

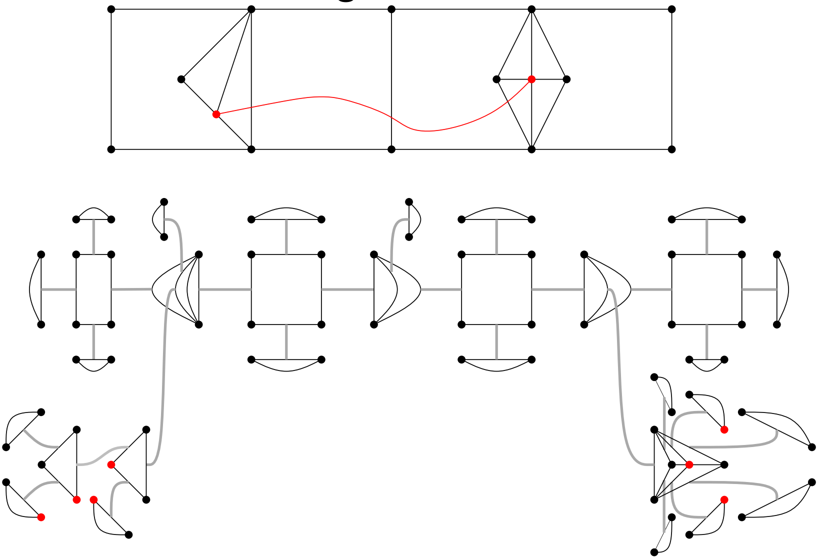




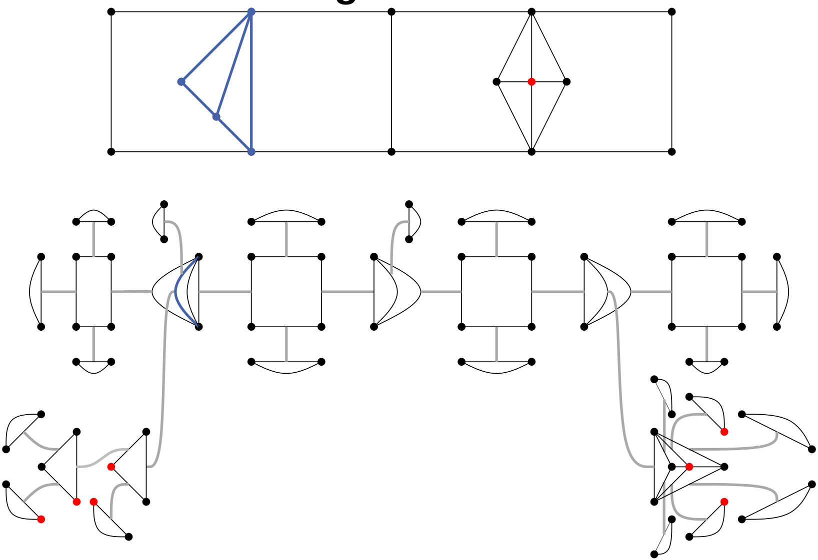
Algorithmus



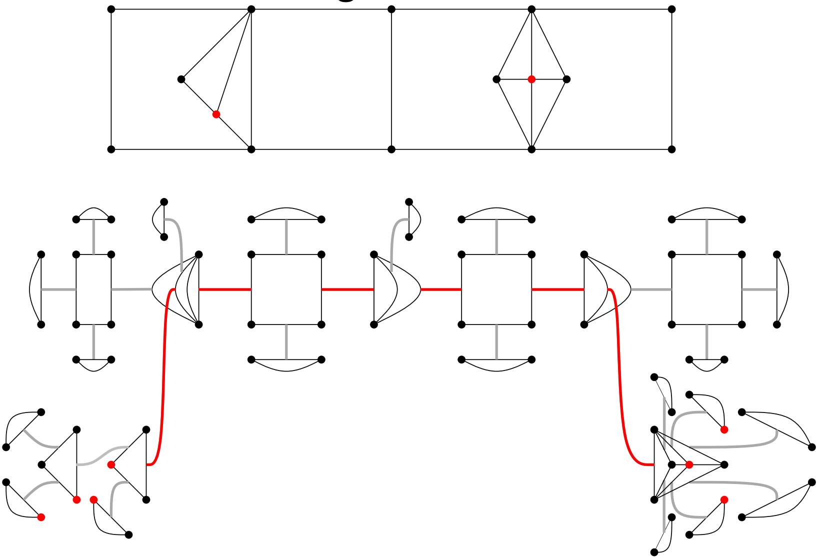
Algorithmus



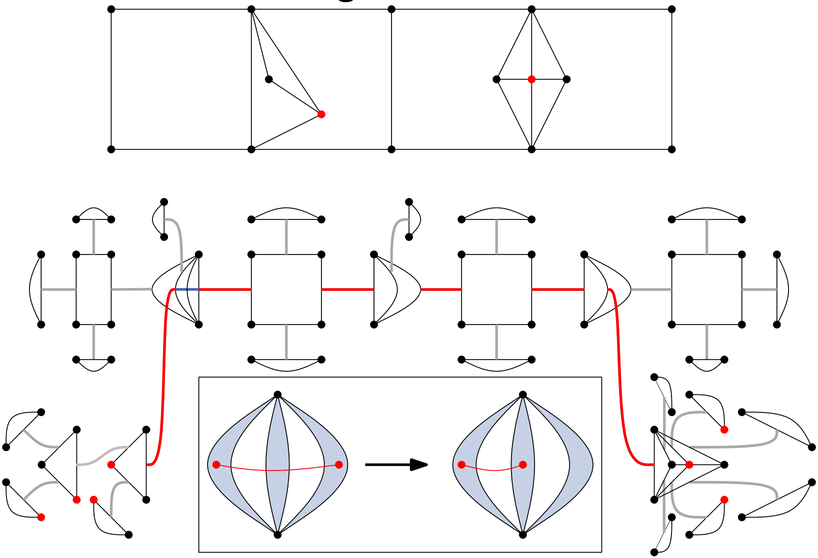
Algorithmus



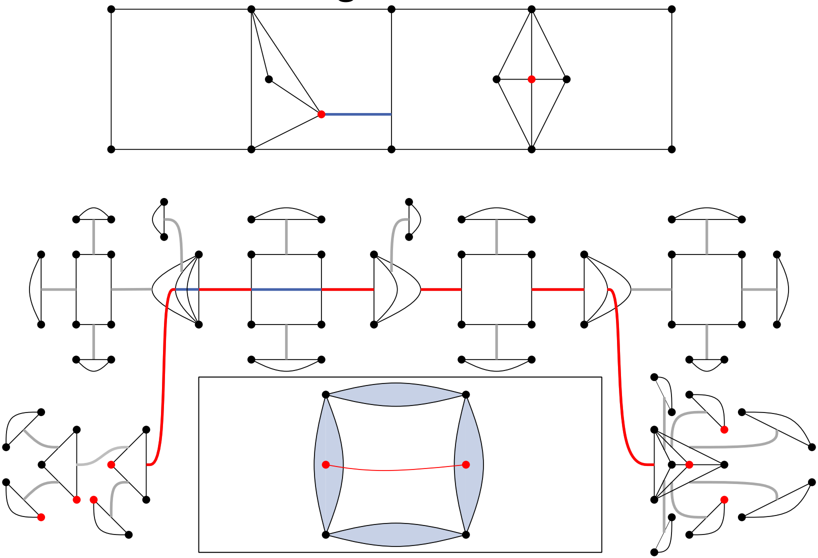
Algorithmus



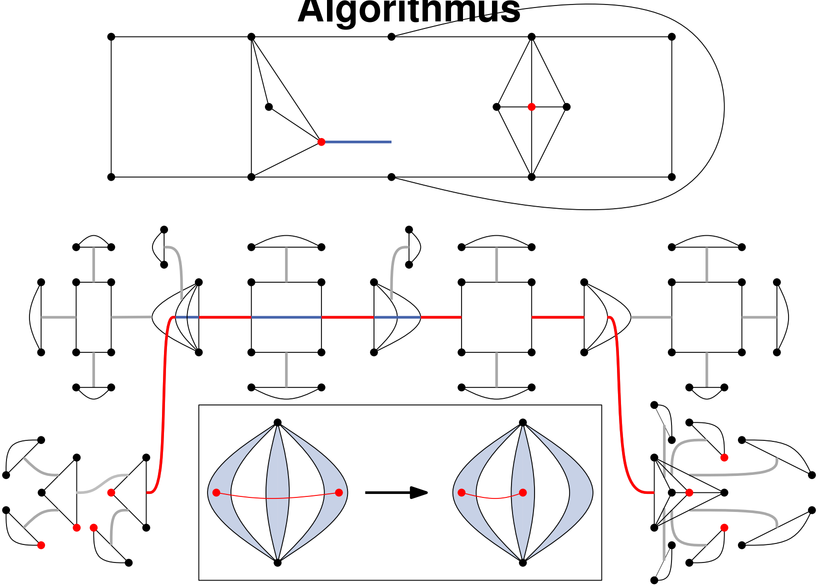
Algorithmus



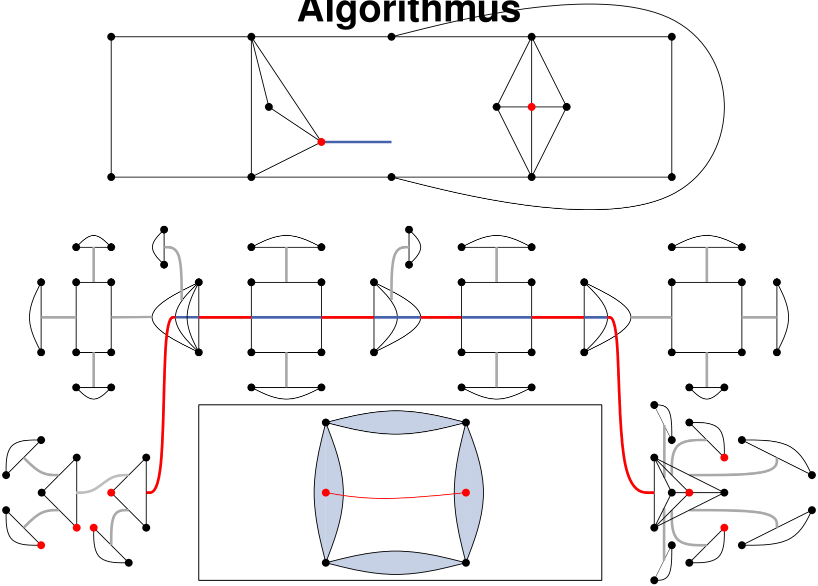
Algorithmus



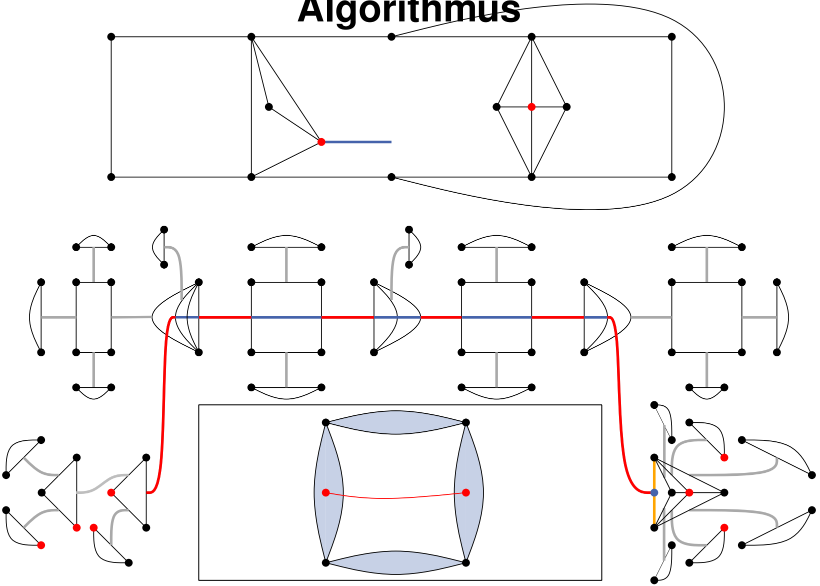
Algorithmus



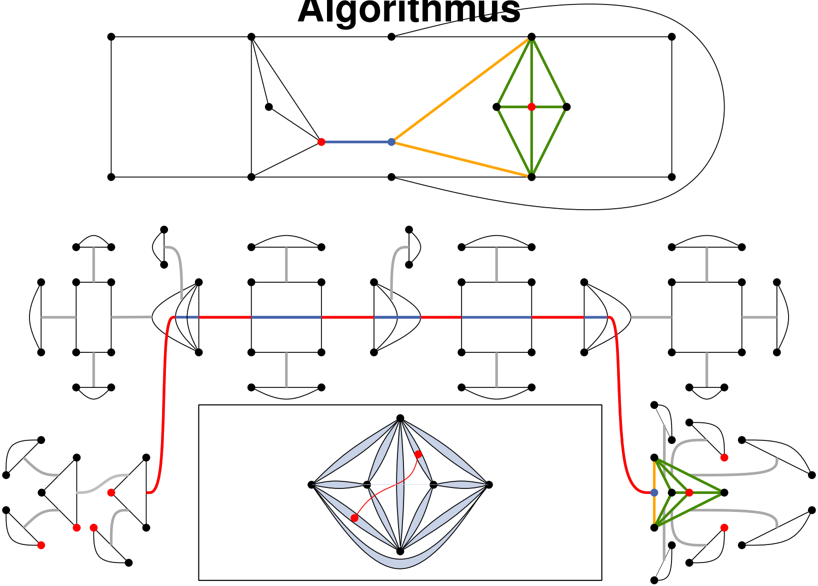
Algorithmus



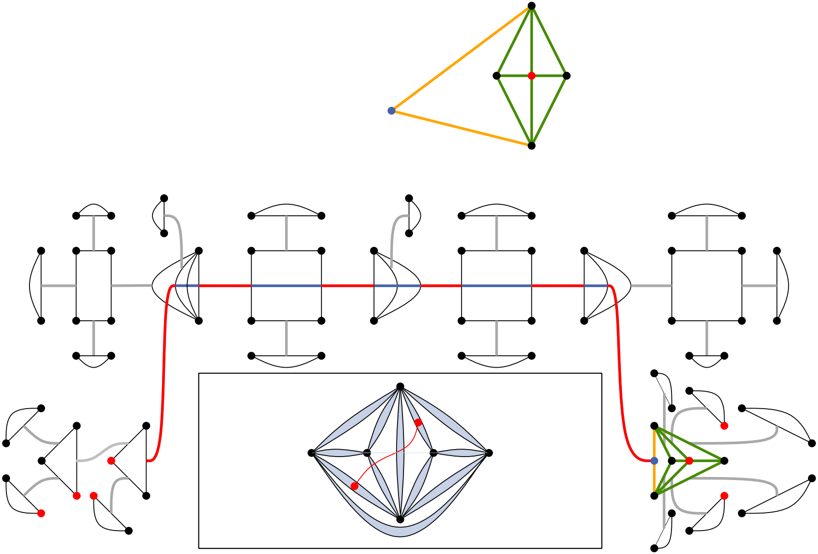
Algorithmus



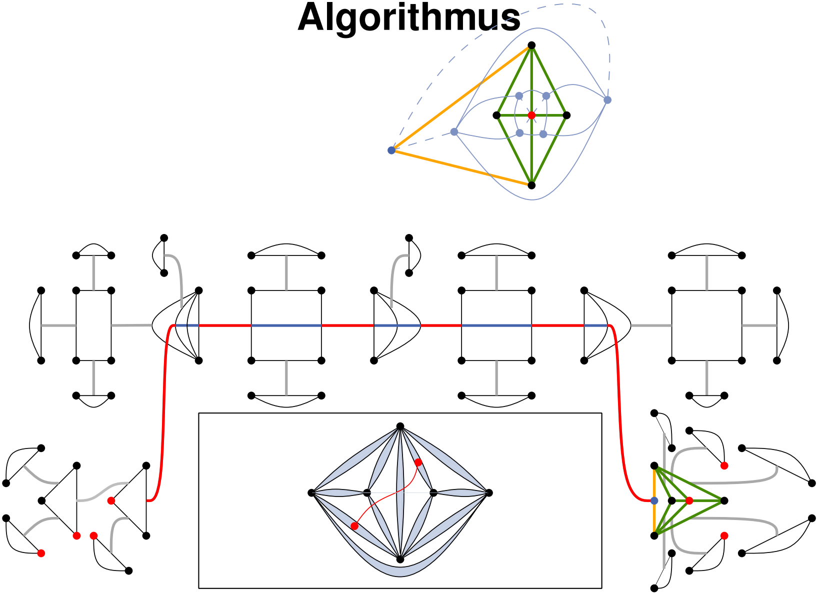
Algorithmus



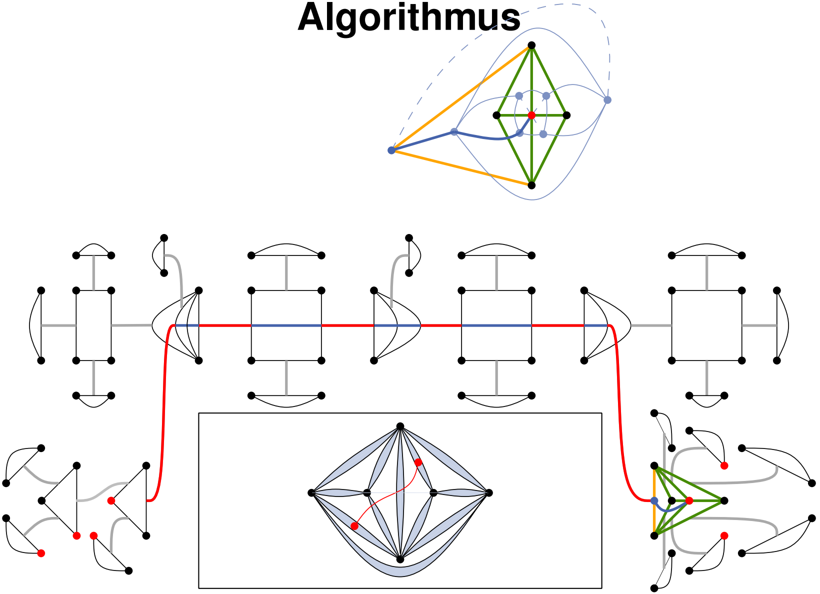
Algorithmus



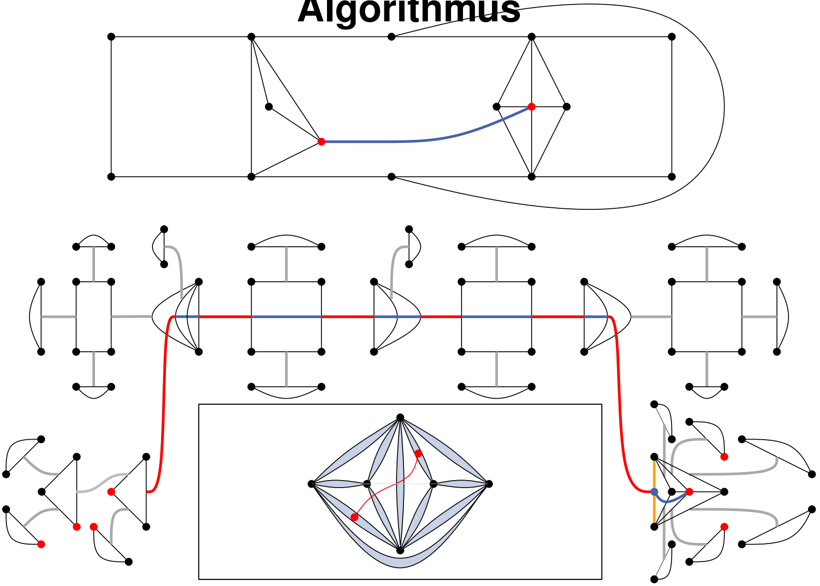
Algorithmus



Algorithmus



Algorithmus



Analyse

- Annahme: es existiert besserer Einfüge-Pfad π

Analyse

- Annahme: es existiert besserer Einfüge-Pfad π
- dann ex. R-Knoten r, in welchem π weniger Kreuzungen hat

Analyse

- Annahme: es existiert besserer Einfüge-Pfad π
- dann ex. R-Knoten r, in welchem π weniger Kreuzungen hat
- betrachte induzierten Pfad π_S in Seklett S von r

Analyse

- Annahme: es existiert besserer Einfüge-Pfad π
- dann ex. R-Knoten r, in welchem π weniger Kreuzungen hat
- betrachte induzierten Pfad π_S in Seklett S von r
- sei e Skelett-Kante, die von π_S geschnitten wird

Analyse

- Annahme: es existiert besserer Einfüge-Pfad π
- dann ex. R-Knoten r, in welchem π weniger Kreuzungen hat
- betrachte induzierten Pfad π_S in Seklett S von r
- sei e Skelett-Kante, die von π_S geschnitten wird
- Anzahl Kreuzungen von Graph G_e und π ist unabhängig von Einbettung von G_e (Strukturelle Induktion auf SPQR-Baum)

Analyse

- Annahme: es existiert besserer Einfüge-Pfad π
- dann ex. R-Knoten r, in welchem π weniger Kreuzungen hat
- betrachte induzierten Pfad π_S in Seklett S von r
- sei e Skelett-Kante, die von π_S geschnitten wird
- Anzahl Kreuzungen von Graph G_e und π ist unabhängig von Einbettung von G_e (Strukturelle Induktion auf SPQR-Baum)
- in Laufzeit $\mathcal{O}(n)$ implementierbar

Ausblick

- Optimales Einfügen einer Kante approximiert $cr(H + e)$ mit Faktor $\Delta/2$
- Optimales Einfügen eines Knotens ist in P
- Optimales Einfügen mehrerer Kanten ist NP-schwer