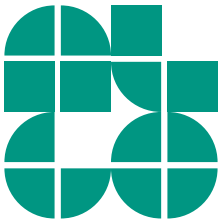
Algorithmen zur Visualisierung von Graphen

Übung 4: Lagenlayouts

INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Tamara Mchedlidze · **Martin Nöllenburg**

07.01.2015



Aufgabe 1 – Feedback Arc Set

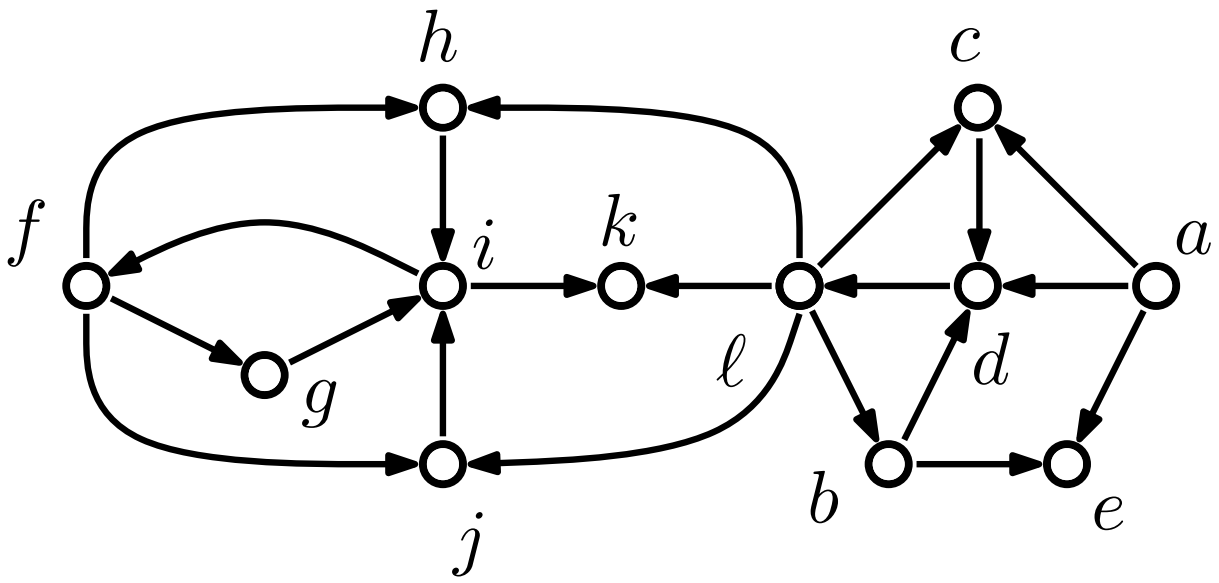
Minimum Feedback Arc Set: Sei $D = (V, A)$ ein gerichteter Graph. Bestimme eine Menge $A_f \subset A$ minimaler Kardinalität, so dass $D_f = (V, A \setminus A_f)$ azyklisch ist.

Minimum Feedback Set: Sei $D = (V, A)$ ein gerichteter Graph. Bestimme eine Menge $A_r \subset A$ minimaler Kardinalität, so dass $D_r = (V, A \setminus A_r \cup \mathrm{rev}(A_r))$ azyklisch ist. Dabei bezeichne $\mathrm{rev}(A)$ die Kantenmenge, die man erhält wenn die Richtung jeder Kante der Menge A invertiert wird.

Wir haben gesehen, dass jedes Feedback Set ein Feedback Arc Set ist, umgekehrt gilt dies im Allgemeinen jedoch nicht. Für minimale Mengen gilt die Äquivalenz: Zeigen Sie, dass die Lösungsmengen der beiden Probleme identisch sind.

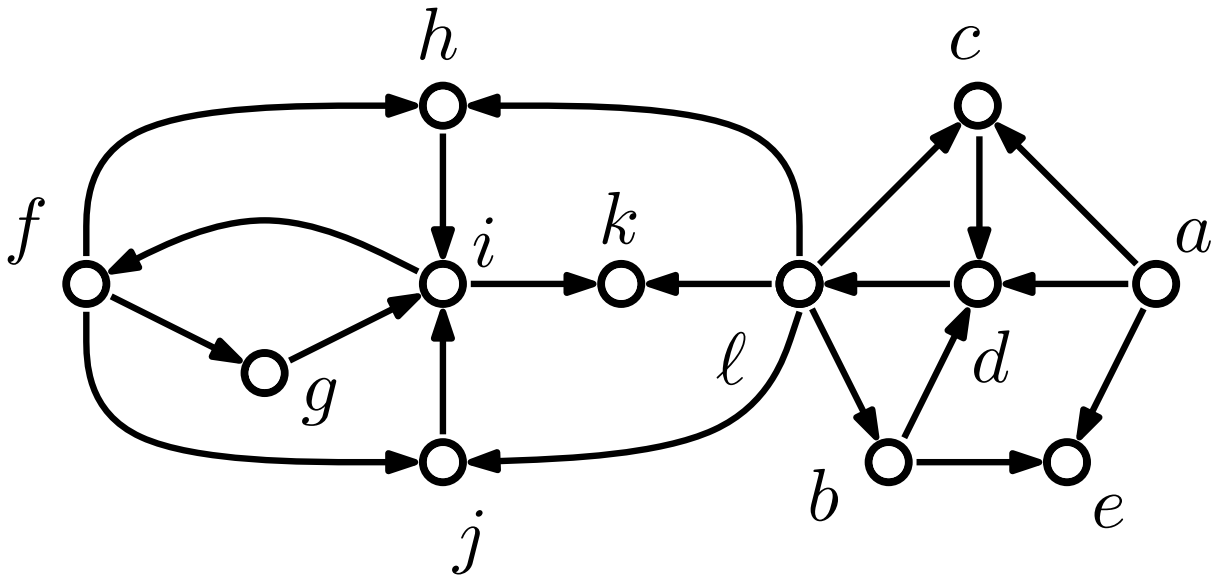
Aufgabe 2 – Lagenlayout

Führen Sie den in der Vorlesung vorgestellten Algorithmus zur Generierung von Lagenlayouts schrittweise (manuell und optimal) für den untenstehenden Graphen aus. Entfernen Sie dazu gerichtete Kreise indem Sie für eine möglichst kleine Menge an Kanten die Richtung umkehren, finden Sie eine Lagenzuordnung minimaler Höhe bei maximaler Breite 4 und ordnen sie die Knoten innerhalb der Lagen so an, dass die Anzahl an Kreuzungen minimal ist.



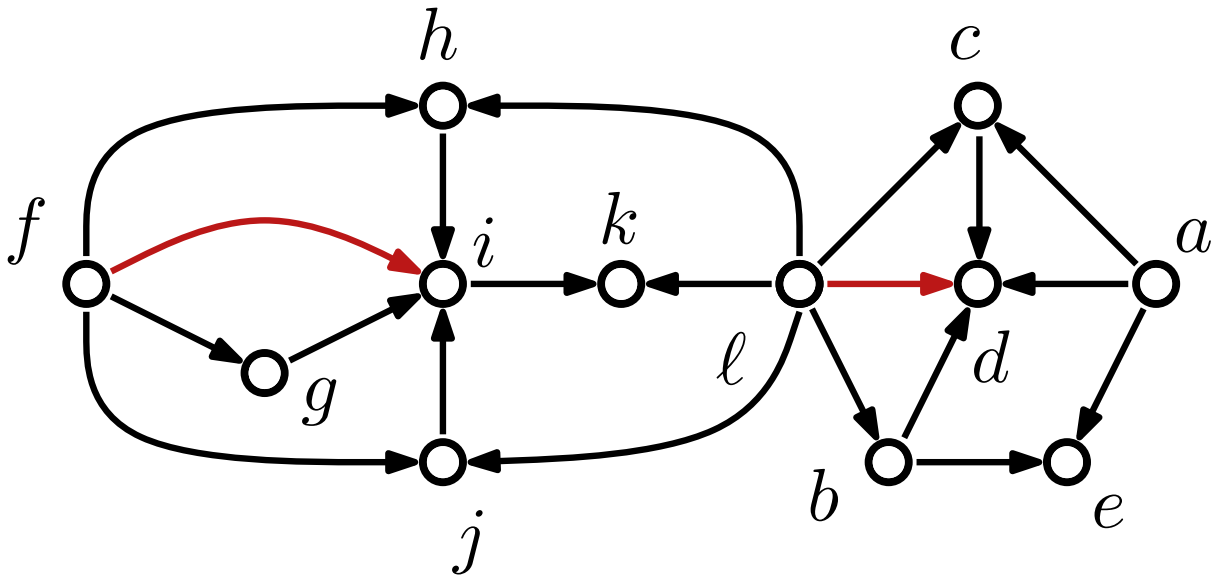
Lagenlayout – 1. FAS

Finde $E' \subseteq E$ sodass $G - E'$ azyklisch und $|E'|$ minimal (FAS)



Lagenlayout – 1. FAS

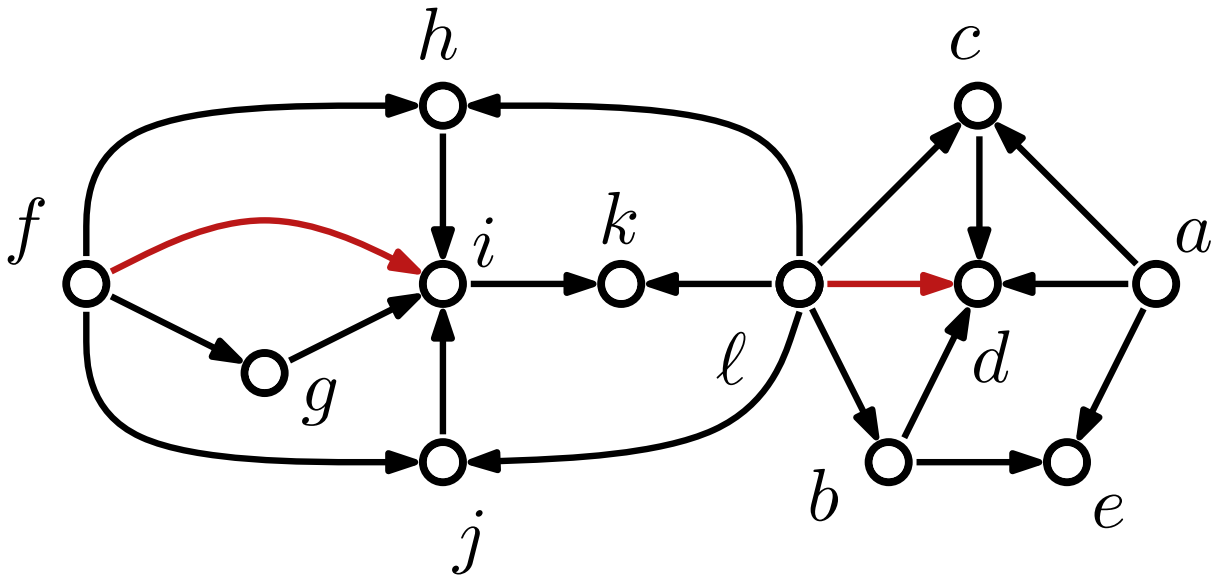
Finde $E' \subseteq E$ sodass $G - E'$ azyklisch und $|E'|$ minimal (FAS)



Lagenlayout – 1. FAS

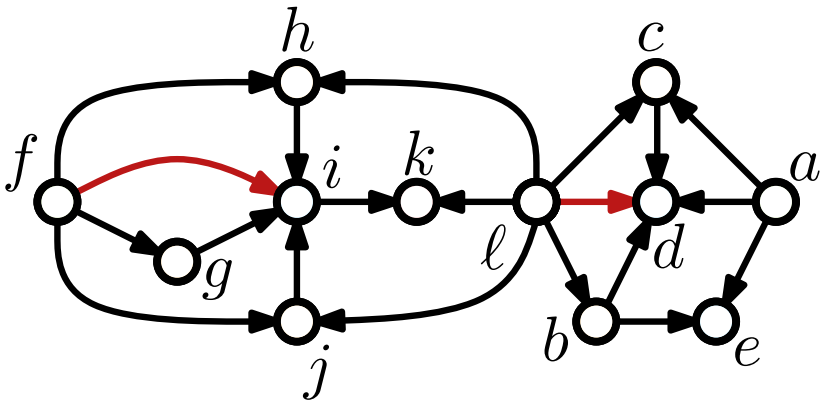
Finde $E' \subseteq E$ sodass $G - E'$ azyklisch und $|E'|$ minimal (FAS)

Kehre die Orientierung der Kanten in E' um



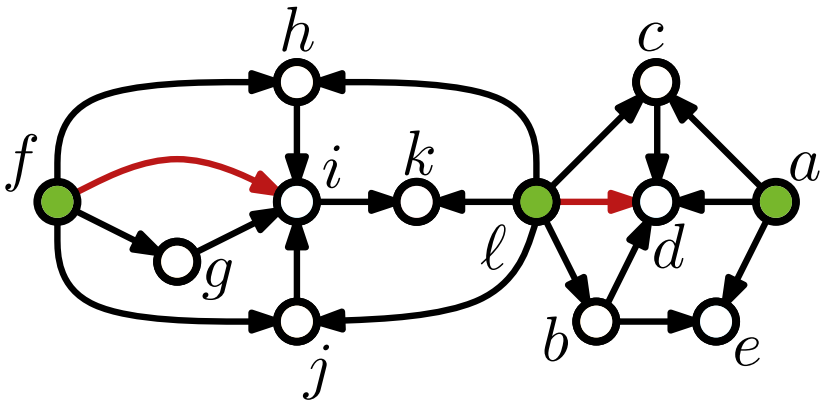
Lagenlayout – 2. Lagenzuordnung

Lagenzuordnung min. Höhe: iteratives Löschen von Quellen



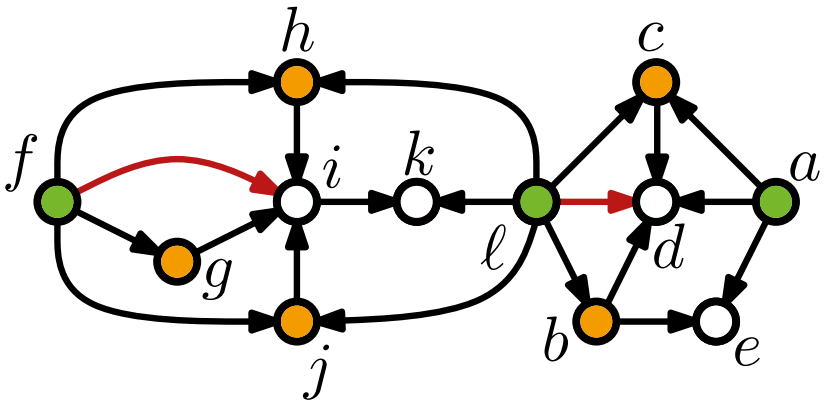
Lagenlayout – 2. Lagenzuordnung

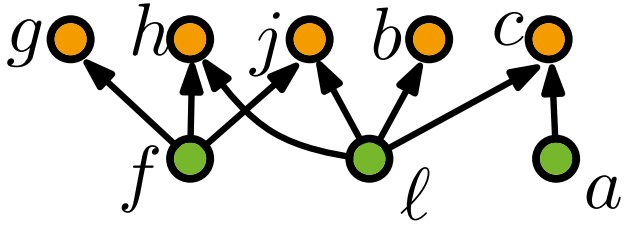
Lagenzuordnung min. Höhe: iteratives Löschen von Quellen



Lagenlayout – 2. Lagenzuordnung

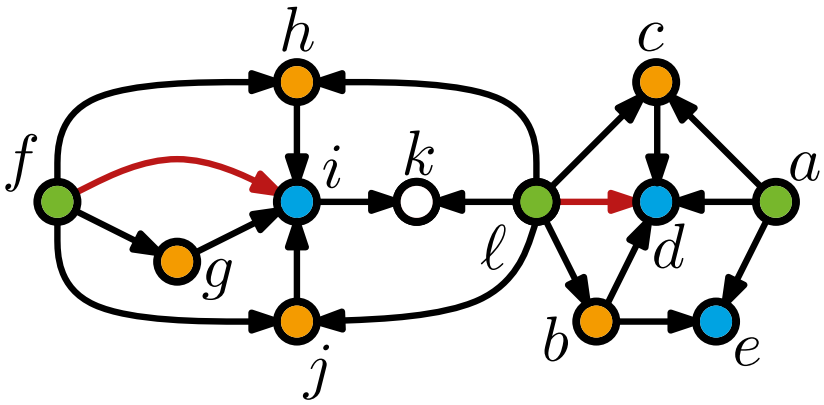
Lagenzuordnung min. Höhe: iteratives Löschen von Quellen

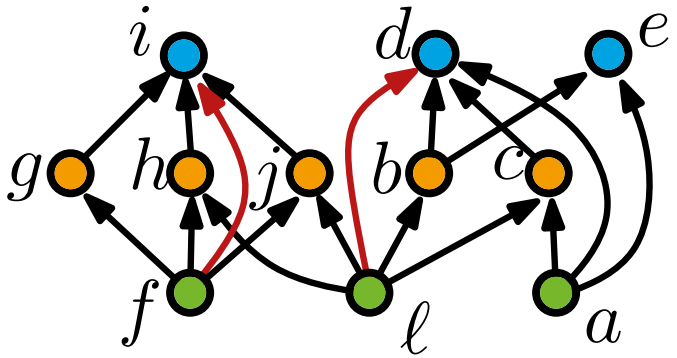




Lagenlayout – 2. Lagenzuordnung

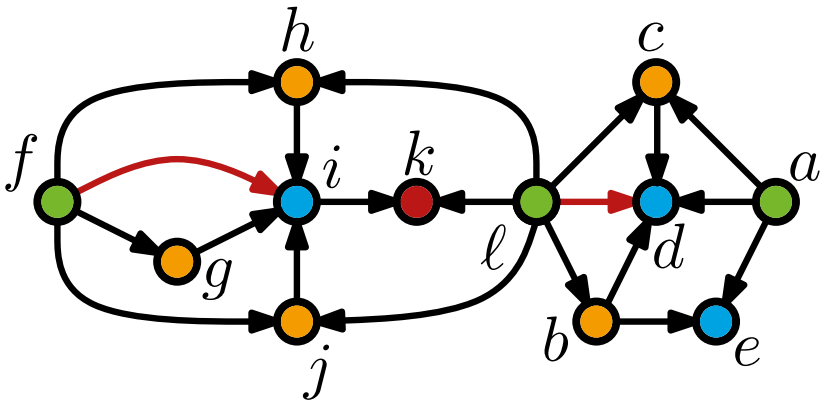
Lagenzuordnung min. Höhe: iteratives Löschen von Quellen

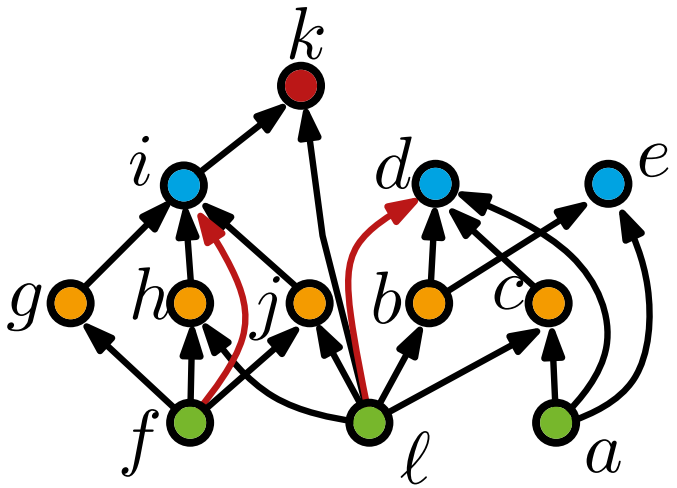




Lagenlayout – 2. Lagenzuordnung

Lagenzuordnung min. Höhe: iteratives Löschen von Quellen

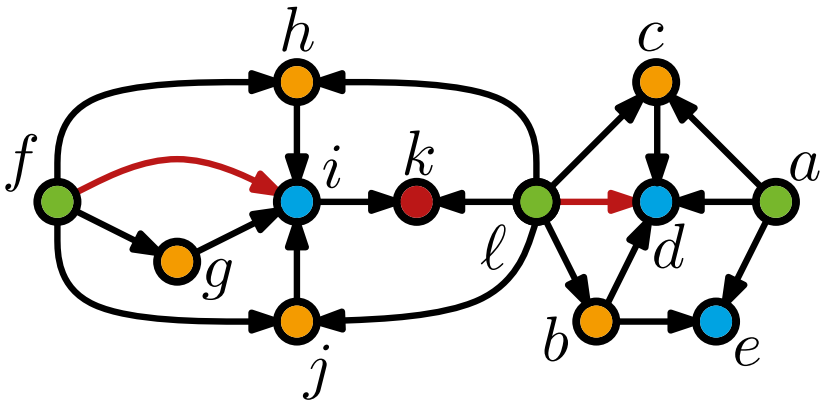


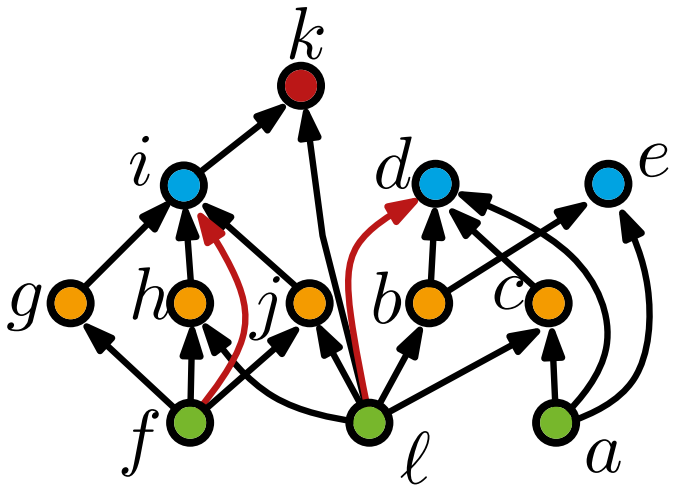


Lagenlayout – 2. Lagenzuordnung

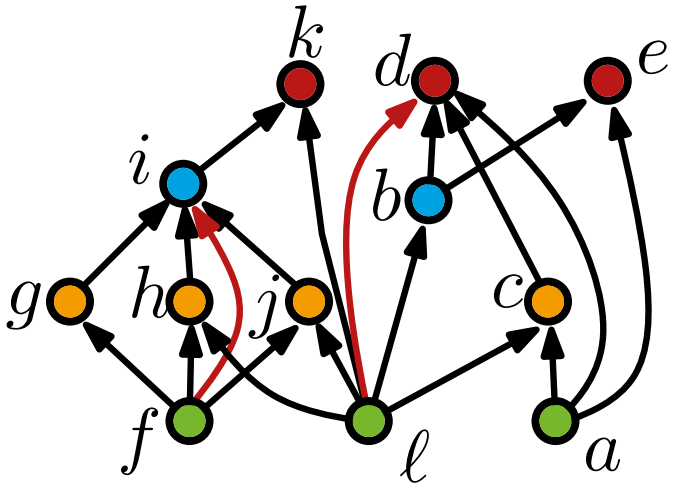
Karlsruhe Institute of Technology

Lagenzuordnung min. Höhe: iteratives Löschen von Quellen





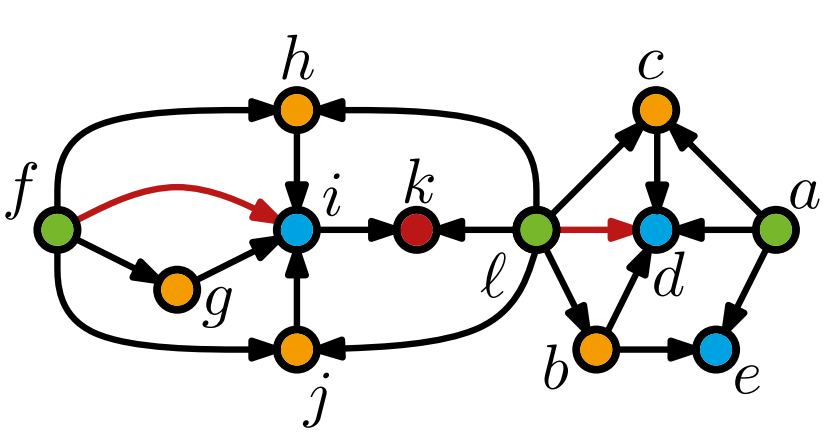
Min Höhe bei auf 4 beschränkter Breite: genaues hinschauen

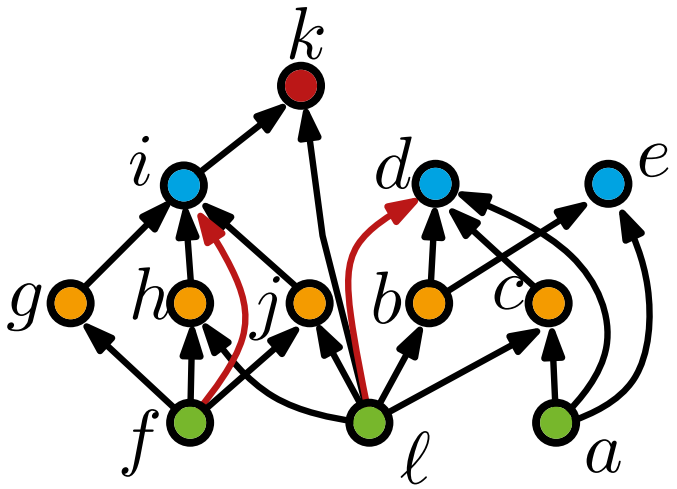


Lagenlayout – 2. Lagenzuordnung

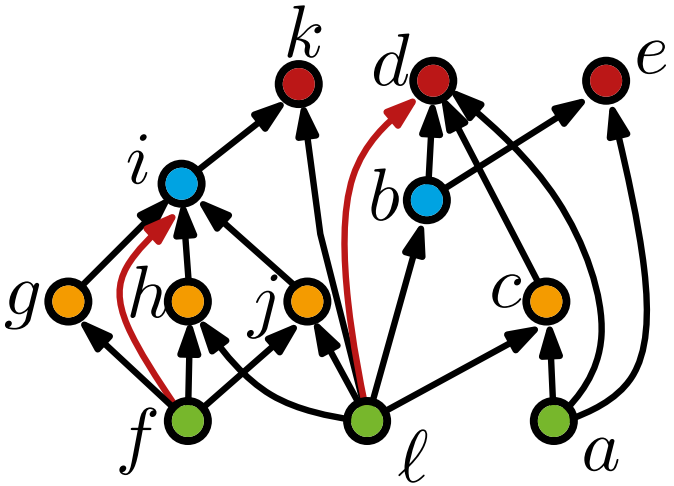
Karlsruhe Institute of Technology

Lagenzuordnung min. Höhe: iteratives Löschen von Quellen

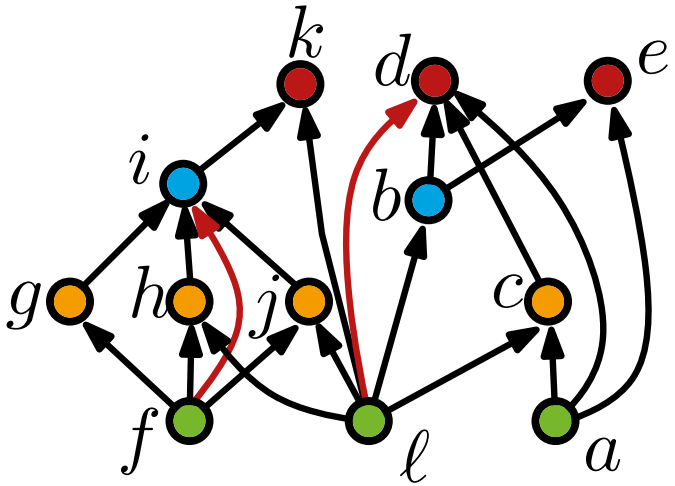




Min Höhe bei auf 4 beschränkter Breite: genaues hinschauen



Kreuzungen minimieren



Aufgabe 3 – Fehlstände Zählen

(a) Sei $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ eine Permutation. Ein Paar (i, j) mit $1 \leq i < j \leq n$ heißt Inversion, wenn $\pi(i) > \pi(j)$. Entwerfen Sie einen Algorithmus, der die Anzahl der Inversionen einer Permutation von n Elementen in $O(n \log n)$ Zeit berechnet.
Hinweis: Denken Sie an Sortieralgorithmen wie Mergesort.

Aufgabe 3 – Fehlstände Zählen

(a) Sei $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ eine Permutation. Ein Paar (i, j) mit $1 \leq i < j \leq n$ heißt Inversion, wenn $\pi(i) > \pi(j)$. Entwerfen Sie einen Algorithmus, der die Anzahl der Inversionen einer Permutation von n Elementen in $O(n \log n)$ Zeit berechnet.
Hinweis: Denken Sie an Sortieralgorithmen wie Mergesort.

(b) Gegeben sei ein einfacher, bipartiter Graph $G = (V, E)$, dessen Knoten gemäß der Bipartition auf zwei parallele Geraden verteilt sind. Die Knoten seien disjunkt und die Kanten geradlinig gezeichnet. Entwerfen Sie einen Algorithmus, der die Anzahl der Kreuzungen in $O(|E| \log |V|)$ Zeit bestimmt. Begründen Sie, warum es nicht möglich ist, in dieser worst-case-Laufzeit alle Kreuzungen (Paare betroffener Kanten) *auszugeben*.

Aufgabe 3 – Lösungsskizze

- modifiziere divide & conquer mergesort-Algorithmus
- zähle beim Sortieren die übersprungenen Elemente mit – genau das sind die Inversionen

- zwei Kanten (u, v) und (w, z) schneiden sich gdw. $u < w$ und $v > z$
- bezeichne Knoten auf Lage 1 als $a, b, c, \ldots$ und Knoten auf Lage 2 als $1, 2, 3, \ldots$
- bilde Folge der Kanten aus Sicht von Lage 2, d.h. $a1, c1, d1, a2, b2, b3, d3, e3, \ldots$
- wende Algorithmus aus (a) an und sortiere lexikographisch
- leicht zu sehen, dass es Graphen mit $O(|E|^2)$ Kreuzungen gibt

Aufgabe 4 – Kreuzungen bei Lagenlayouts

Karlsruhe Institute of Technology

Zeigen Sie, dass die Baryzenter-Heuristik zur einseitigen Kreuzungsreduktion die optimale Lösung liefert, falls diese keine Kreuzung enthält.

Aufgabe 4 – Kreuzungen bei Lagenlayouts

Karlsruhe Institute of Technology

Zeigen Sie, dass die Baryzenter-Heuristik zur einseitigen Kreuzungsreduktion die optimale Lösung liefert, falls diese keine Kreuzung enthält.

Lösungsskizze

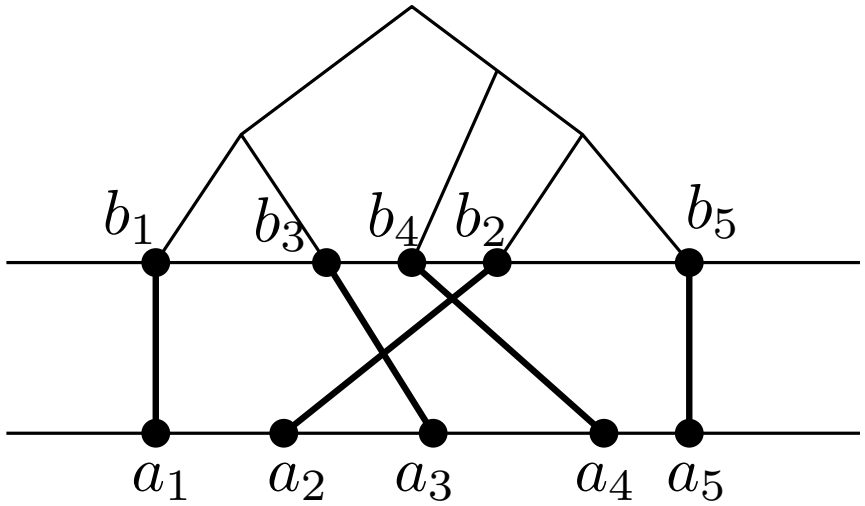
- kreuzungsfrei gdw. alle Nachbarn auf fester Lage konsekutiv sind
- Nachbarn bilden im Inneren disjunkte Intervalle, also getrennte Schwerpunkte
- Schwerpunkte nur identisch bei Grad-1 Knoten $\rightarrow$ Reihenfolge beliebig

Aufgabe 5 – Tanglegram Layout

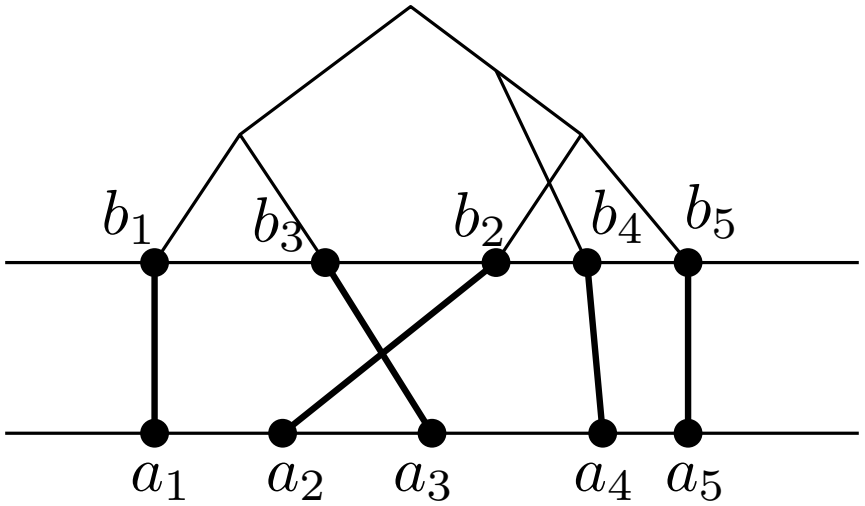
One Tree Crossing Minimization (OTCM): Gegeben sei ein perfektes Matching M auf $2n$ Knoten $A = \{a_1, \ldots, a_n\}$ und $B = \{b_1, \ldots, b_n\}$, mit Matchingkanten $E = \{(a_i, b_i) \mid i = 1, \ldots, n\}$. Bestimme für eine feste Permutation π_1 der Menge A eine Permutation π_2 der Menge B, die die Anzahl der Kreuzungen von M minimiert, wobei A und B geordnet nach π_1 und π_2 auf zwei horizontalen Geraden platziert sind. Dabei muss π_2 *kompatibel* zu einem Binärbaum T mit Blattmenge B sein.

Die Permutation π_2 heißt dabei *kompatibel* zu T, wenn eine planare Zeichnung von T in der Halbebene $y \geq 0$ existiert, die die Blätter in der Reihenfolge π_2 auf der x-Achse platziert und alle inneren Knoten im Bereich $y > 0$.

Aufgabe 5 – Tanglegram Layout



kompatibel



nicht kompatibel

Aufgabe 5 – Tanglegram Layout

(a) Zeigen Sie, dass das Problem OTCM in polynomieller Zeit gelöst werden kann. Welchen asymptotischen Zeitaufwand hat Ihr Algorithmus?

Hinweis: Verwenden Sie dynamische Programmierung.

Aufgabe 5 – Tanglegram Layout

(a) Zeigen Sie, dass das Problem OTCM in polynomieller Zeit gelöst werden kann. Welchen asymptotischen Zeitaufwand hat Ihr Algorithmus?

Hinweis: Verwenden Sie dynamische Programmierung.

(b) Angenommen beide Permutationen π_1 und π_2 sind variabel und jeweils durch Kompatibilität mit zwei Binärbäumen T_1 und T_2 auf den Blättern A bzw. B eingeschränkt. Zeigen Sie, dass in polynomieller Zeit entschieden werden kann, ob das Matching kreuzungsfrei gezeichnet werden kann.

Hinweis: Verwenden Sie, dass für einen gerichteten Graphen mit einer einzigen Quelle in linearer Zeit geprüft werden kann, ob er aufwärtsplanar ist. [Bertolazzi et al. 1998]

Aufgabe 5 – Lösungsskizze

- definiere Tabelle C indiziert durch Baumknoten mit $C[v] =$ min. # Kreuzungen des Teilbaums T_v
- initialisiere $C[v] = 0$ für Blätter
- sonst: $C[v] = C[u] + C[w] + \min\{cr(u,w), cr(w,u)\}$, wobei u und w die Kinder von v sind und $cr(u,w)$ die Anzahl Kreuzungen von Matchingkanten von T_u mit Matchingkanten von T_v falls u links von v liegt
- Zeitaufwand $O(n^2)$ bei Berechnung der cr-Werte in jeweils $O(n)$
- Berechnung auch in $O(n \log n)$ möglich

(s. [Fernau, Kaufmann, Poths, Comparing Trees via Crossing Minimization, 2010])

- erzeuge DAG aus T_1 und T_2 gerichtet von der Wurzel von T_1 zur Wurzel von T_2