

# Algorithmen zur Visualisierung von Graphen

## Lagenlayouts – Teil 2

INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

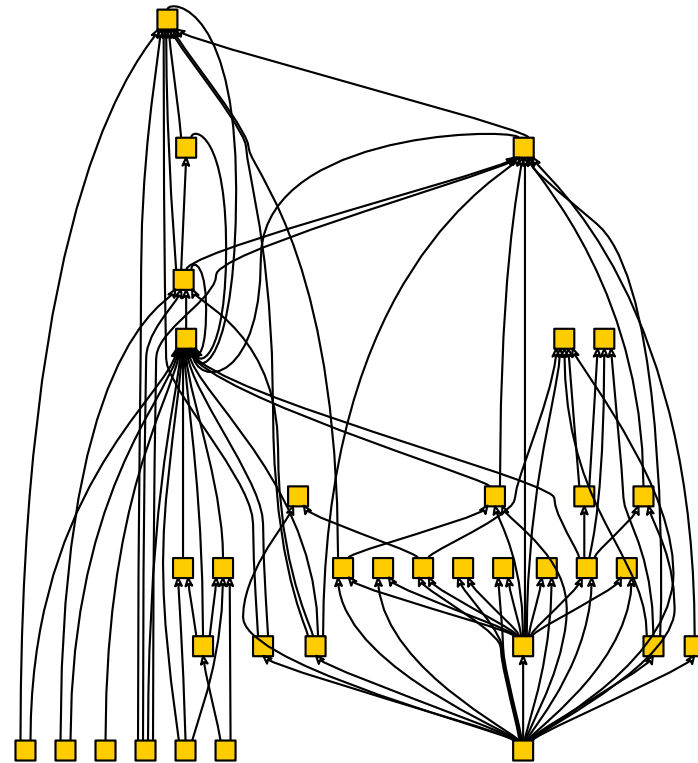
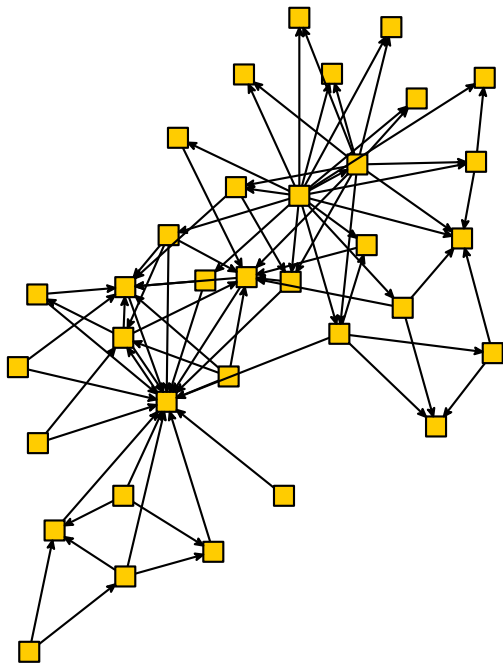
Tamara Mchedlidze · **Martin Nöllenburg**  
21.01.2014



# Lagenlayouts

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

**Ges.:** Zeichnung von  $D$ , die die Hierarchie verdeutlicht



# Lagenlayouts

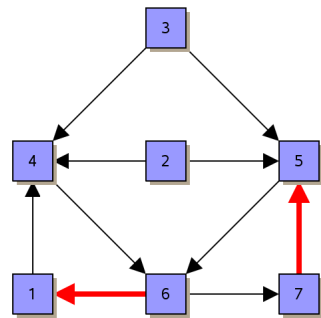
**Geg.:** gerichteter Graph  $D = (V, A)$

**Ges.:** Zeichnung von  $D$ , die die Hierarchie verdeutlicht

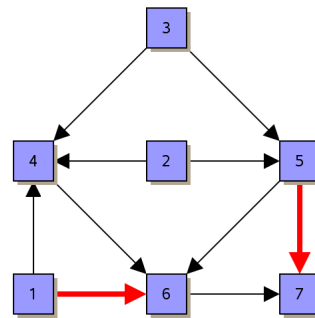
## Kriterien:

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

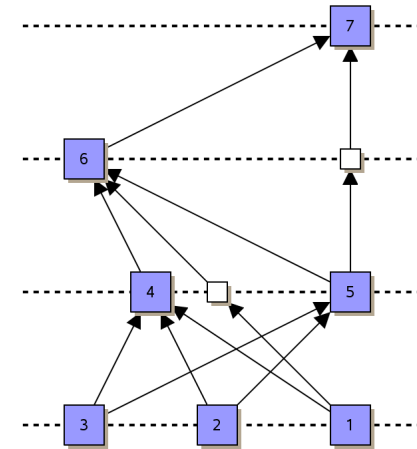
# Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



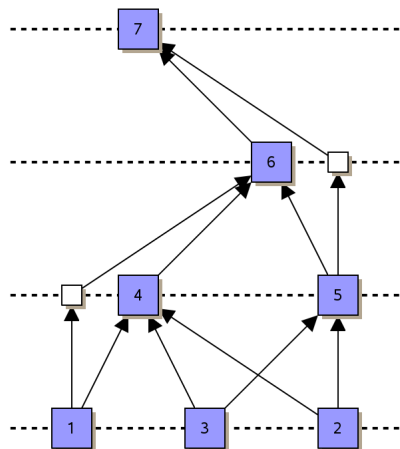
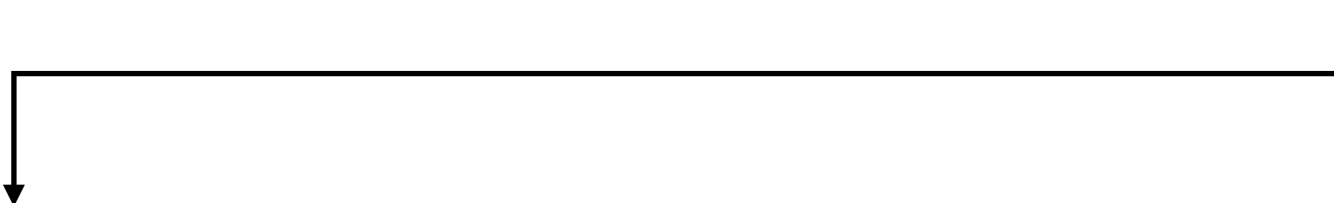
Eingabe



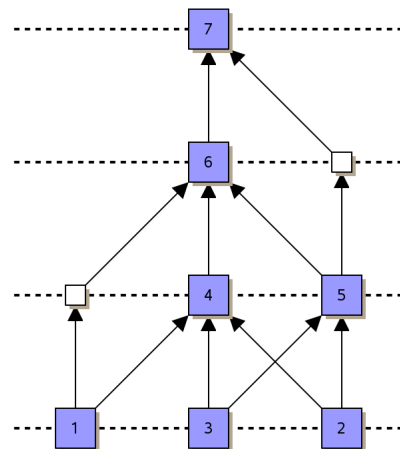
Kreise brechen



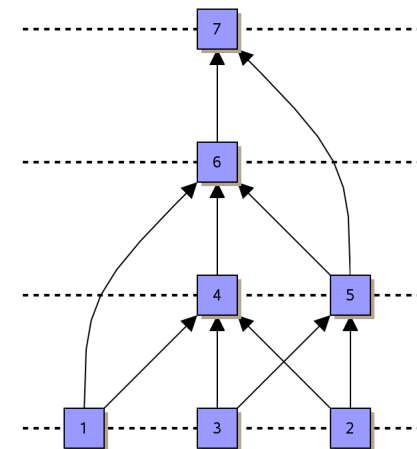
Lagenzuordnung



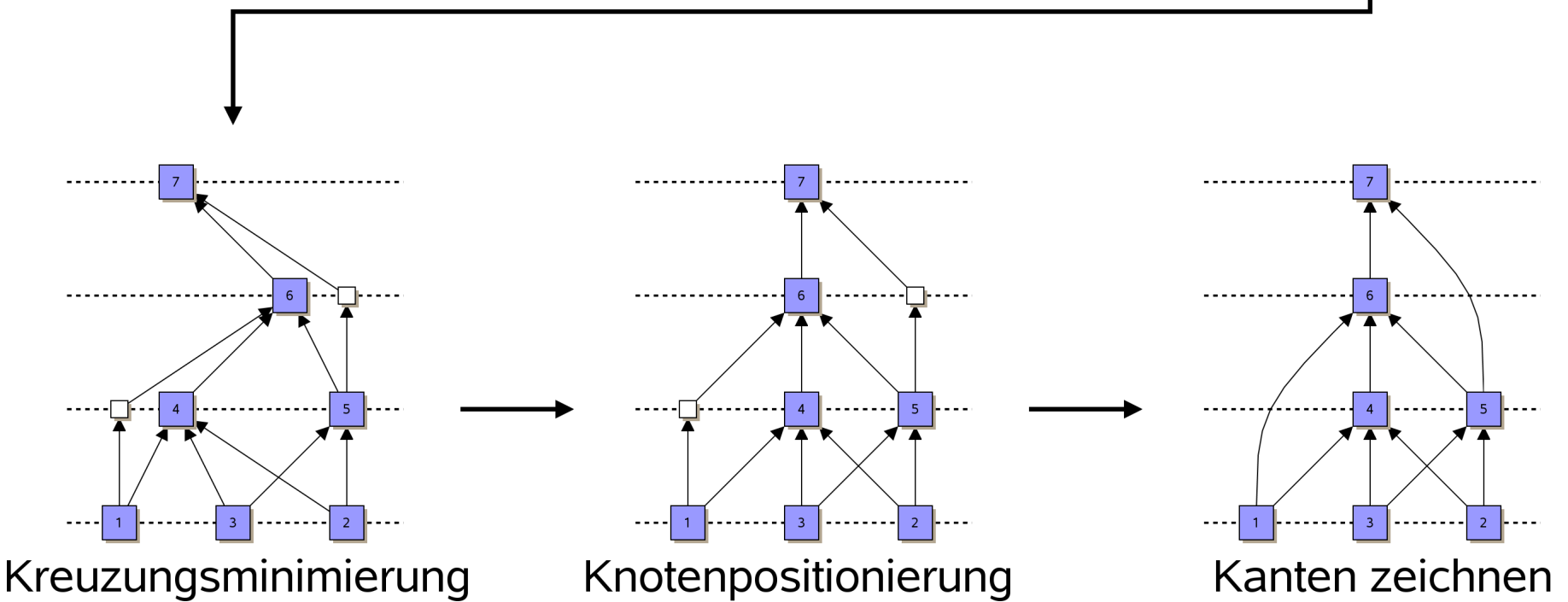
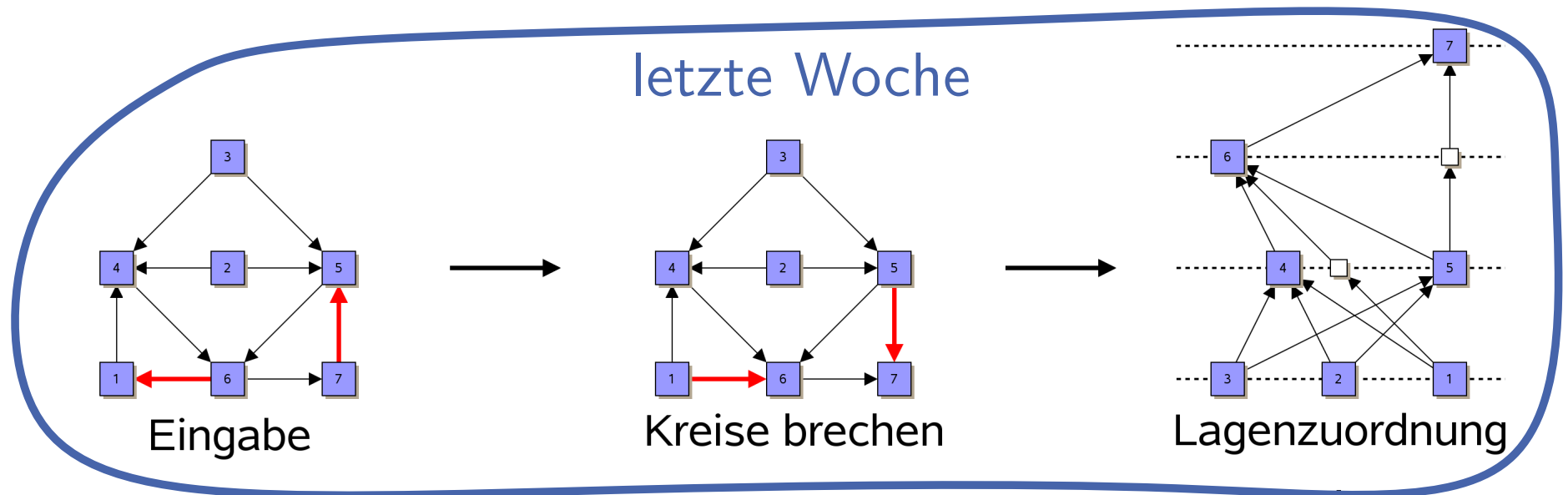
Kreuzungsminimierung



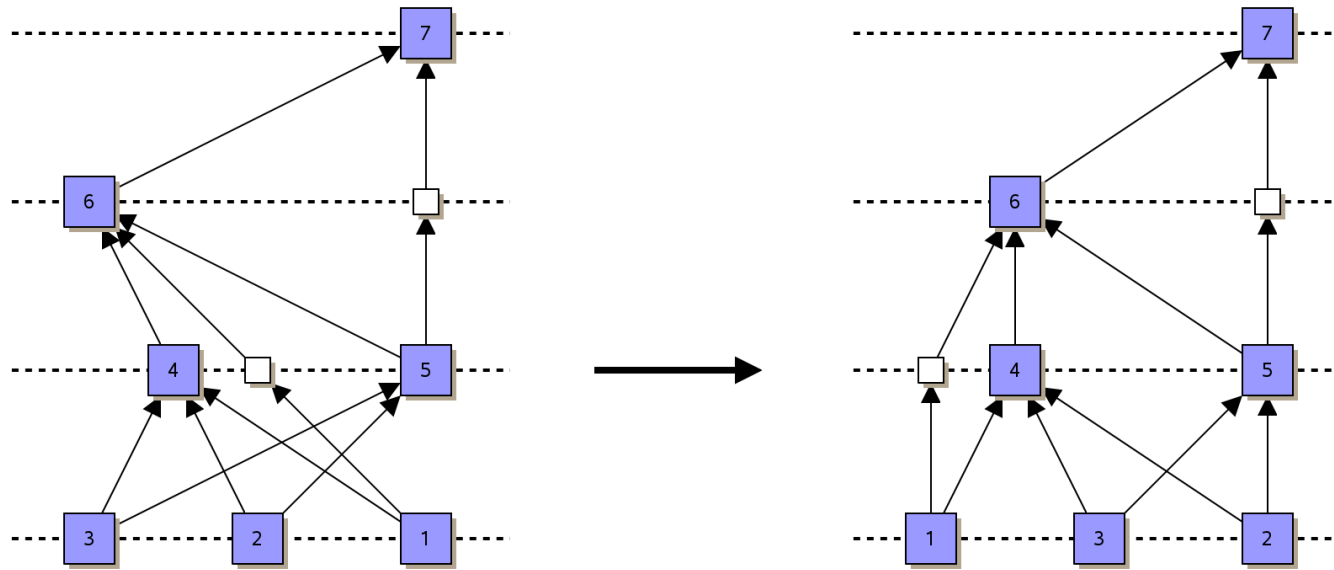
Knotenpositionierung



Kanten zeichnen



# Schritt 3: Kreuzungsminimierung



Wie würden Sie vorgehen?

# Problemstellung

**Geg.:** DAG  $D = (V, A)$ , Knoten partitioniert in disjunkte Lagen

**Ges.:** Ordnung der Knoten in jeder Lage, so dass die Anzahl Kantenkreuzungen minimal ist

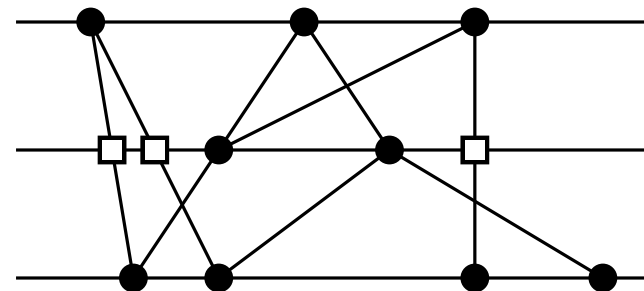
# Problemstellung

**Geg.:** DAG  $D = (V, A)$ , Knoten partitioniert in disjunkte Lagen

**Ges.:** Ordnung der Knoten in jeder Lage, so dass die Anzahl Kantenkreuzungen minimal ist

## Eigenschaften

- Problem ist NP-schwer schon für zwei Lagen  
(BIPARTITE CROSSING NUMBER [Garey, Johnson '83])
- kaum Ansätze über mehrere Lagen gleichzeitig
- meist iteratives Optimieren für je zwei benachbarte Lagen
- dazu: füge in jeder Lage Dummyknoten für alle Kanten ein, die diese Lage überspringen





# Einseitige Kreuzungsminimierung (OSCM)

**Geg.:** 2-Lagen-Graph  $G = (L_1, L_2, E)$  und  
Knotenordnung  $x_1$  von  $L_1$

**Ges.:** Knotenordnung  $x_2$  von  $L_2$ , so dass die Anzahl  
Kreuzungen von Kanten in  $E$  minimal ist

# Einseitige Kreuzungsminimierung (OSCM)

**Geg.:** 2-Lagen-Graph  $G = (L_1, L_2, E)$  und  
Knotenordnung  $x_1$  von  $L_1$

**Ges.:** Knotenordnung  $x_2$  von  $L_2$ , so dass die Anzahl  
Kreuzungen von Kanten in  $E$  minimal ist

## Beobachtung:

- Anzahl Kreuzungen einer 2-Lagen-Zeichnung von  $G$  hängt nur von  $x_1$  und  $x_2$  ab, nicht von tatsächlichen Positionen
- für  $u, v \in L_2$  hängt Anzahl Kreuzungen inzidenter Kanten nur von  $x_2(u) < x_2(v)$  oder  $x_2(v) < x_2(u)$  ab

# Einseitige Kreuzungsminimierung (OSCM)

**Geg.:** 2-Lagen-Graph  $G = (L_1, L_2, E)$  und  
Knotenordnung  $x_1$  von  $L_1$

**Ges.:** Knotenordnung  $x_2$  von  $L_2$ , so dass die Anzahl  
Kreuzungen von Kanten in  $E$  minimal ist

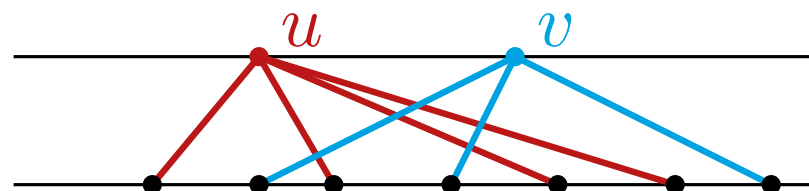
## Beobachtung:

- Anzahl Kreuzungen einer 2-Lagen-Zeichnung von  $G$  hängt nur von  $x_1$  und  $x_2$  ab, nicht von tatsächlichen Positionen
- für  $u, v \in L_2$  hängt Anzahl Kreuzungen inzidenter Kanten nur von  $x_2(u) < x_2(v)$  oder  $x_2(v) < x_2(u)$  ab

## Def: Kreuzungszahl

$$c_{uv} := |\{(uw, vz) \mid w \in N(u), z \in N(v), x_1(z) < x_1(w)\}|$$

für  $x_2(u) < x_2(v)$



$$c_{uv} = 5$$
$$c_{vu} = 7$$

# Einseitige Kreuzungsminimierung (OSCM)

**Geg.:** 2-Lagen-Graph  $G = (L_1, L_2, E)$  und  
Knotenordnung  $x_1$  von  $L_1$

**Ges.:** Knotenordnung  $x_2$  von  $L_2$ , so dass die Anzahl  
Kreuzungen von Kanten in  $E$  minimal ist

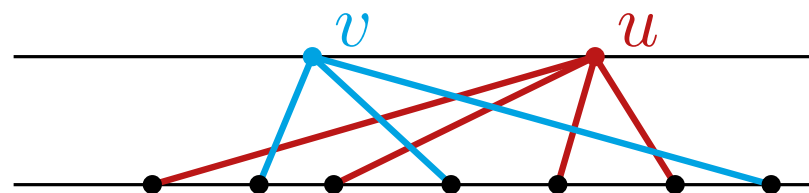
## Beobachtung:

- Anzahl Kreuzungen einer 2-Lagen-Zeichnung von  $G$  hängt nur von  $x_1$  und  $x_2$  ab, nicht von tatsächlichen Positionen
- für  $u, v \in L_2$  hängt Anzahl Kreuzungen inzidenter Kanten nur von  $x_2(u) < x_2(v)$  oder  $x_2(v) < x_2(u)$  ab

## Def: Kreuzungszahl

$$c_{uv} := |\{(uw, vz) \mid w \in N(u), z \in N(v), x_1(z) < x_1(w)\}|$$

für  $x_2(u) < x_2(v)$



$$c_{uv} = 5$$

$$c_{vu} = 7$$

**Def:** Kreuzungszahl für  $G$  mit Ordnungen  $x_1$  und  $x_2$  sei  
 $cr(G, x_1, x_2)$ ;  
für festes  $x_1$  sei  $opt(G, x_1) = \min_{x_2} cr(G, x_1, x_2)$

**Lemma 1:** Es gelten folgende Eigenschaften:

- $cr(G, x_1, x_2) = \sum_{x_2(u) < x_2(v)} c_{uv}$
- $opt(G, x_1) \geq \sum_{u,v} \min\{c_{uv}, c_{vu}\}$

**Def:** Kreuzungszahl für  $G$  mit Ordnungen  $x_1$  und  $x_2$  sei  $cr(G, x_1, x_2)$ ;  
für festes  $x_1$  sei  $opt(G, x_1) = \min_{x_2} cr(G, x_1, x_2)$

**Lemma 1:** Es gelten folgende Eigenschaften:

- $cr(G, x_1, x_2) = \sum_{x_2(u) < x_2(v)} c_{uv}$
- $opt(G, x_1) \geq \sum_{u,v} \min\{c_{uv}, c_{vu}\}$

Effizientes Berechnen von  $cr(G, x_1, x_2)$  s. Übungsblatt

# Iterative Kreuzungsminimierung

Sei  $G = (V, E)$  ein DAG mit Lagenzuordnung  $L_1, \dots, L_h$ .

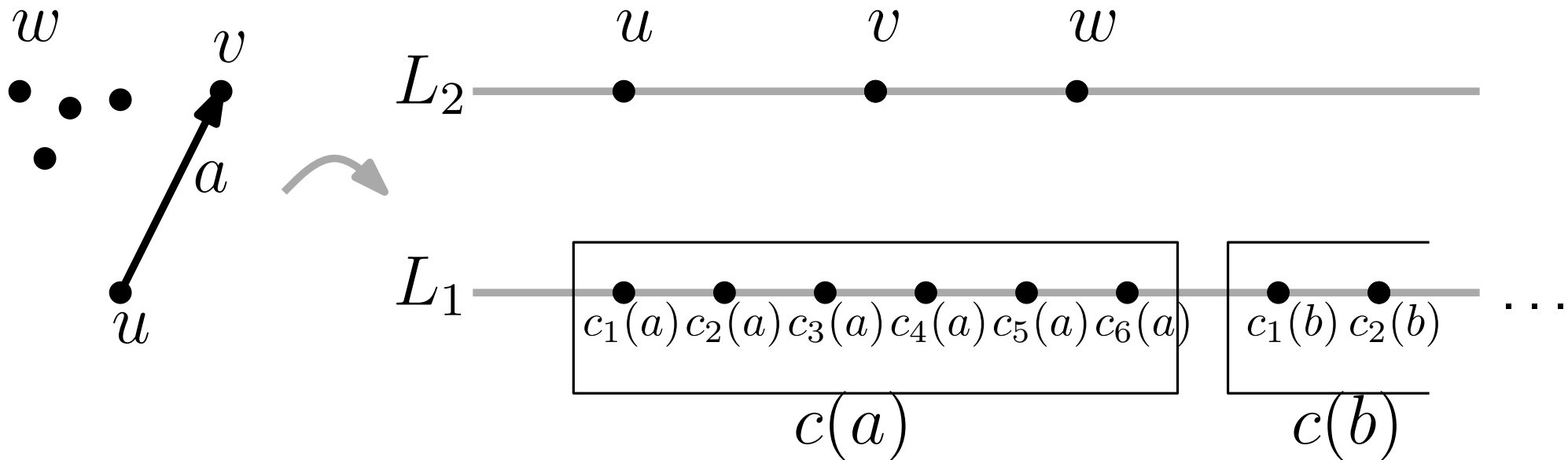
- (1) berechne zufällige Ordnung  $x_1$  für Lage  $L_1$
- (2) für  $i = 1, \dots, h - 1$  betrachte Lagen  $L_i$  und  $L_{i+1}$  und minimiere  $cr(G, x_i, x_{i+1})$  bei festem  $x_i$  ( $\rightarrow$  **OSCM**)
- (3) für  $i = h - 1, \dots, 1$  betrachte Lagen  $L_{i+1}$  und  $L_i$  und minimiere  $cr(G, x_i, x_{i+1})$  bei festem  $x_{i+1}$  ( $\rightarrow$  **OSCM**)
- (4) wiederhole (2) und (3) bis keine weitere Verbesserung
- (5) wiederhole ggf. Schritte (1)–(4) mit anderem  $x_1$
- (6) gib beste gefundene Lösung aus

**Satz 1:** Das einseitige Kreuzungsminimierungsproblem (OSCM) ist NP-schwer.

**Beweisskizze:** Reduktion von Feedback Arc Set

$$D = (V, A)$$

$$B = (W, E)$$



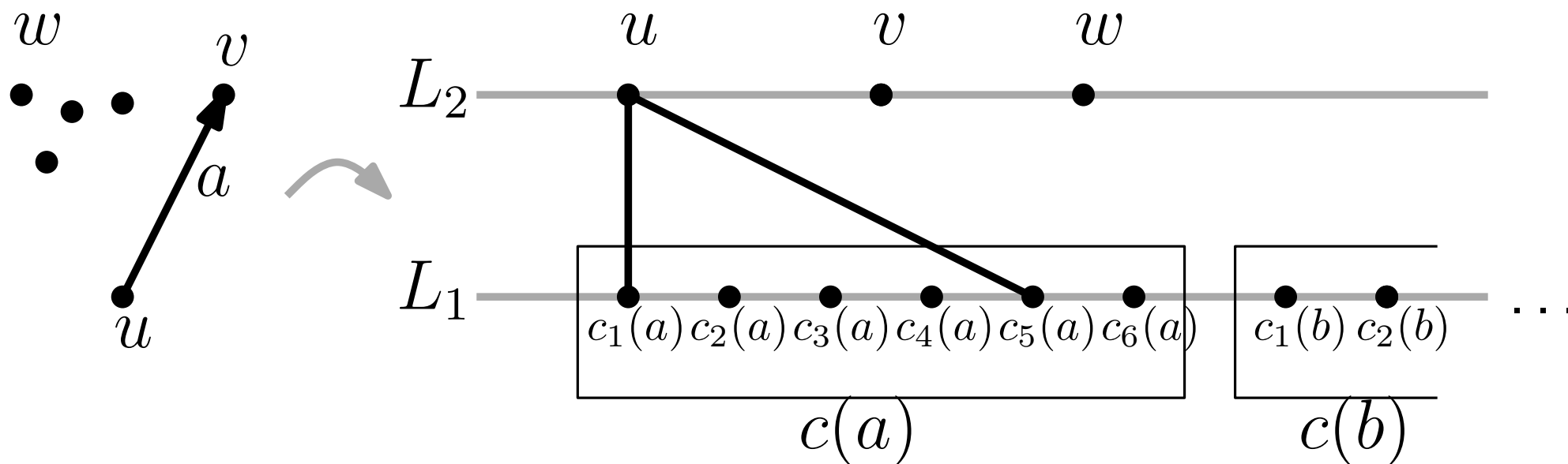


**Satz 1:** Das einseitige Kreuzungsminimierungsproblem (OSCM) ist NP-schwer.

**Beweisskizze:** Reduktion von Feedback Arc Set

$$D = (V, A)$$

$$B = (W, E)$$

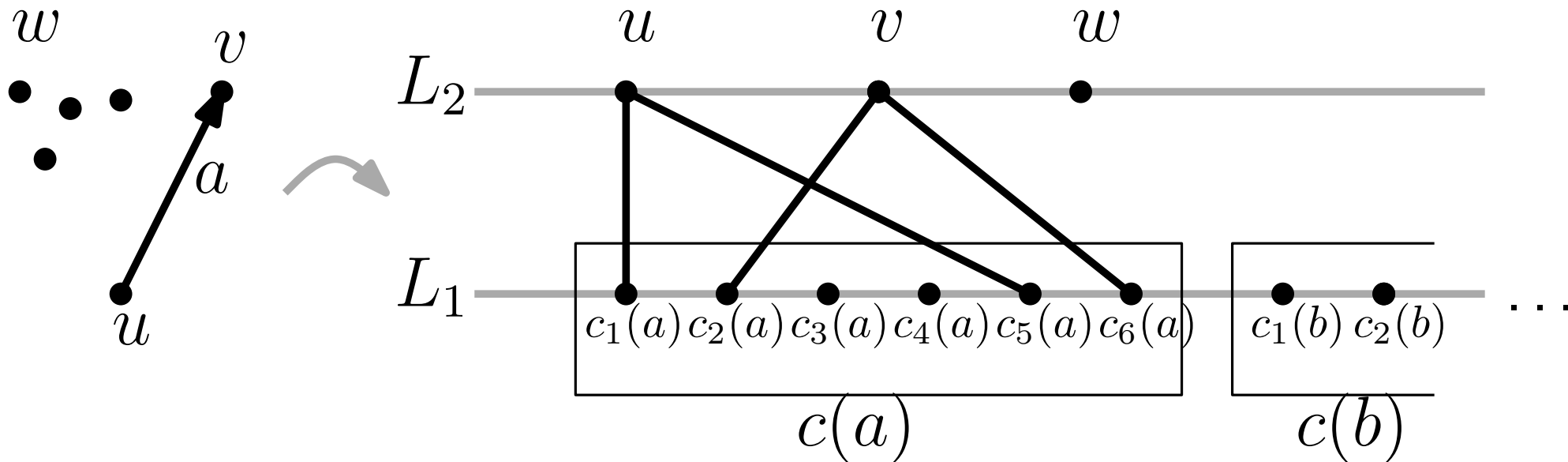


**Satz 1:** Das einseitige Kreuzungsminimierungsproblem (OSCM) ist NP-schwer.

**Beweisskizze:** Reduktion von Feedback Arc Set

$$D = (V, A)$$

$$B = (W, E)$$

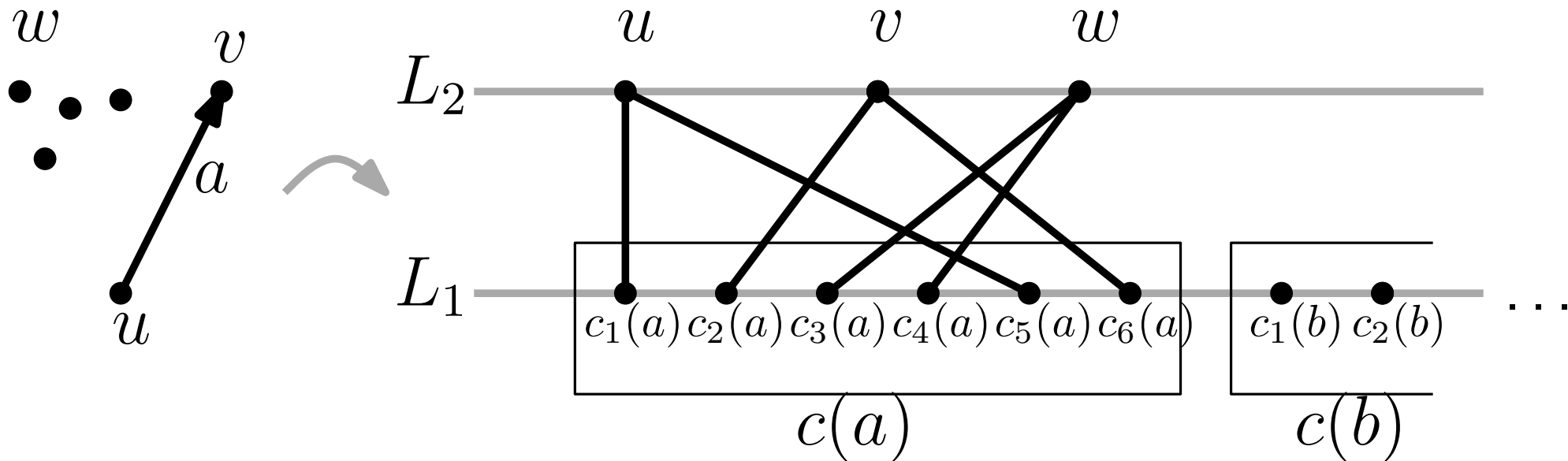


**Satz 1:** Das einseitige Kreuzungsminimierungsproblem (OSCM) ist NP-schwer.

**Beweisskizze:** Reduktion von Feedback Arc Set

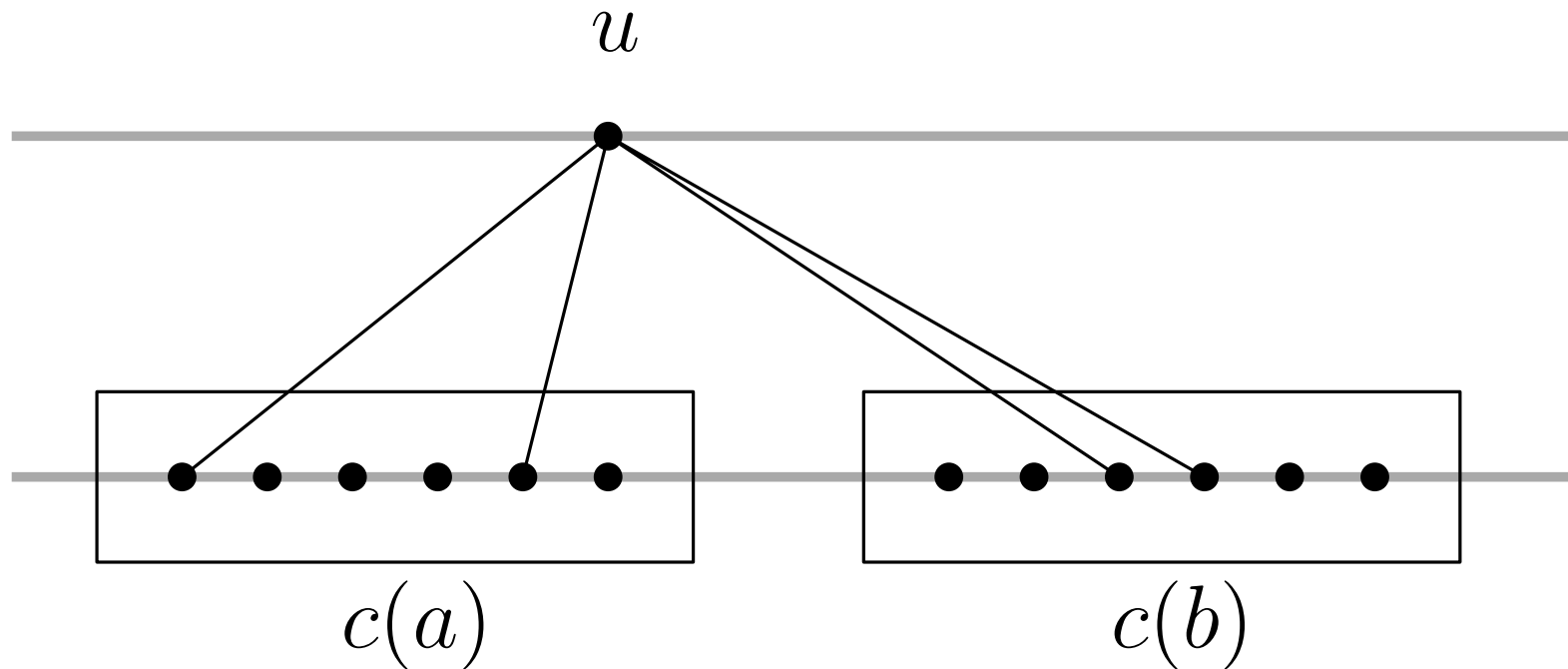
$$D = (V, A)$$

$$B = (W, E)$$



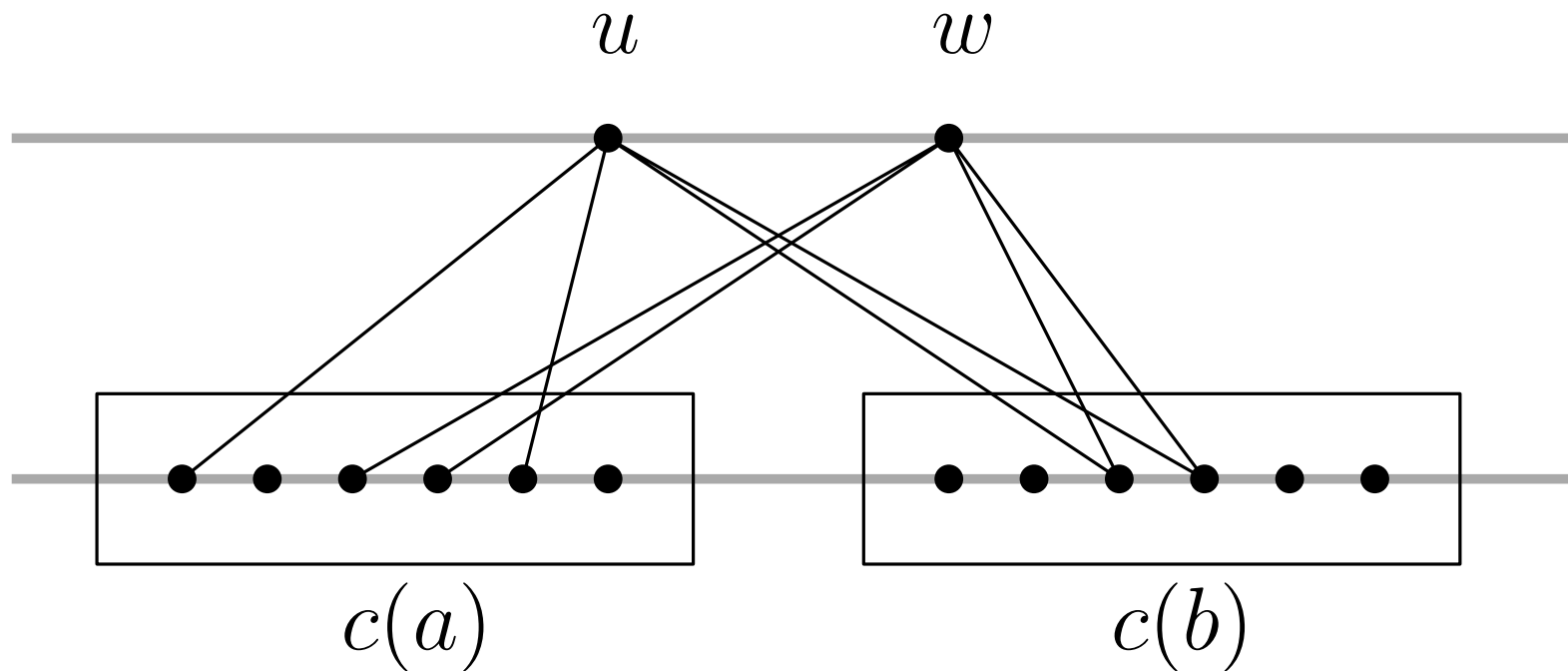
# Komplexität OSCM

## Kreuzungen zählen



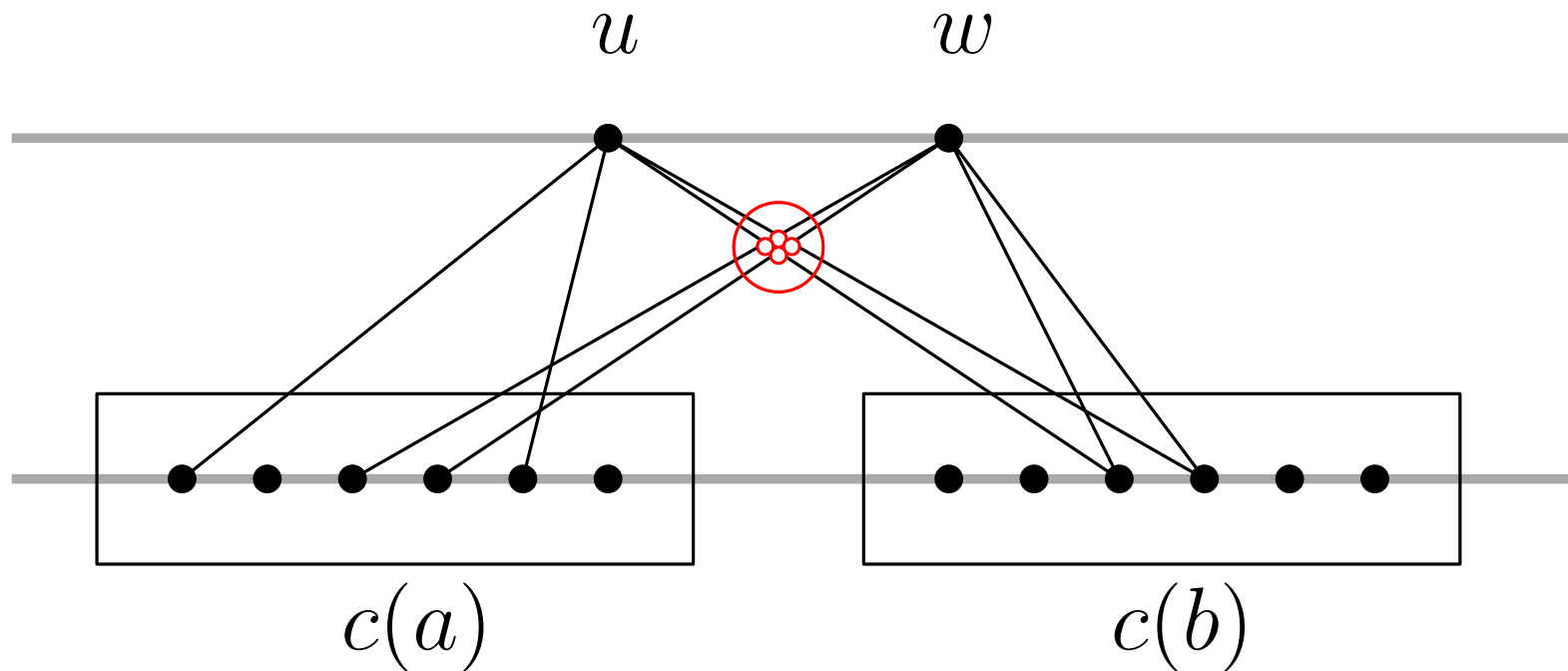
# Komplexität OSCM

## Kreuzungen zählen



# Komplexität OSCM

## Kreuzungen zählen

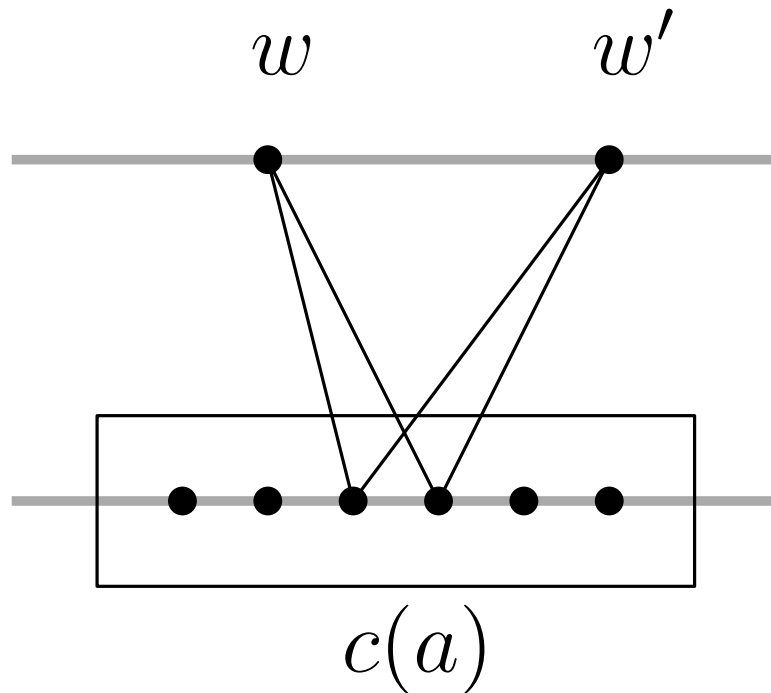


$$4 \cdot \binom{n}{2} \cdot \binom{m}{2}$$

# Komplexität OSCM

## Kreuzungen zählen

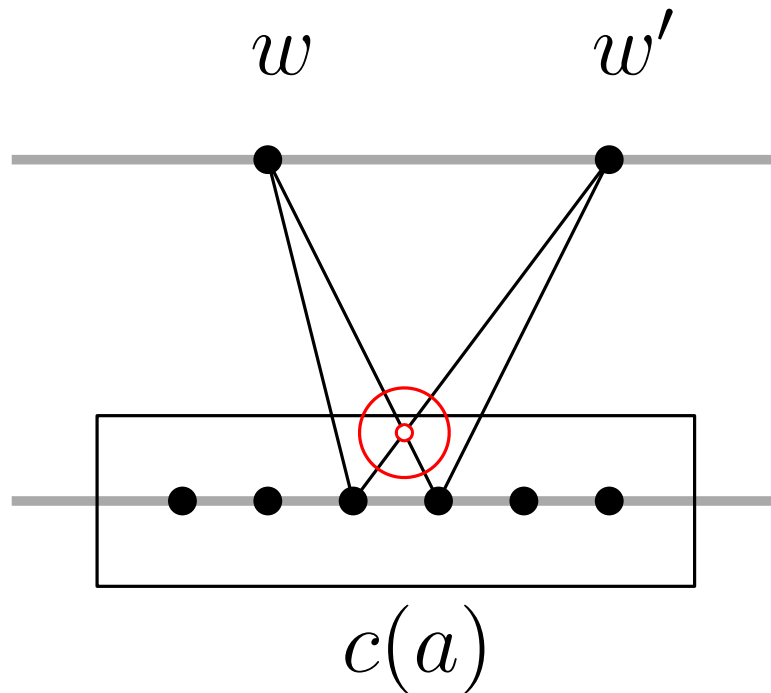
$$4 \cdot \binom{n}{2} \cdot \binom{m}{2}$$



# Komplexität OSCM

## Kreuzungen zählen

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



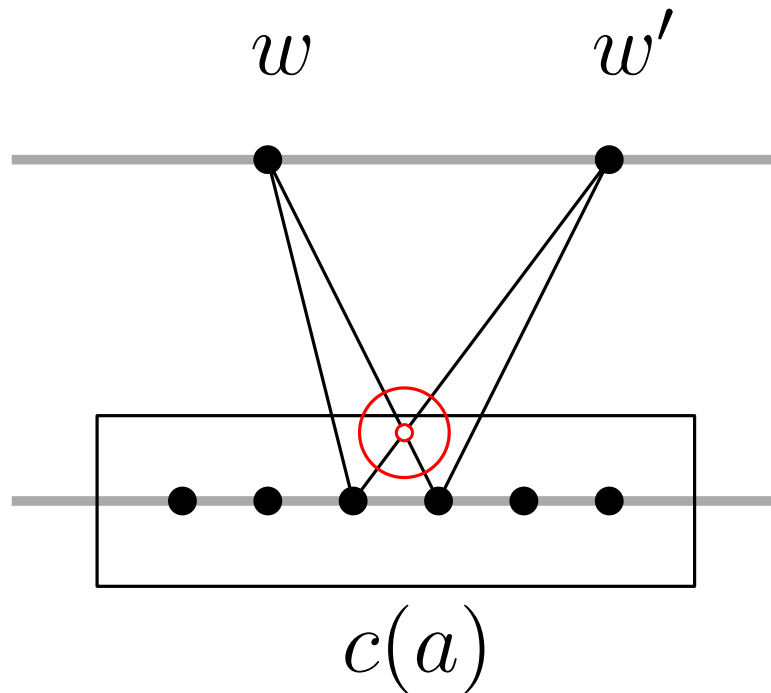
$$+ m \cdot \binom{n-2}{2}$$



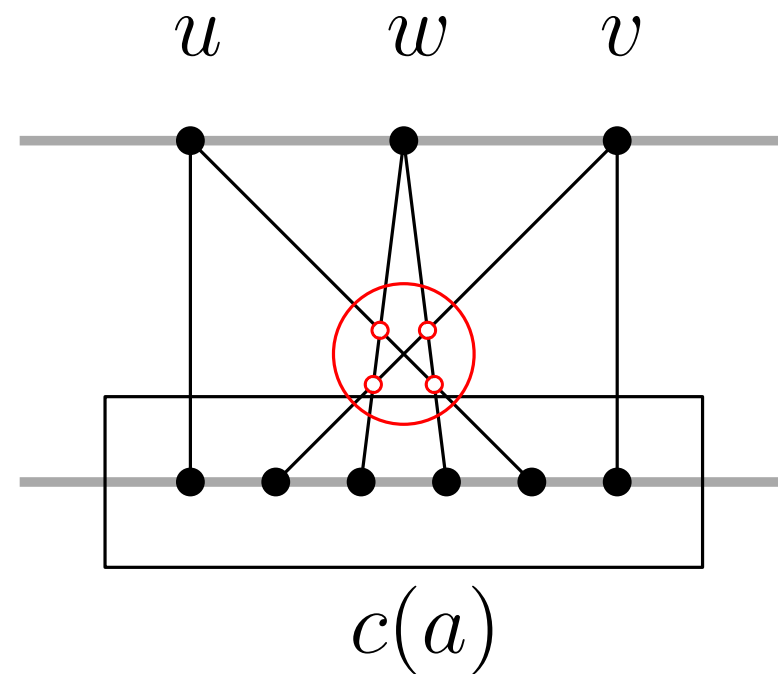
# Komplexität OSCM

## Kreuzungen zählen

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



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

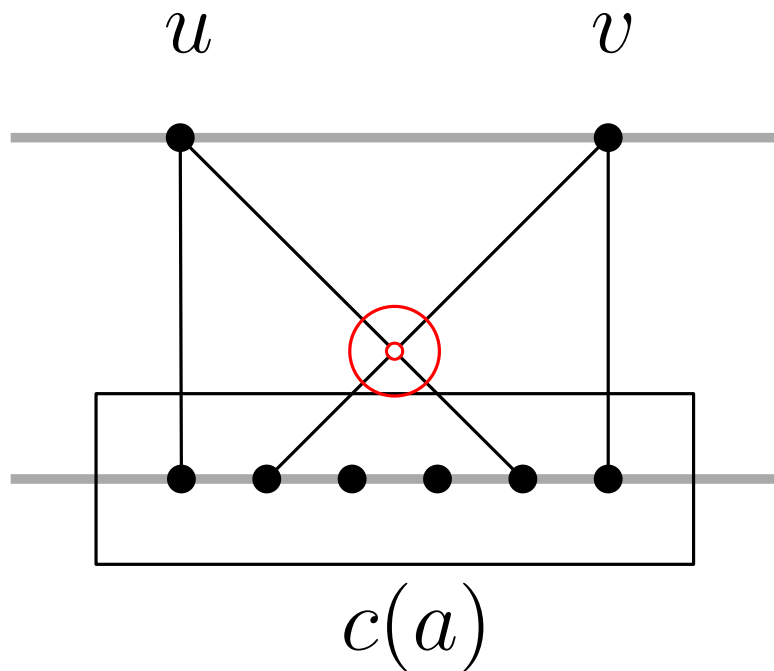


$$+ 4 \cdot m \cdot (n - 2)$$

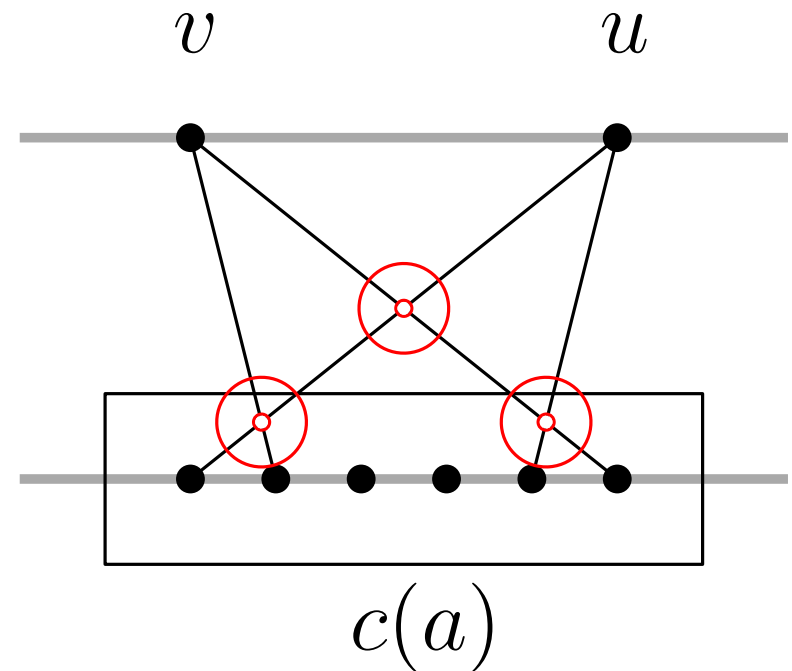
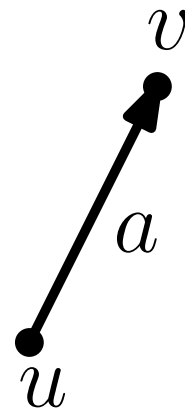
# Komplexität OSCM

## Kreuzungen zählen

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



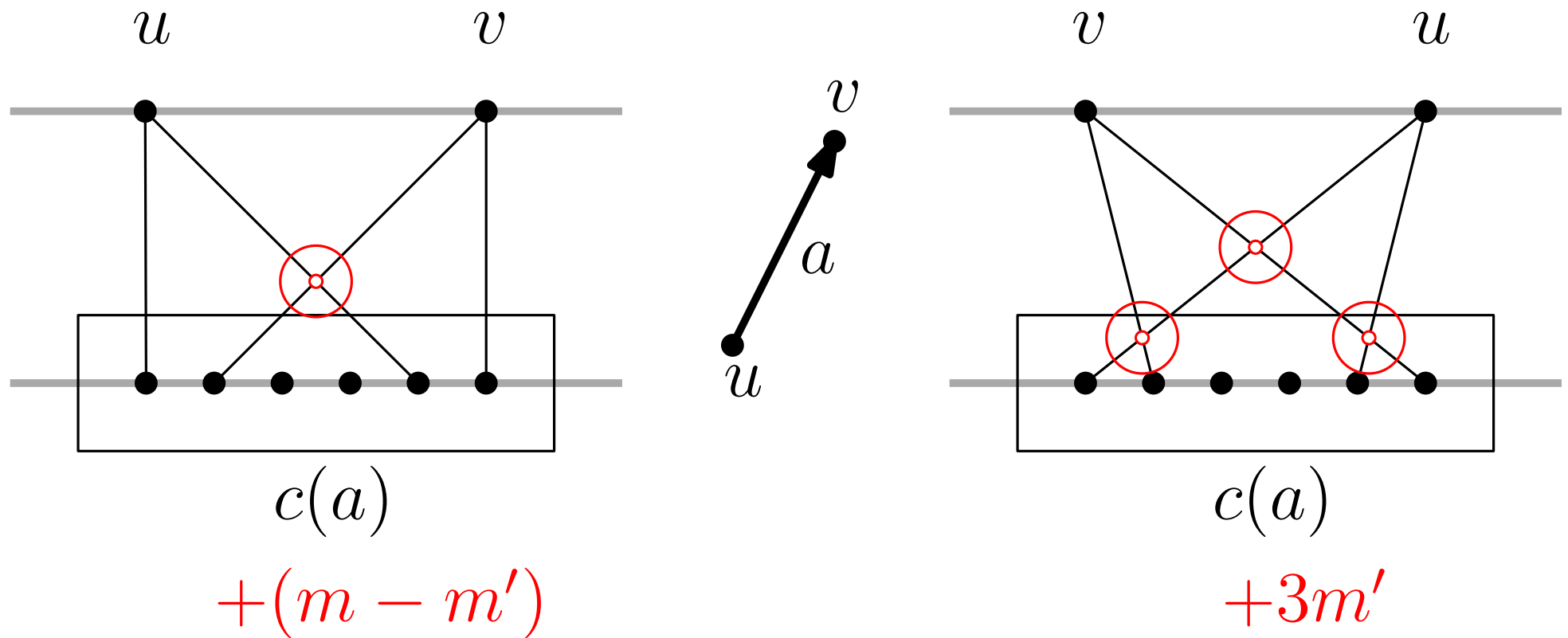
$$+(m - m')$$



$$+3m'$$

## Kreuzungen zählen

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



$$\text{opt}(G, x_1) \leq M + m + 2m'$$

$$\Leftrightarrow D \text{ hat FAS der Gr\u00f6\u00dfe } \leq m'$$

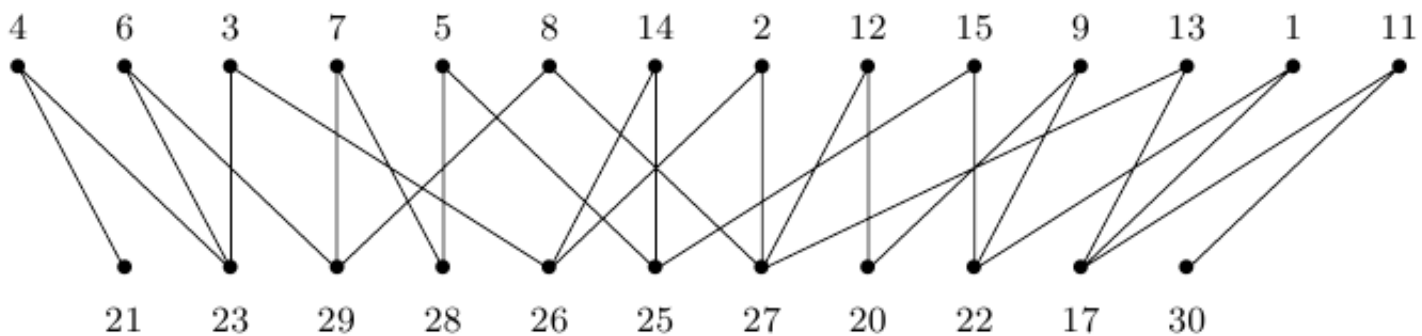
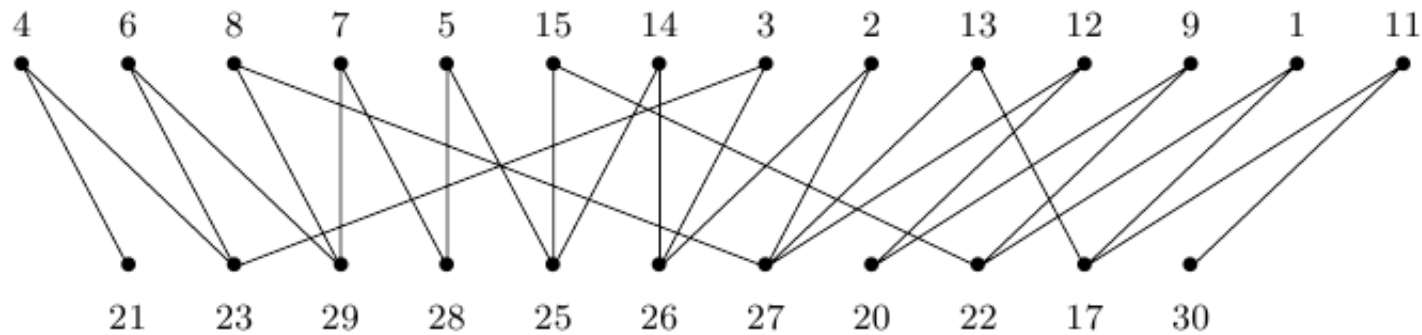
□

## Heuristiken:

- Barycenter
- Median
- ...

## Exakt:

- ILP Modellierung



**Idee:** wenige Kreuzungen, wenn Knoten nah an den Nachbarn

- setze

$$x_2(u) = \frac{1}{\deg(u)} \sum_{v \in N(u)} x_1(v)$$

- bei Gleichheit um kleinen Wert trennen

**Idee:** wenige Kreuzungen, wenn Knoten nah an den Nachbarn

- setze

$$x_2(u) = \frac{1}{\deg(u)} \sum_{v \in N(u)} x_1(u)$$

- bei Gleichheit um kleinen Wert trennen

## Eigenschaften:

- geringer Implementationsaufwand
- schnell
- meist sehr gute Ergebnisse
- findet Optimum falls  $\text{opt}(G, x_1) = 0$  (s. Übungsblatt)
- im worst case um Faktor  $\Omega(\sqrt{n})$  schlechter

**Idee:** setze Koordinate auf Median der Nachbarn

- für Knoten  $v \in L_2$  mit Nachbarn  $v_1, \dots, v_k$  setze  
 $x_2(v) = \text{med}(v) = x_1(v_{\lceil k/2 \rceil})$   
bzw.  $x_2(v) = 0$  falls  $N(v) = \emptyset$
- falls  $x_2(u) = x_2(v)$  mit ungleicher Gradparität, setze ungeraden Knoten nach links
- falls  $x_2(u) = x_2(v)$  mit gleicher Gradparität, setze beliebigen Knoten nach links
- Berechnung mit Linearzeit-Medialgorithmus in  $O(|E|)$

**Idee:** setze Koordinate auf Median der Nachbarn

- für Knoten  $v \in L_2$  mit Nachbarn  $v_1, \dots, v_k$  setze  
 $x_2(v) = \text{med}(v) = x_1(v_{\lceil k/2 \rceil})$   
bzw.  $x_2(v) = 0$  falls  $N(v) = \emptyset$
- falls  $x_2(u) = x_2(v)$  mit ungleicher Gradparität, setze ungeraden Knoten nach links
- falls  $x_2(u) = x_2(v)$  mit gleicher Gradparität, setze beliebigen Knoten nach links
- Berechnung mit Linearzeit-Medialgorithmus in  $O(|E|)$

**Eigenschaften:**

- geringer Implementationsaufwand
- schnell
- meist gute Ergebnisse
- findet Optimum falls  $\text{opt}(G, x_1) = 0$
- **Faktor-3 Approximation**

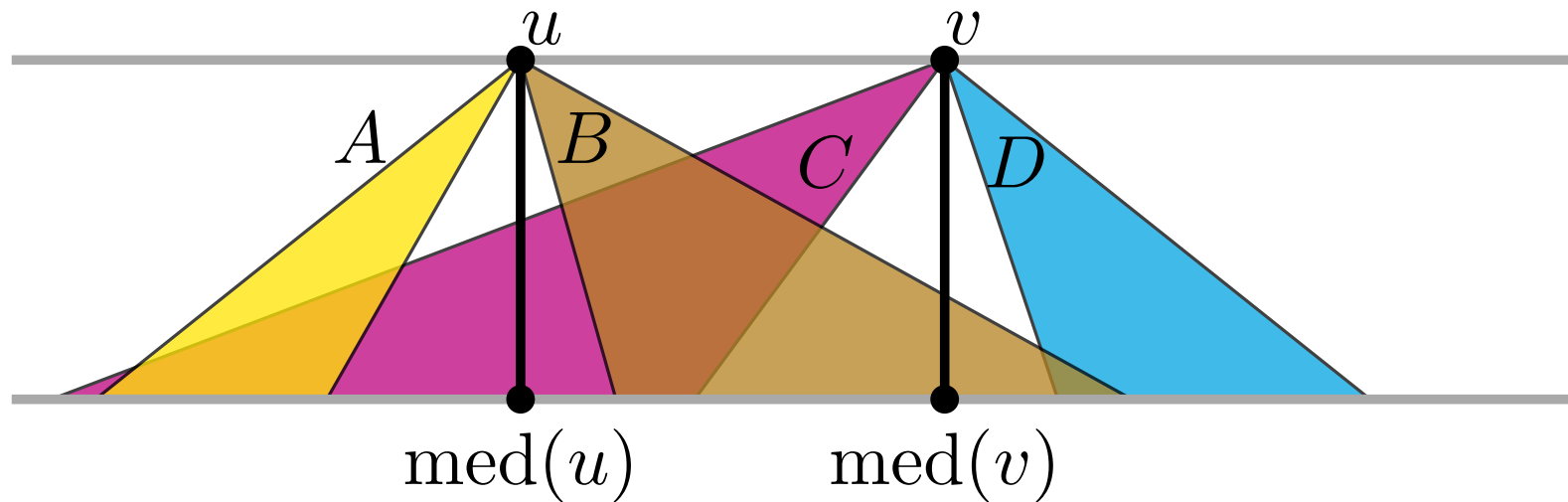


**Satz 2:** Sei  $G = (L_1, L_2, E)$  ein 2-Lagen-Graph und  $x_1$  eine beliebige Ordnung von  $L_1$ . Dann gilt

$$\text{med}(G, x_1) \leq 3 \text{opt}(G, x_1).$$

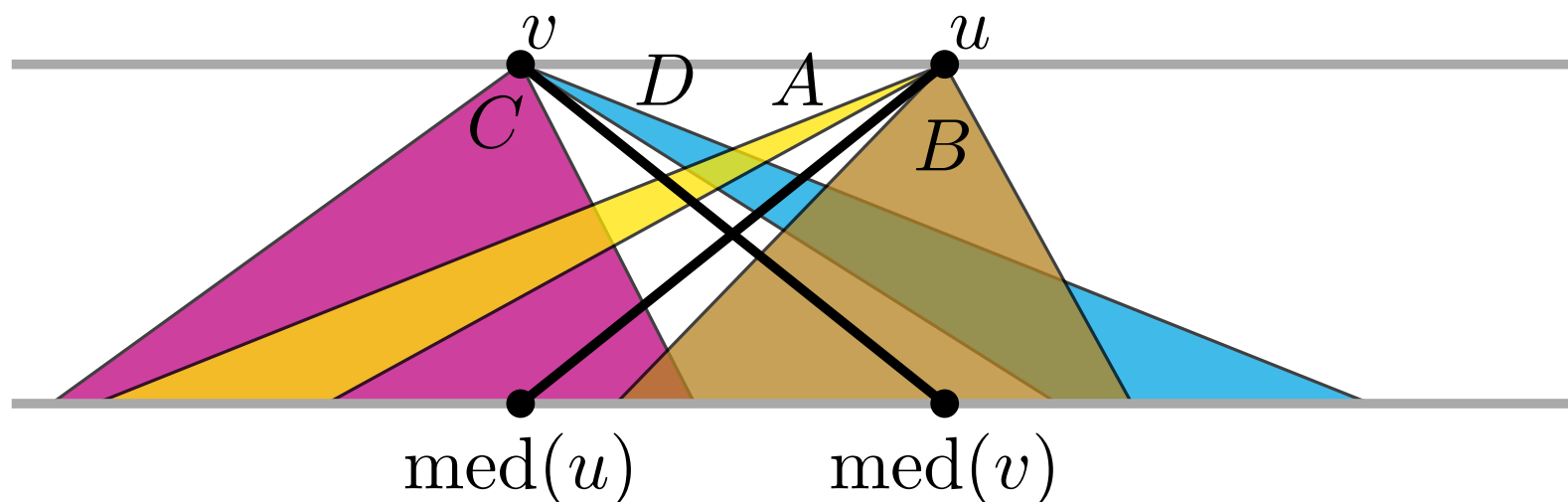
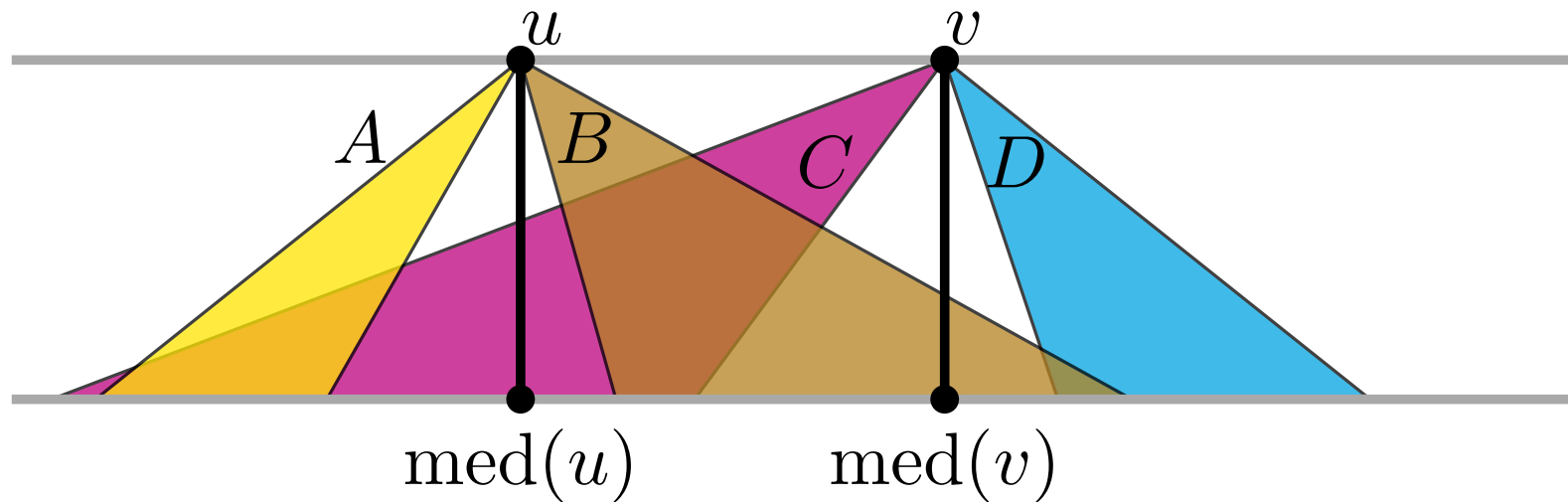
# Approximationsfaktor

**Satz 2:** Sei  $G = (L_1, L_2, E)$  ein 2-Lagen-Graph und  $x_1$  eine beliebige Ordnung von  $L_1$ . Dann gilt

$$\text{med}(G, x_1) \leq 3 \text{opt}(G, x_1).$$


# Approximationsfaktor

**Satz 2:** Sei  $G = (L_1, L_2, E)$  ein 2-Lagen-Graph und  $x_1$  eine beliebige Ordnung von  $L_1$ . Dann gilt

$$\text{med}(G, x_1) \leq 3 \text{opt}(G, x_1).$$


## Eigenschaften:

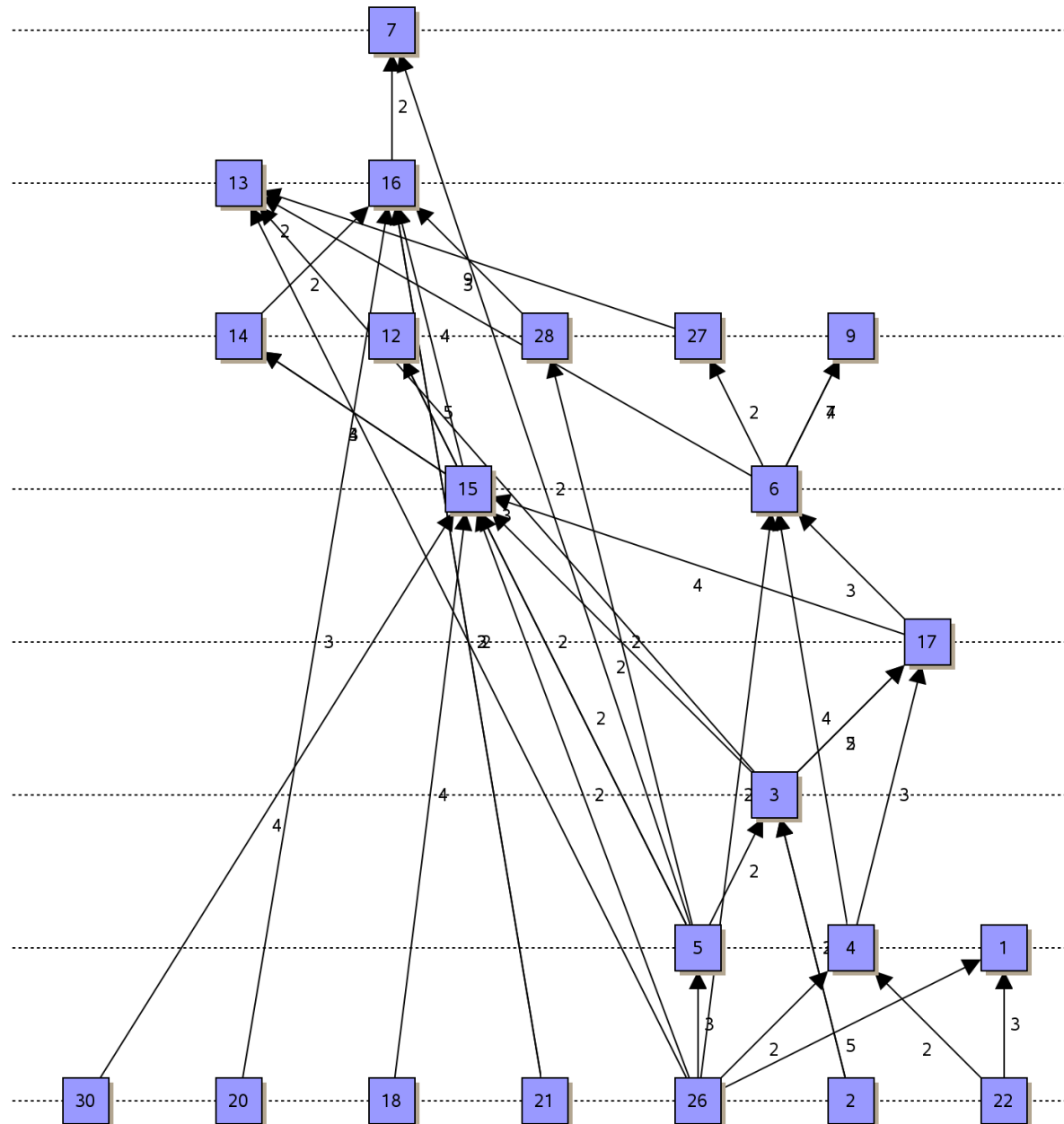
- branch-and-cut Technik für DAGs beschränkter Größe
- findet optimale Lösung
- Lösung jedoch nicht in polynomieller Zeit garantiert
- nützlich für kleine bis mittlere Graphen

## Eigenschaften:

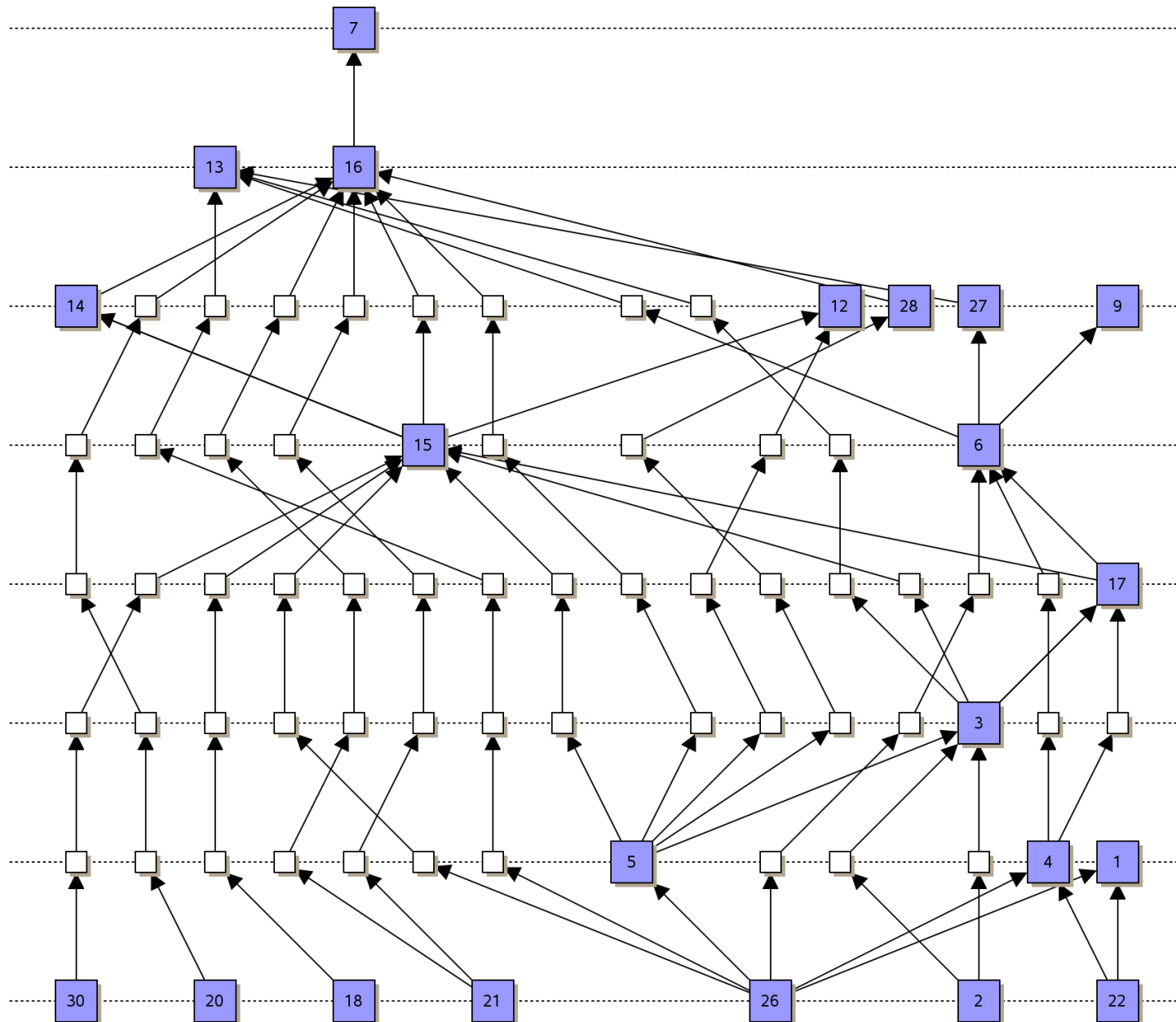
- branch-and-cut Technik für DAGs beschränkter Größe
- findet optimale Lösung
- Lösung jedoch nicht in polynomieller Zeit garantiert
- nützlich für kleine bis mittlere Graphen

**Modell:** siehe Tafel

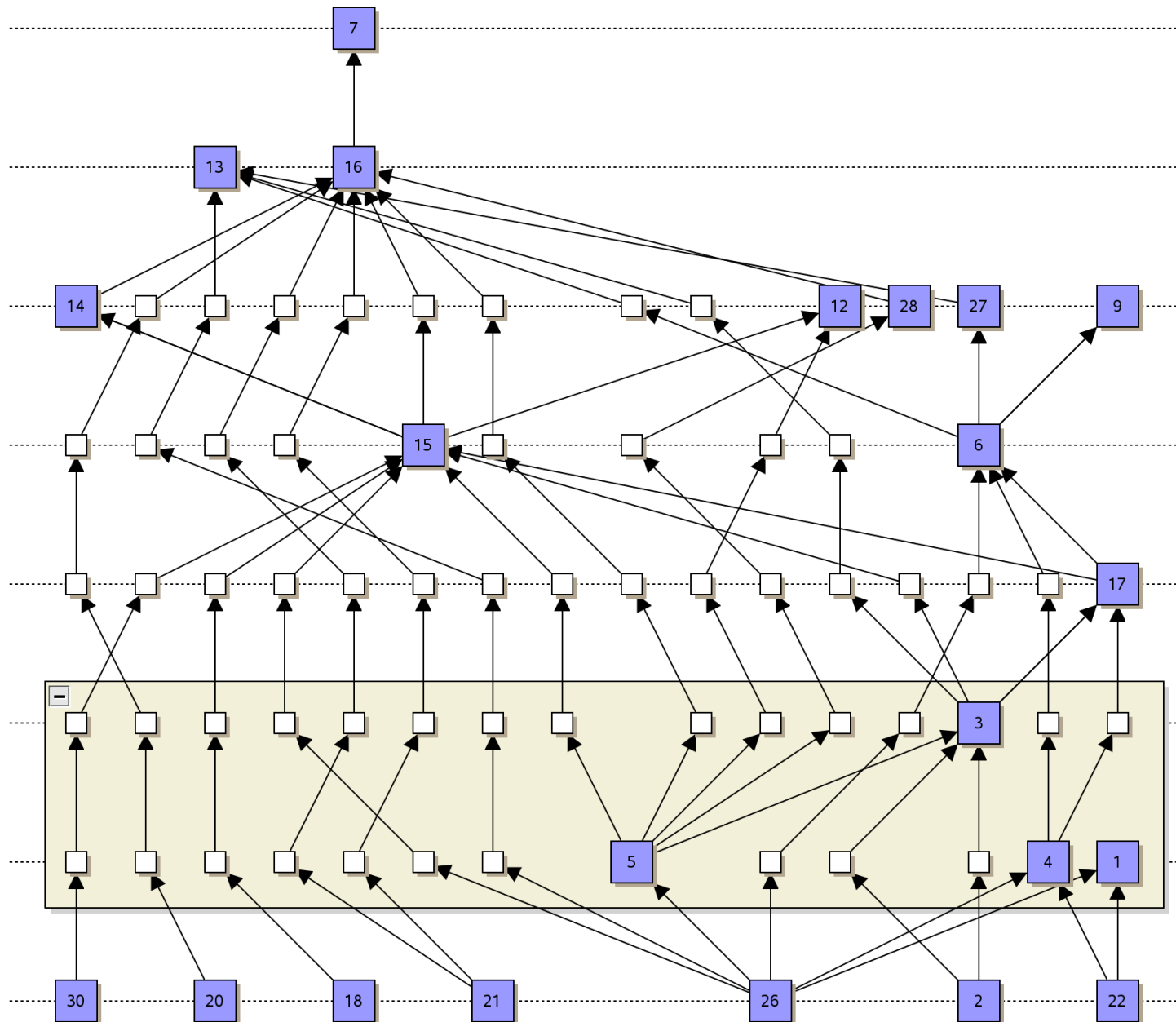
# Beispiel



# Beispiel

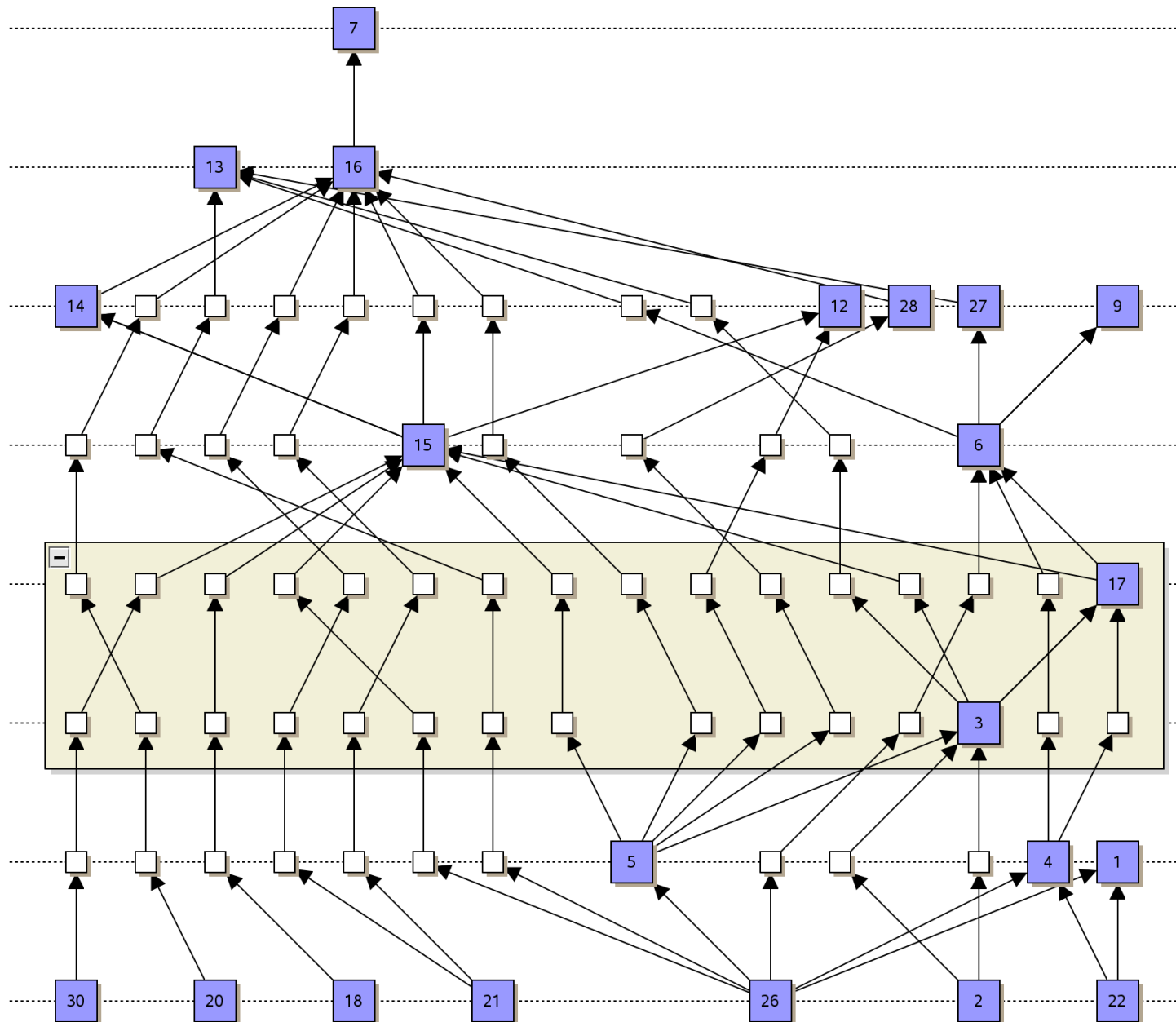


# Beispiel

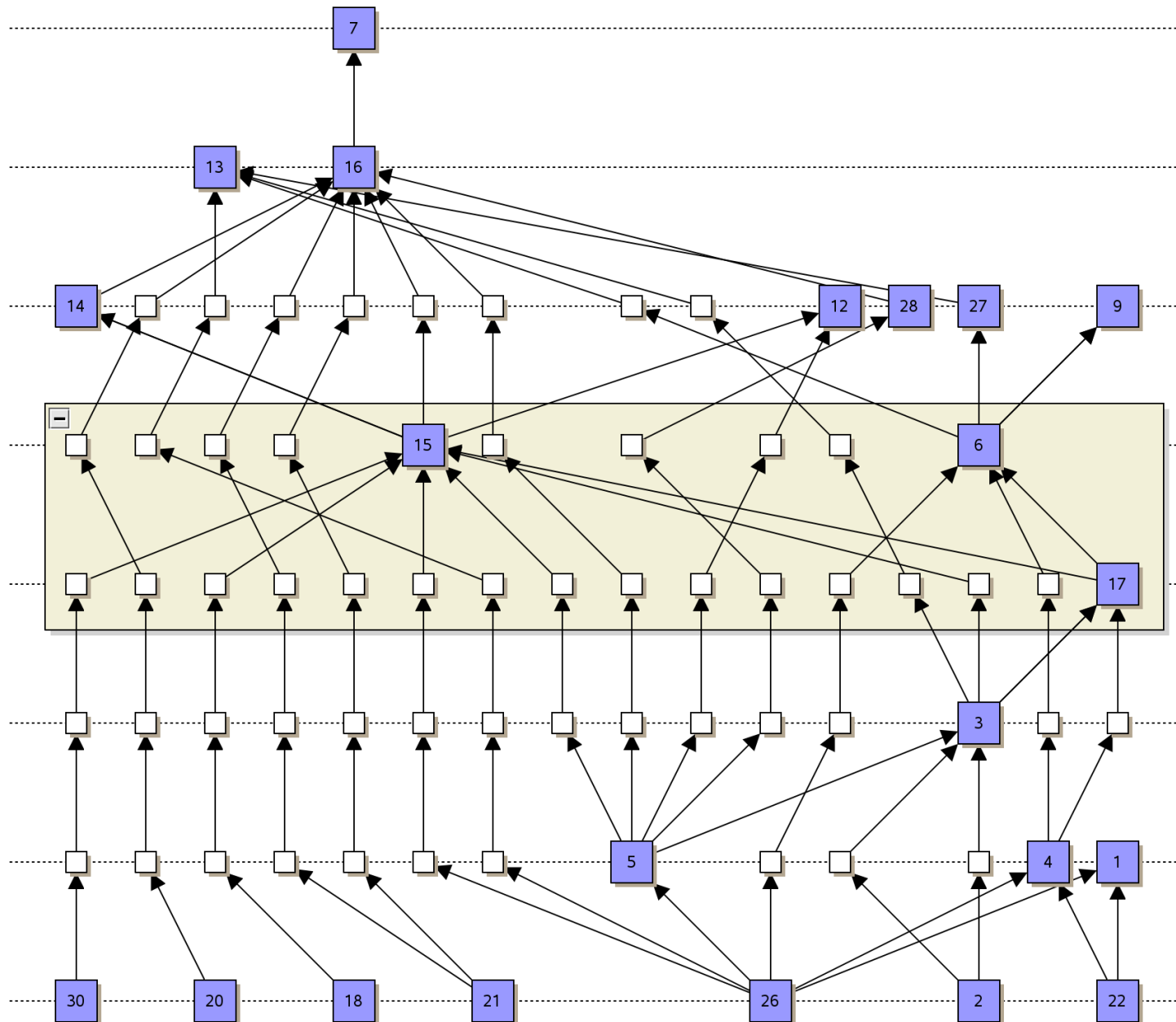


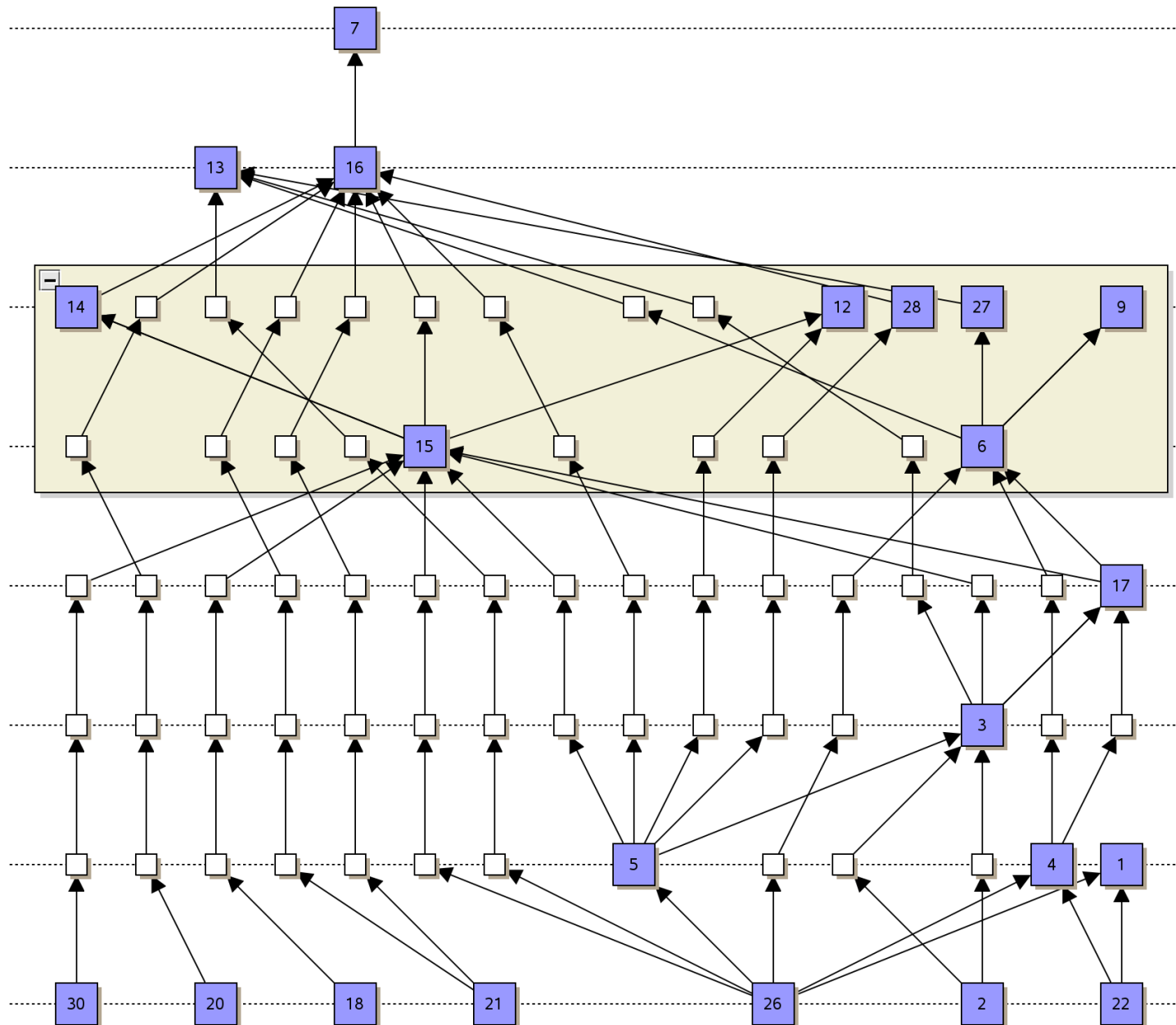


# Beispiel

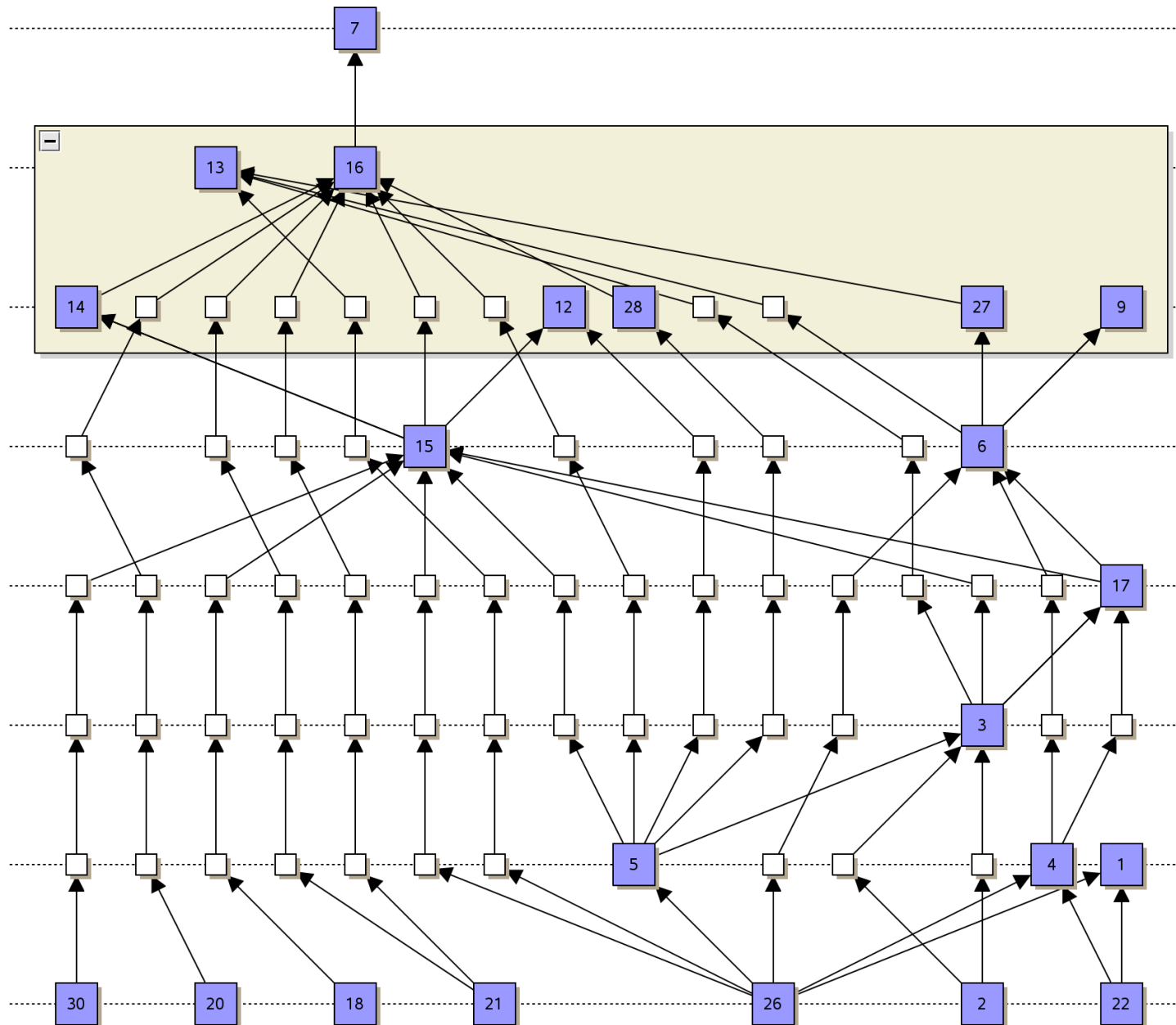


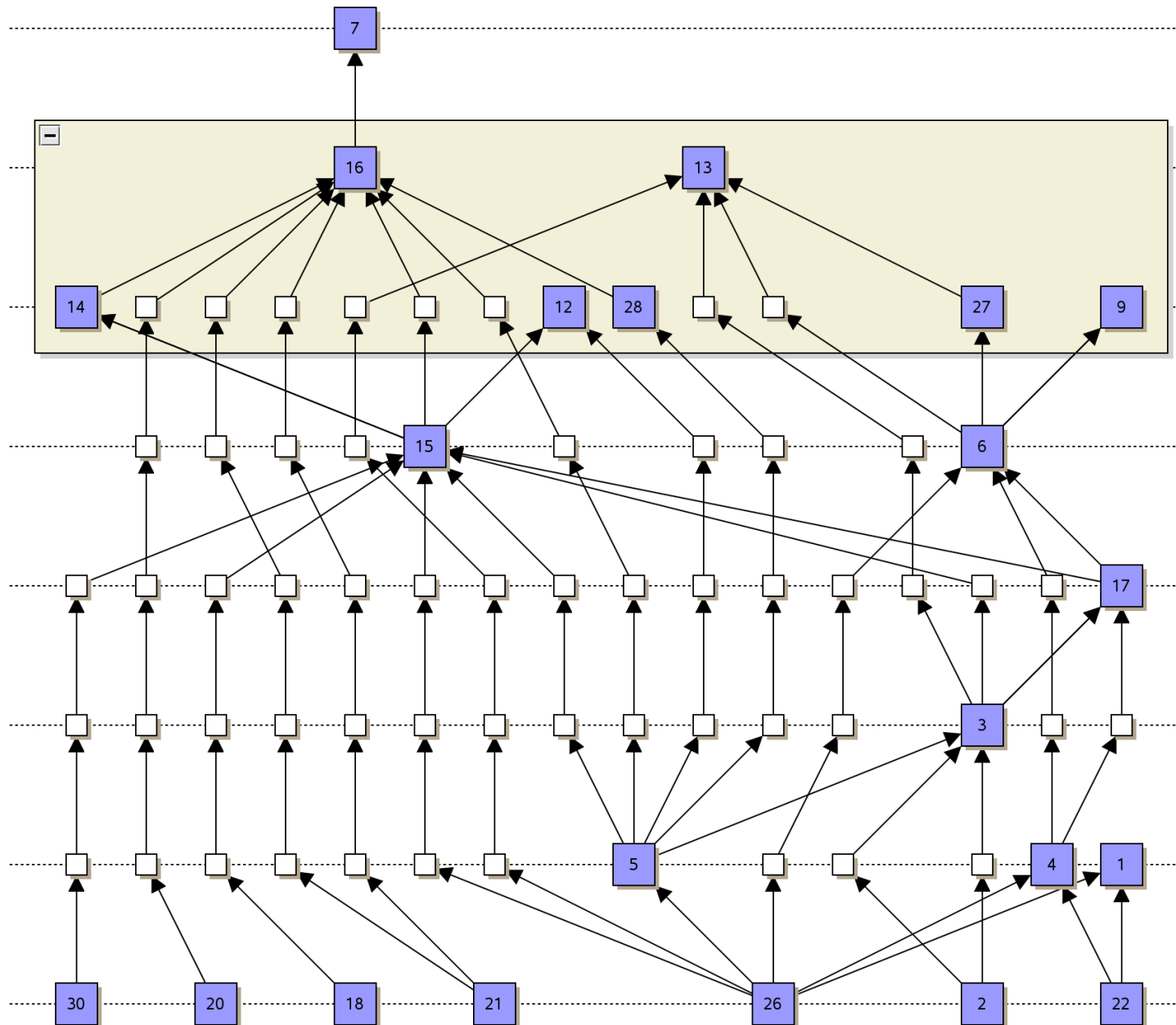
# Beispiel



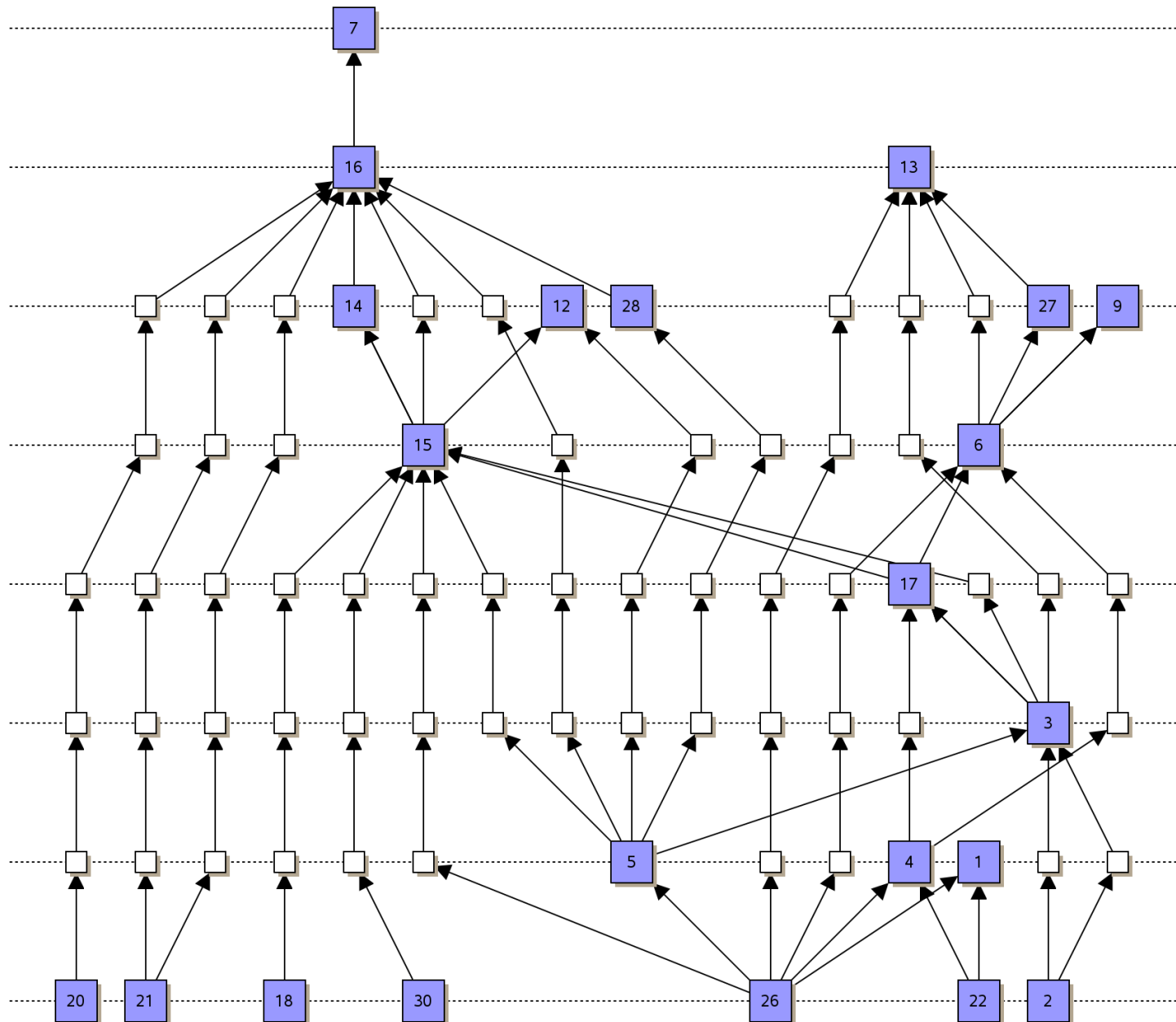


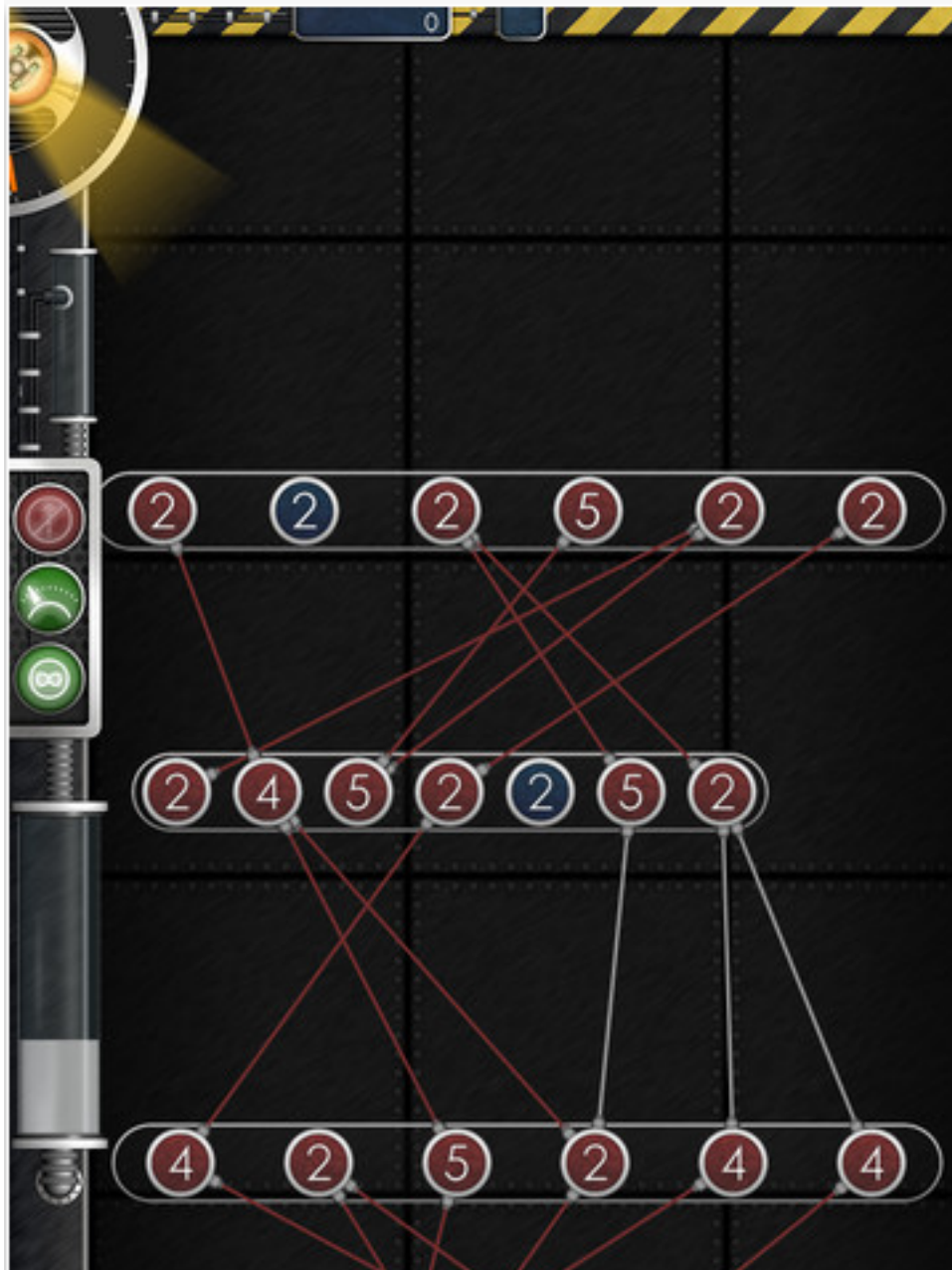
# Beispiel





# Beispiel

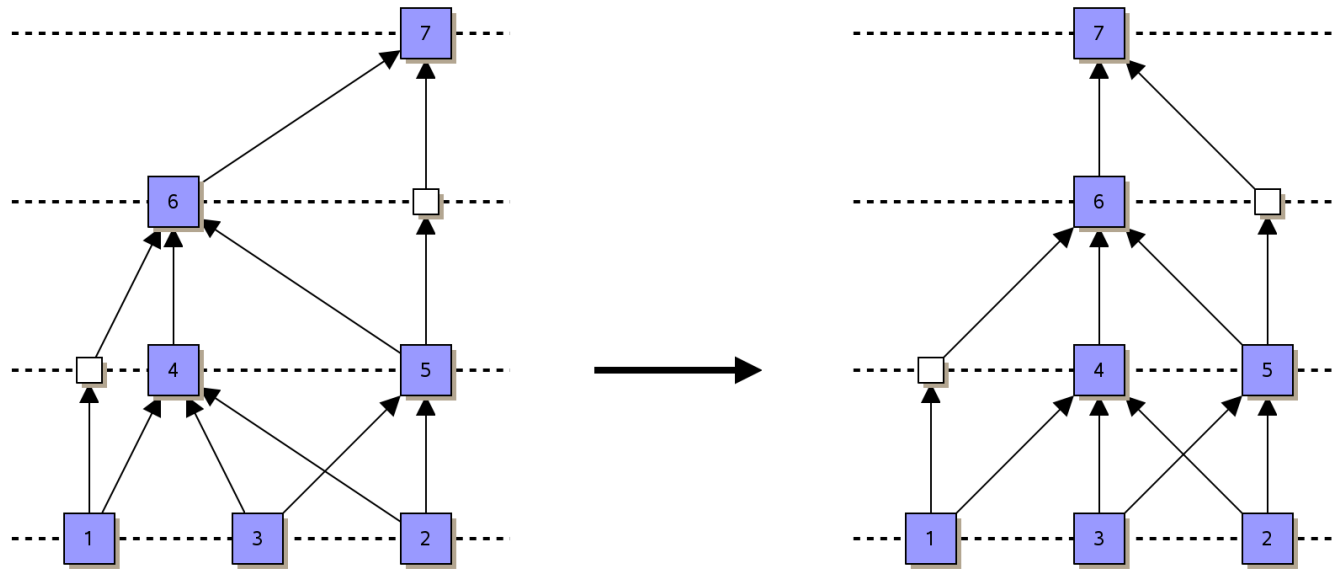




Gibt sogar ein iPad  
Spiel **CrossingX**  
für das OSCM  
Problem!

Gewinner Graph Drawing Game Contest 2012

# Schritt 4: Koordinatenzuweisung



Welche Ziele gibt es?



# Kanten begradigen

**Ziel:** minimiere Abweichung von gerader Linie für Kanten mit Dummy-Knoten

**Idee:** nutze quadratisches Programm

- sei  $p_{uv} = (u, d_1, \dots, d_k, v)$  Pfad mit  $k$  Dummyknoten zwischen  $u$  und  $v$
- sei  $a_i = x(u) + \frac{i}{k+1}(x(v) - x(u))$  die  $x$ -Koordinate von  $d_i$  bei gerader Linie
- minimiere  $\sum_{i=1}^k (x(d_i) - a_i)^2$  über alle Pfade
- Nebenbedingungen:  $x(w) - x(z) \geq \delta$  für alle Knoten der gleichen Lage und  $w$  rechts von  $z$  ( $\delta$  Abstandsparameter)

# Kanten begradigen

**Ziel:** minimiere Abweichung von gerader Linie für Kanten mit Dummy-Knoten

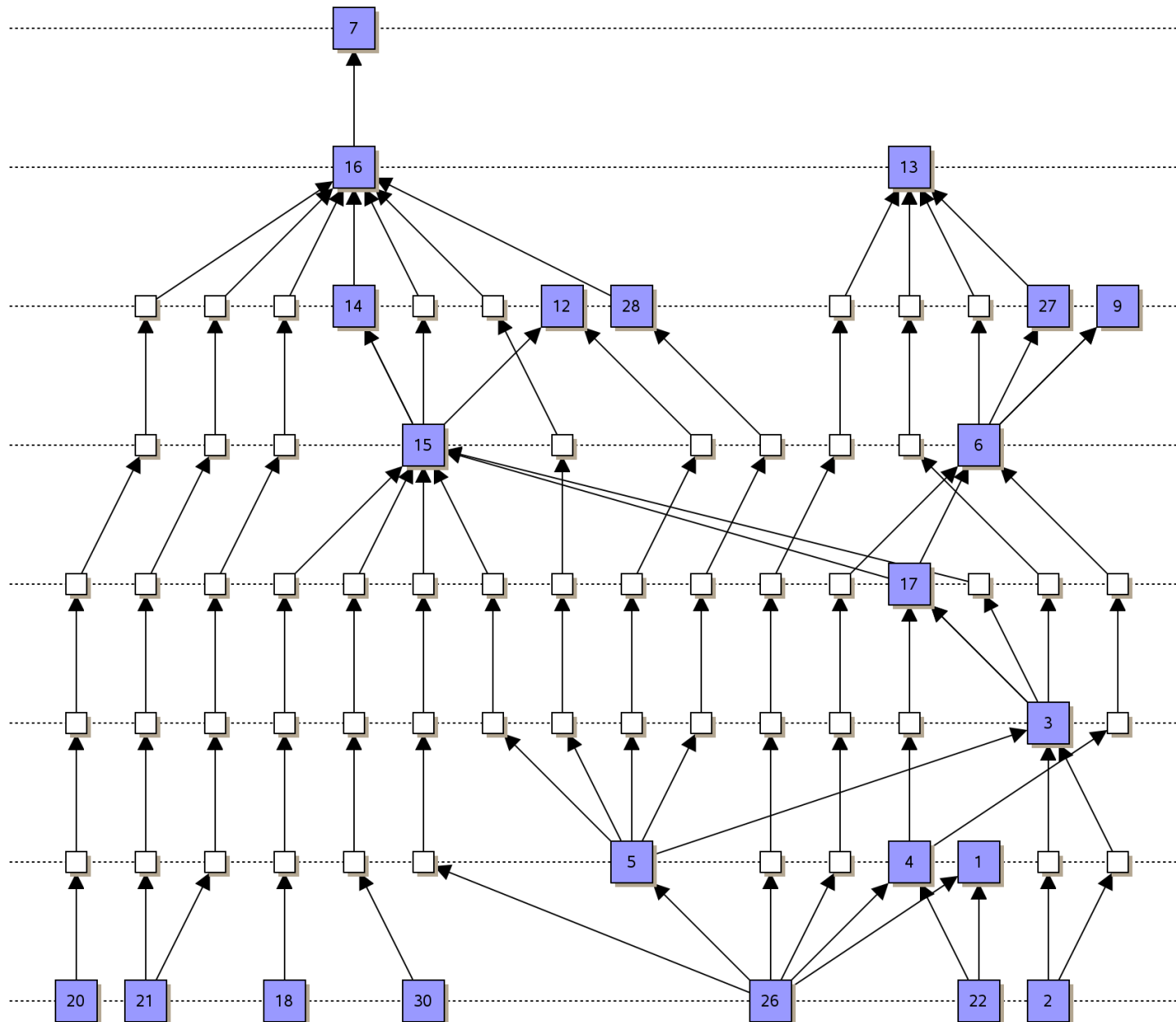
**Idee:** nutze quadratisches Programm

- sei  $p_{uv} = (u, d_1, \dots, d_k, v)$  Pfad mit  $k$  Dummyknoten zwischen  $u$  und  $v$
- sei  $a_i = x(u) + \frac{i}{k+1}(x(v) - x(u))$  die  $x$ -Koordinate von  $d_i$  bei gerader Linie
- minimiere  $\sum_{i=1}^k (x(d_i) - a_i)^2$  über alle Pfade
- Nebenbedingungen:  $x(w) - x(z) \geq \delta$  für alle Knoten der gleichen Lage und  $w$  rechts von  $z$  ( $\delta$  Abstandsparameter)

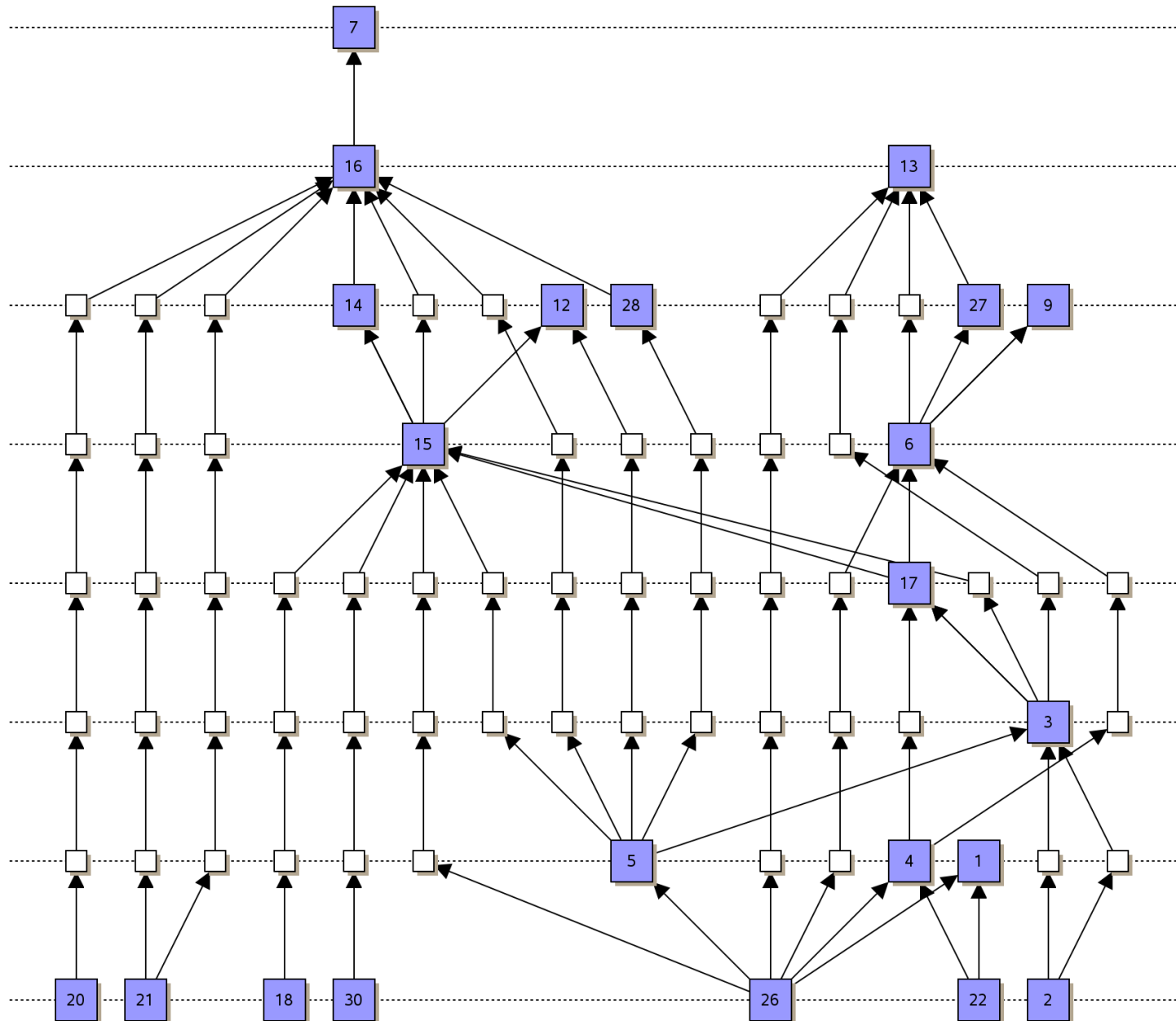
**Eigenschaften:**

- quadratisches Programm oft laufzeitintensiv
- Breite kann exponentiell wachsen
- Zielfunktion anpassbar für möglichst vertikale Kanten

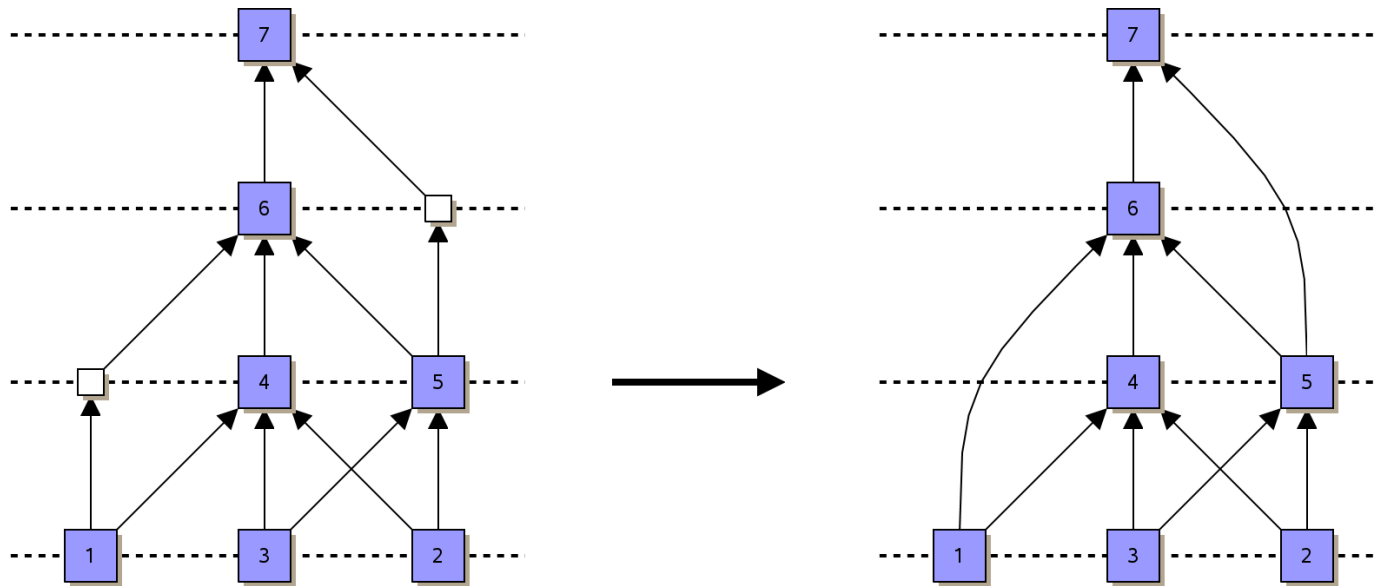
# Beispiel



# Beispiel

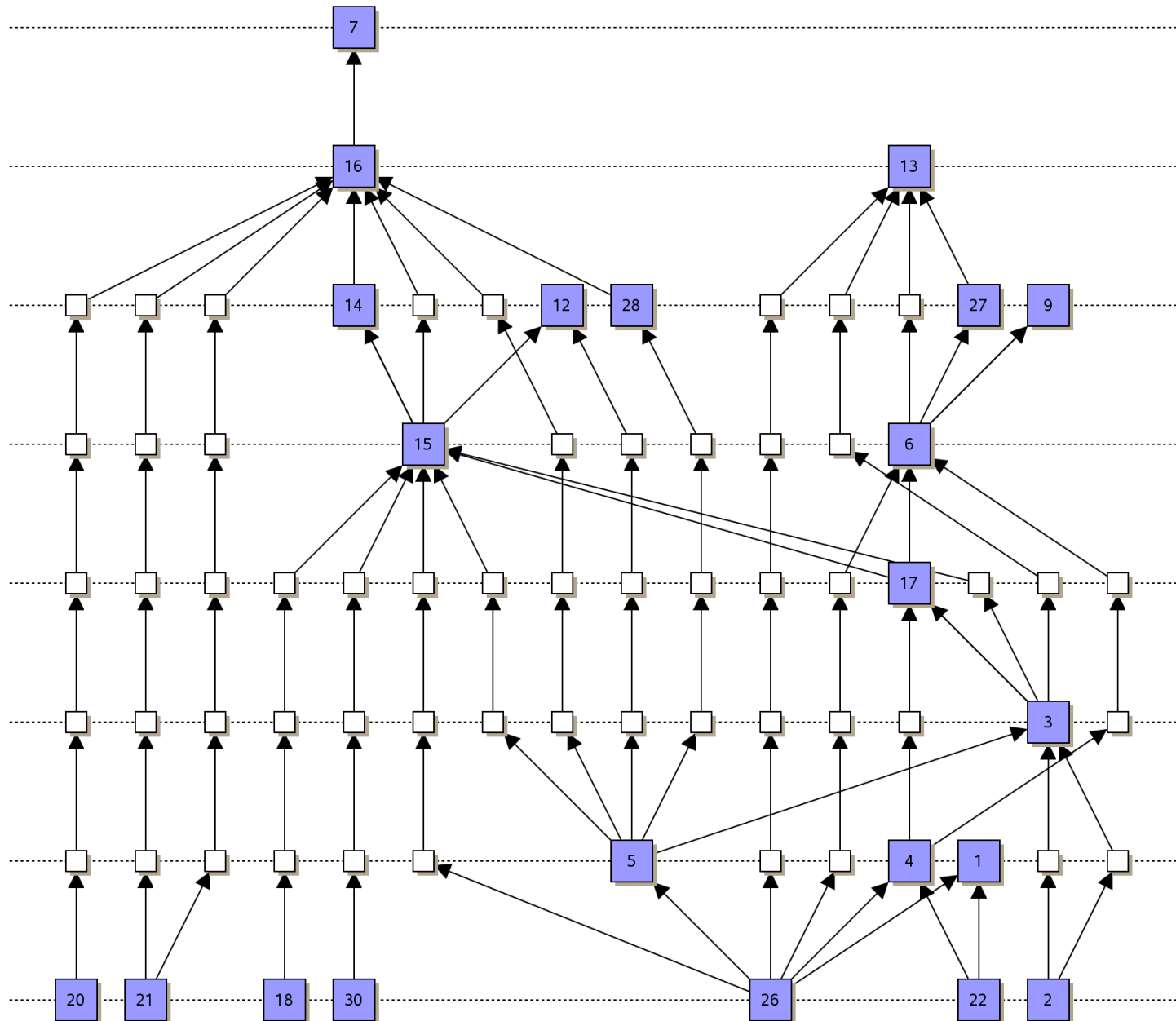


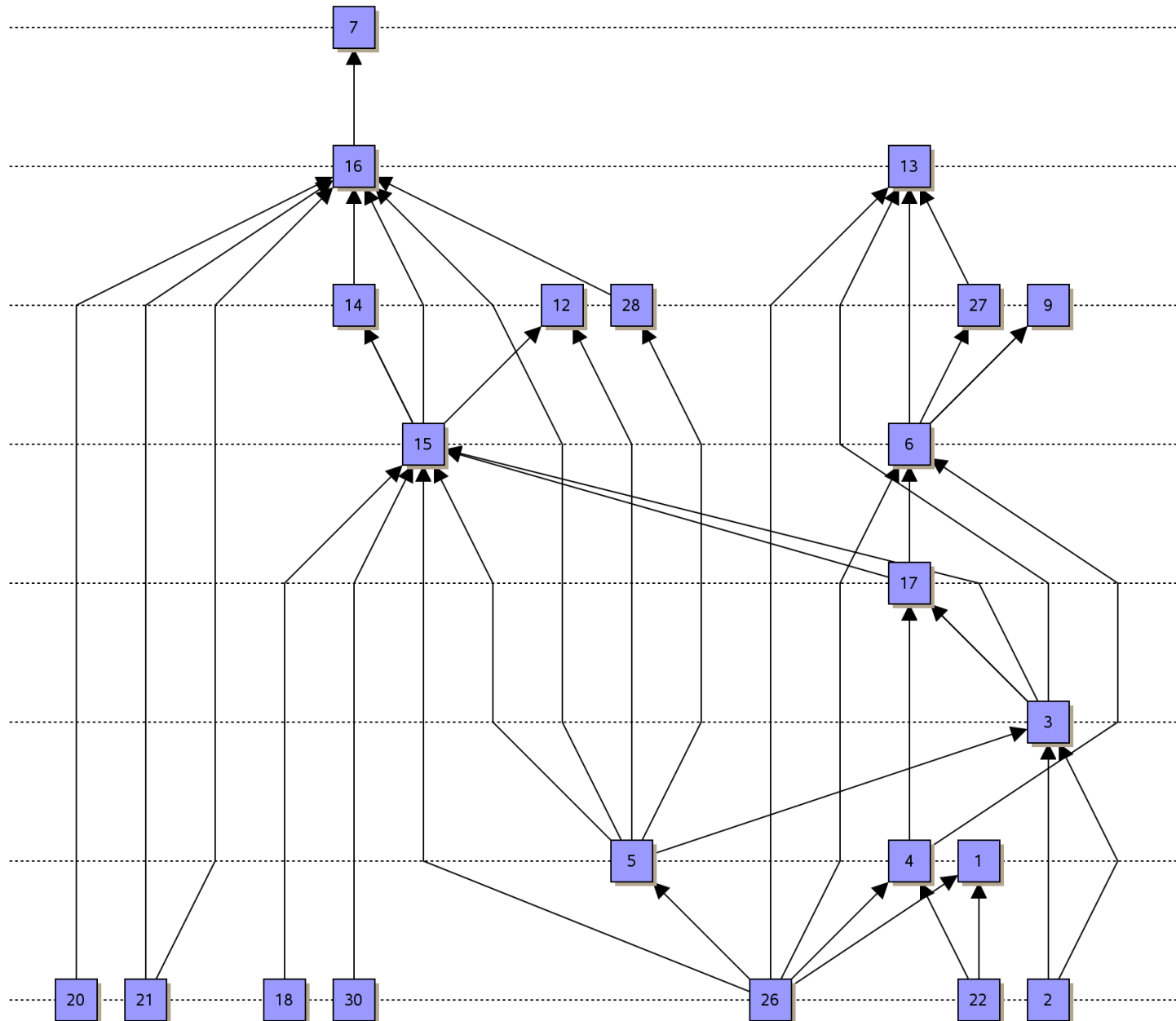
# Schritt 5: Kantenzeichnen

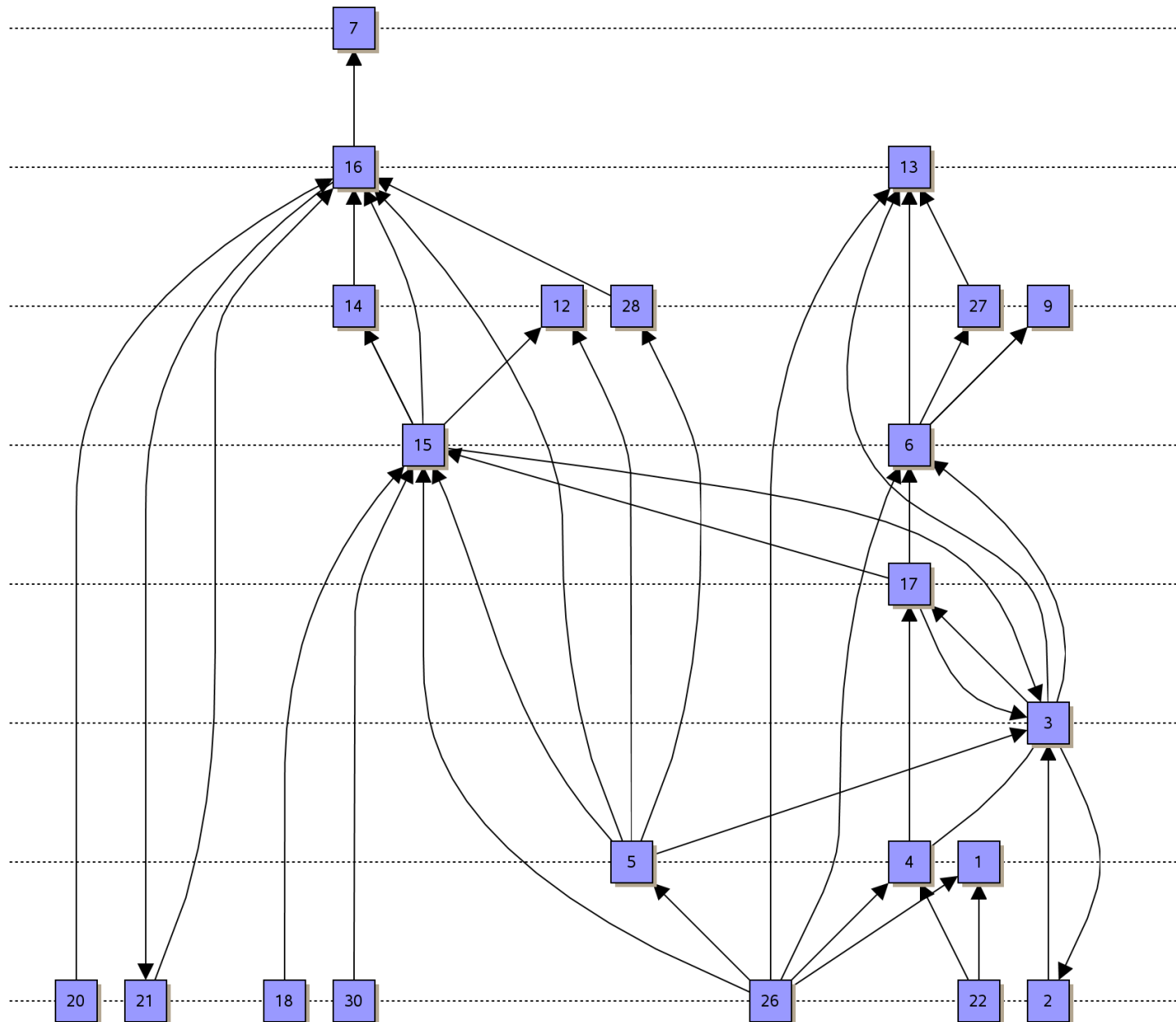


Möglichkeit: Polylines durch Bézierkurven ersetzen

# Beispiel

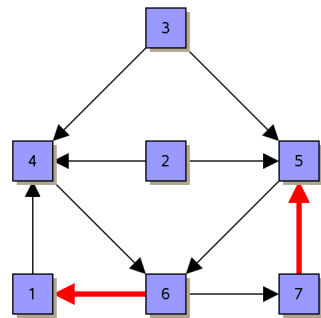




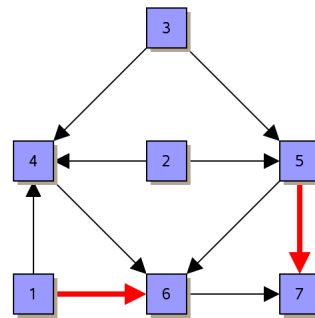




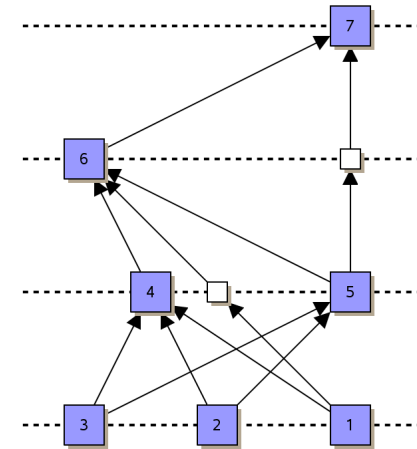
# Zusammenfassung



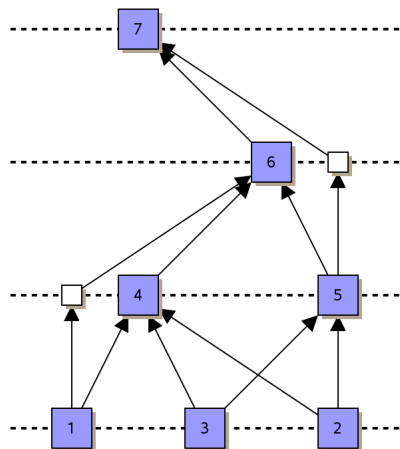
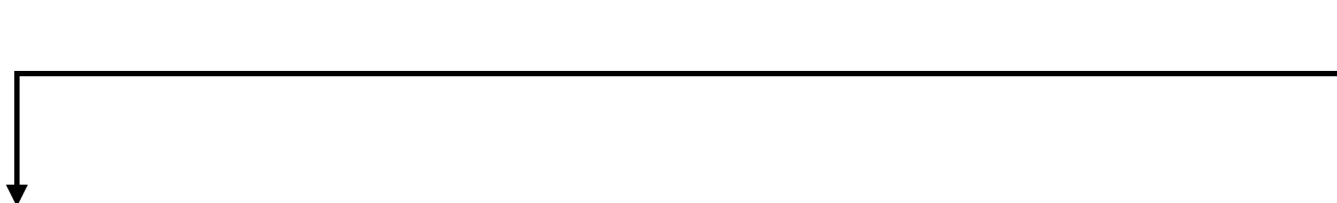
Eingabe



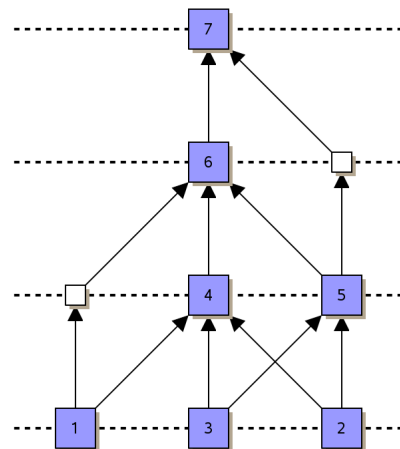
Kreise brechen



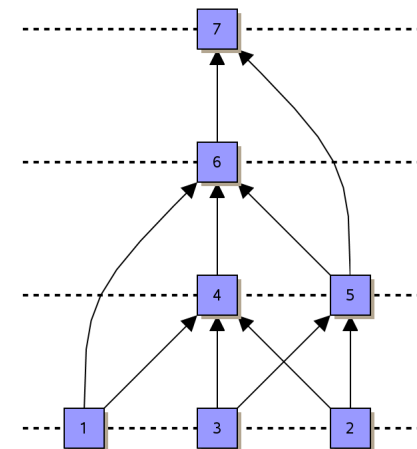
Lagenzuordnung



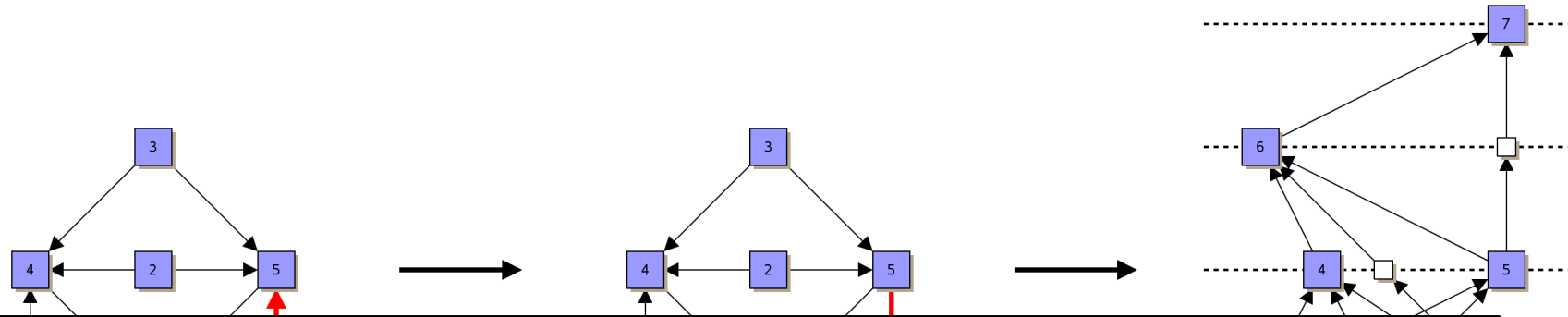
Kreuzungsminimierung



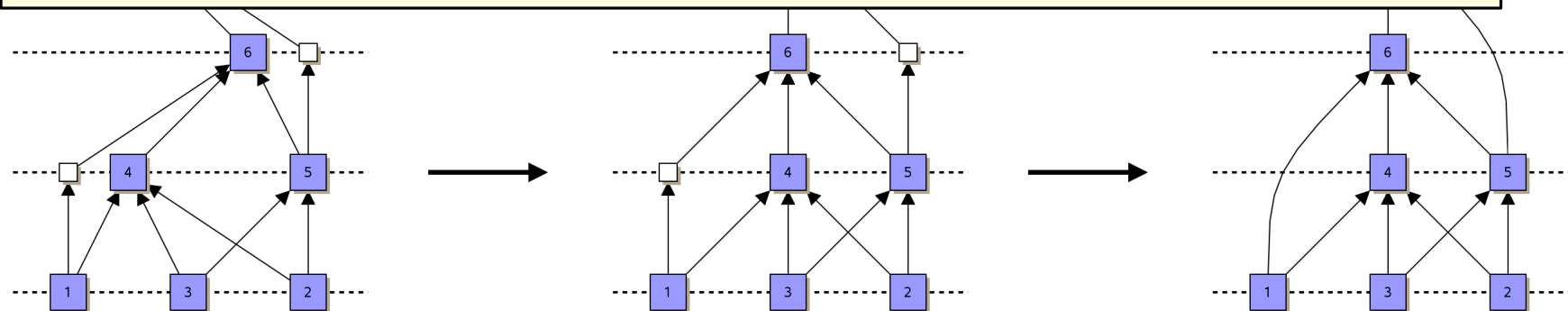
Knotenpositionierung



Kanten zeichnen



- flexibles Framework zum Zeichnen gerichteter Graphen
- sequentielle Optimierung verschiedener Kriterien
- Modularisierung in NP-schwere aber meist einzeln handhabbare Teilprobleme
- dadurch Einschränkung der möglichen Lösungen



Kreuzungsminimierung

Knotenpositionierung

Kanten zeichnen