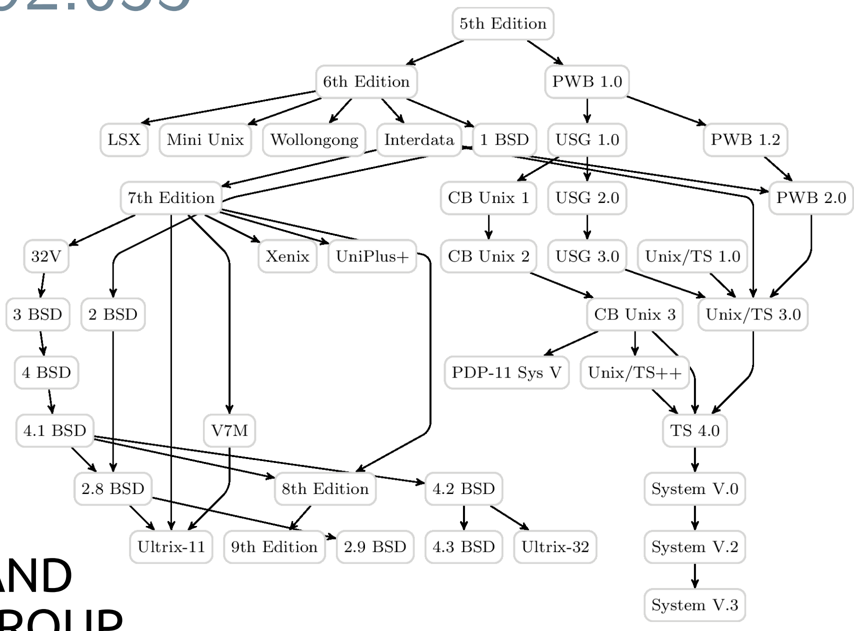
Layered Graph Drawing – Part 2

Lecture Graph Drawing Algorithms · 192.053

Martin Nöllenburg

29.05.2018

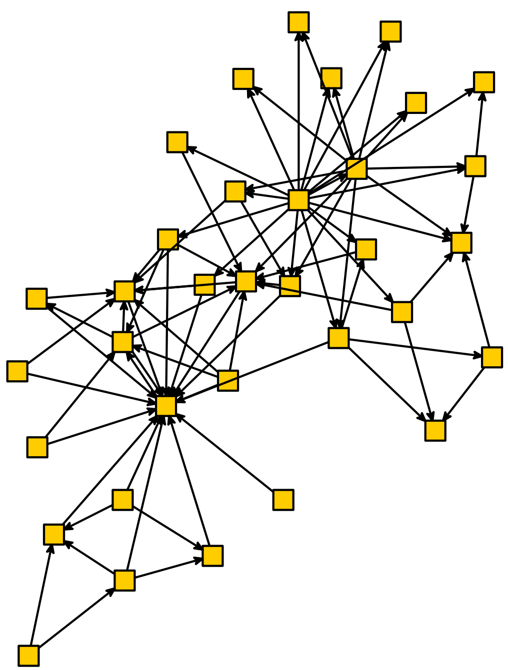


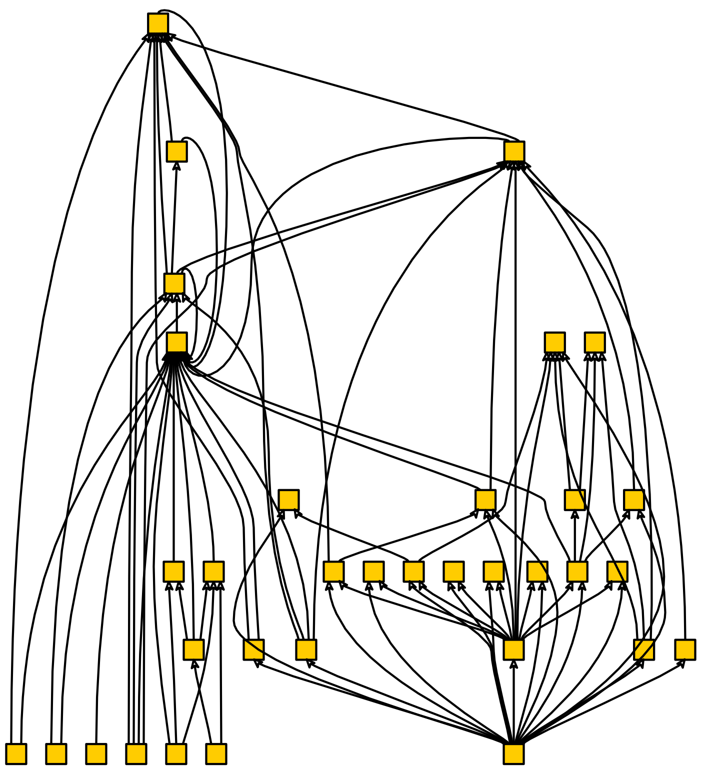
ALGORITHMS AND COMPLEXITY GROUP

Layered Graph Layout

Input: directed graph $D = (V, A)$

Output: drawing of D that emphasizes its hierarchical structure





Layered Graph Layout

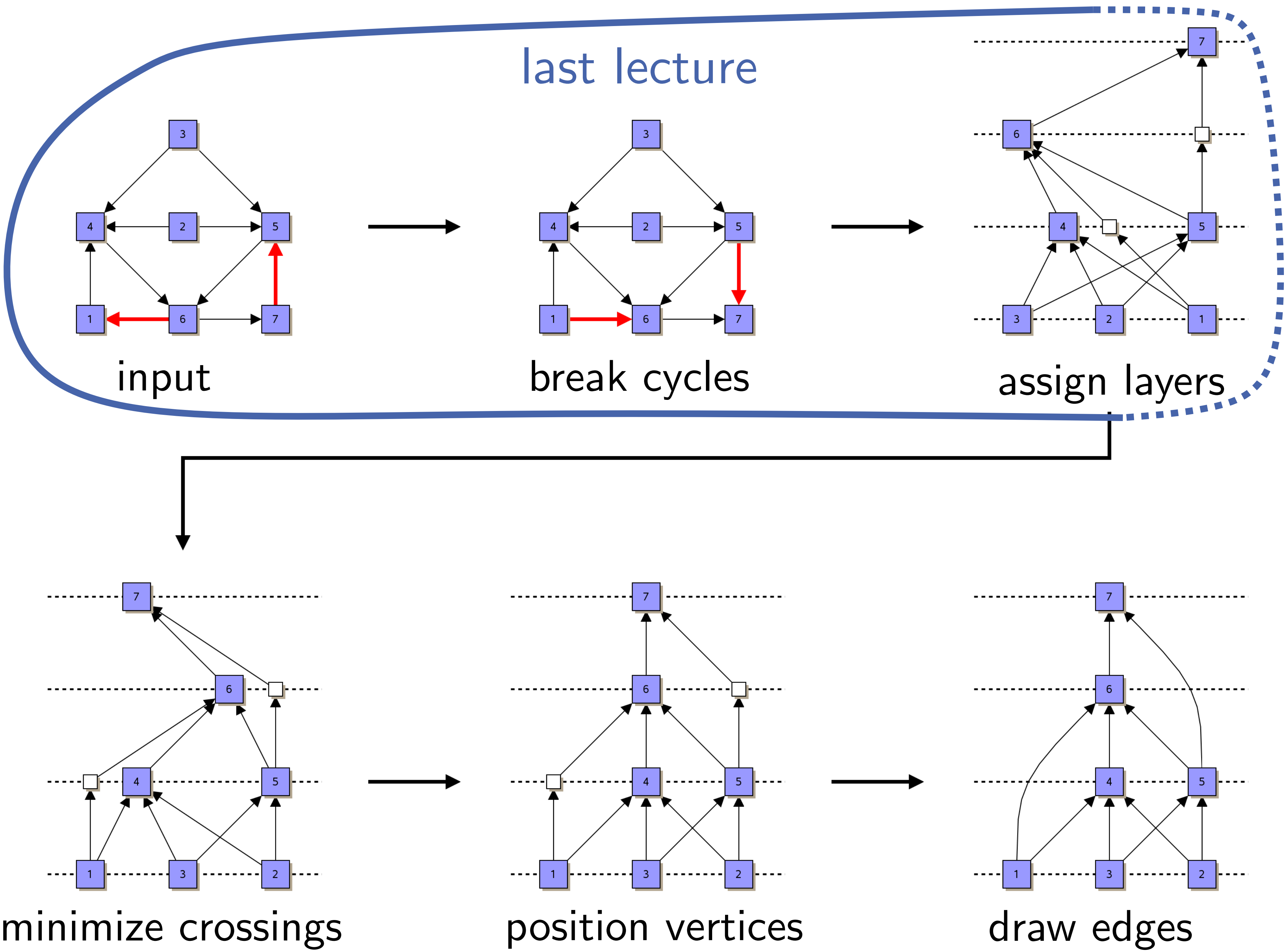
Input: directed graph $D = (V, A)$

Output: drawing of D that emphasizes its hierarchical structure

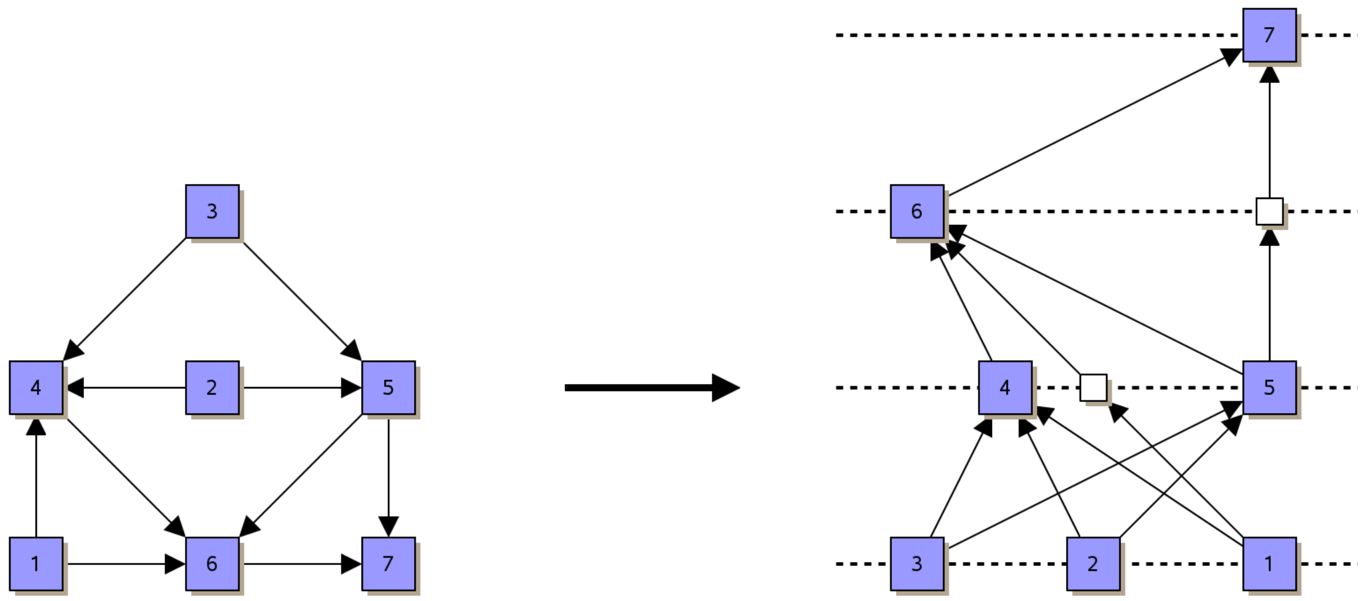
Criteria:

- many edges pointing upward (or some other given direction)
- ideally short, straight and vertical edges
- vertices placed on (few) horizontal layers
- few edge crossings
- evenly distributed vertices

Overview Sugiyama Framework



Step 2: Assign Layers



Layer Assignment

Input: directed acyclic graph $D = (V, A)$

Output: partition of V into disjoint subsets (**layers**) $L_1, \dots, L_h$
s.t. $(u, v) \in A$, $u \in L_i$, $v \in L_j \Rightarrow i < j$

Define: y-Coordinate $y(u) = i \Leftrightarrow u \in L_i$

Layer Assignment

Input: directed acyclic graph $D = (V, A)$

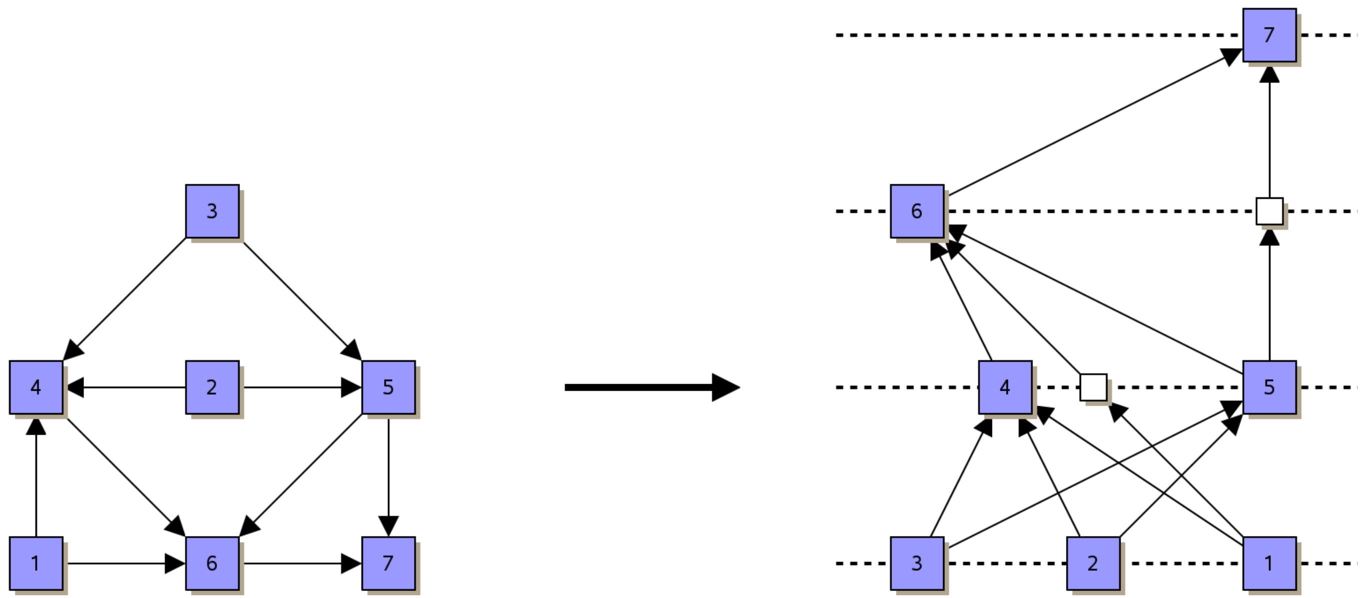
Output: partition of V into disjoint subsets (**layers**) $L_1, \dots, L_h$ s.t. $(u, v) \in A$, $u \in L_i$, $v \in L_j \Rightarrow i < j$

Define: y-Coordinate $y(u) = i \Leftrightarrow u \in L_i$

Some optimization criteria:

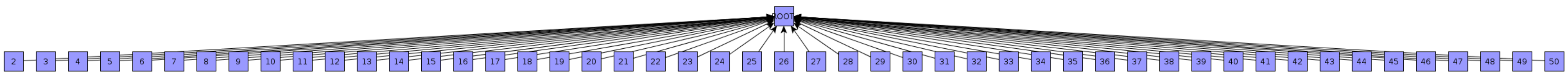
- minimize the number h of layers ($=$ height of the layouts)
- minimize the width, i.e., $\max\{|L_i| \mid 1 \leq i \leq h\}$
- minimize the longest edge, i.e., $\max\{j - i \mid (u, v) \in A,\, u \in L_i,\, v \in L_j\}$
- minimize the total edge length ($\approx$ number of dummy vertices)

Last Lecture

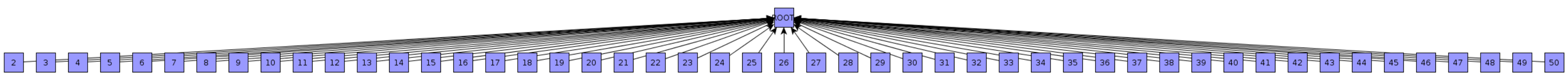


- Height minimization using topological sorting (linear time)
  $\rightarrow$ puts each vertex on lowest possible layer
- Minimization of total edge length using integer linear programming (ILP)
  $\rightarrow$ polynomial time via LP relaxation as constraint matrix is totally unimodular

Height/Edge Length Minimization is not all



Height/Edge Length Minimization is not all

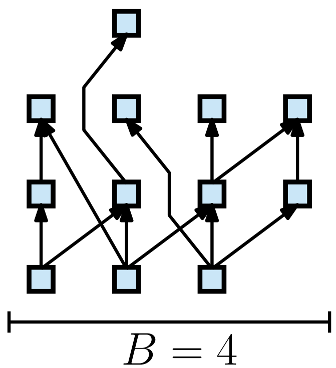


$\rightarrow$ bound the width!

Fixed-Width Layer Assignment

Input: directed acyclic graph $D = (V, A)$, width B

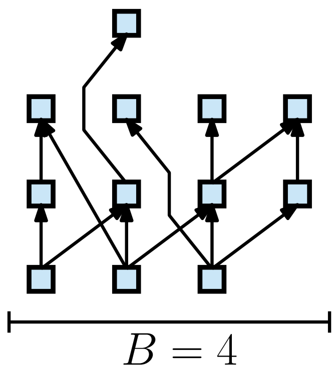
Output: layer assignment $\mathcal{L}$ of minumum height with at most B nodes per layer

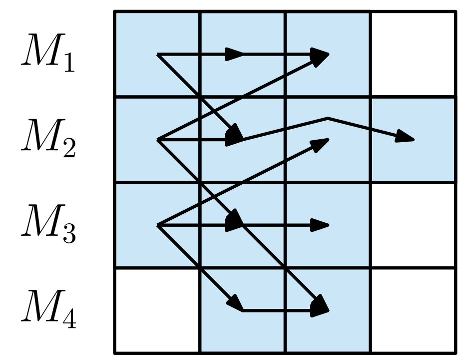


Fixed-Width Layer Assignment

Input: directed acyclic graph $D = (V, A)$, width B

Output: layer assignment $\mathcal{L}$ of minumum height with at most B nodes per layer





$\rightarrow$ this is equivalent to the following job scheduling problem:

Minimum Precedence Constrained Scheduling (MPCS)

Input: n jobs $J_1, \ldots, J_n$ with identical unit processing time, precedence constraints $J_i < J_k$, and B identical machines

Output: Schedule of minimum length that satisfies all the precedence constraints

Complexity

Theorem: For n jobs $J_1, \ldots, J_n$ of equal length, a precedence relation $<$, a number B of identical machines, and an integer T it is NP-complete to decide if a schedule of length at most T exists, even for $T = 3$.

Complexity

Theorem: For n jobs $J_1, \ldots, J_n$ of equal length, a precedence relation $<$, a number B of identical machines, and an integer T it is NP-complete to decide if a schedule of length at most T exists, even for $T = 3$.

Proof:

- reduce NP-complete problem CLIQUE to MPCS with $T = 3$

CLIQUE: Given graph $G = (V, E)$ and $k \in \mathbb{N}$, is there a complete subgraph on $\geq k$ vertices in G?

Define: $k' = |V| - k$ $\quad L = \frac{k \cdot (k-1)}{2}$ (= # edges of Clique) $L' = |E| - L$

$B = \max\{k, L + k', L'\} + 1$

Jobs: J_v for every $v \in V$, J_e for every $e \in E$

for each edge $e = (u, v) \in E$ $\quad J_u < J_e, \; J_v < J_e$

Complexity

Theorem: For n jobs $J_1, \dots, J_n$ of equal length, a precedence relation $<$, a number B of identical machines, and an integer T it is NP-complete to decide if a schedule of length at most T exists, even for $T = 3$.

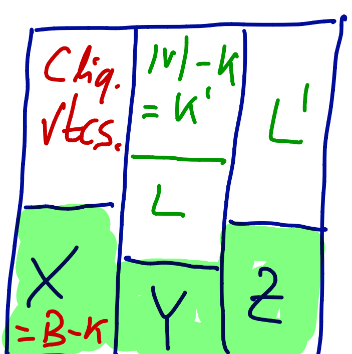
Dummy jobs X_j, $j = 1, \dots, B-K$ Z_j, $j = 1, \dots, B-L'$

Y_j, $j = 1, \dots, B-L-K'$

$X_j < Y_i < Z_\ell$

$\#\text{jobs}: \cancel{|V|} + \cancel{|E|} + (B-\cancel{K}) + (B-\cancel{L}-\underbrace{K'}_{\cancel{|V|-K}}) + (B-\underbrace{L'}_{\cancel{|E|-L}}) = 3B$

claim: G has a k-Clique $\Leftrightarrow$ Schedule of length $T=3$ exists



$\square$

Complexity

Theorem: For n jobs $J_1, \ldots, J_n$ of equal length, a precedence relation $<$, a number B of identical machines, and an integer T it is NP-complete to decide if a schedule of length at most T exists, even for $T = 3$.

Corollary: If $\mathcal{P} \neq \mathcal{NP}$ there is no polynomial-time approximation algorithm for MPCS with approximation ratio $< 4/3$.

Approximation

Theorem: MPCS has a polynomial-time approximation algorithm with approximation ratio $\leq 2 - \frac{1}{B}$.

Approximation

Theorem: MPCS has a polynomial-time approximation algorithm with approximation ratio $\leq 2 - \frac{1}{B}$.

List scheduling algorithm:

$n = \#$ jobs

- order jobs arbitrarily as a list $\mathcal{L}$
- if a machine is free, assign to it the first feasible job in $\mathcal{L}$; if no feasible job exists machine remains idle

observations $\quad OPT \geq \frac{n}{B}$, $\quad OPT \geq \ell$ = longest "path" in the order $<$

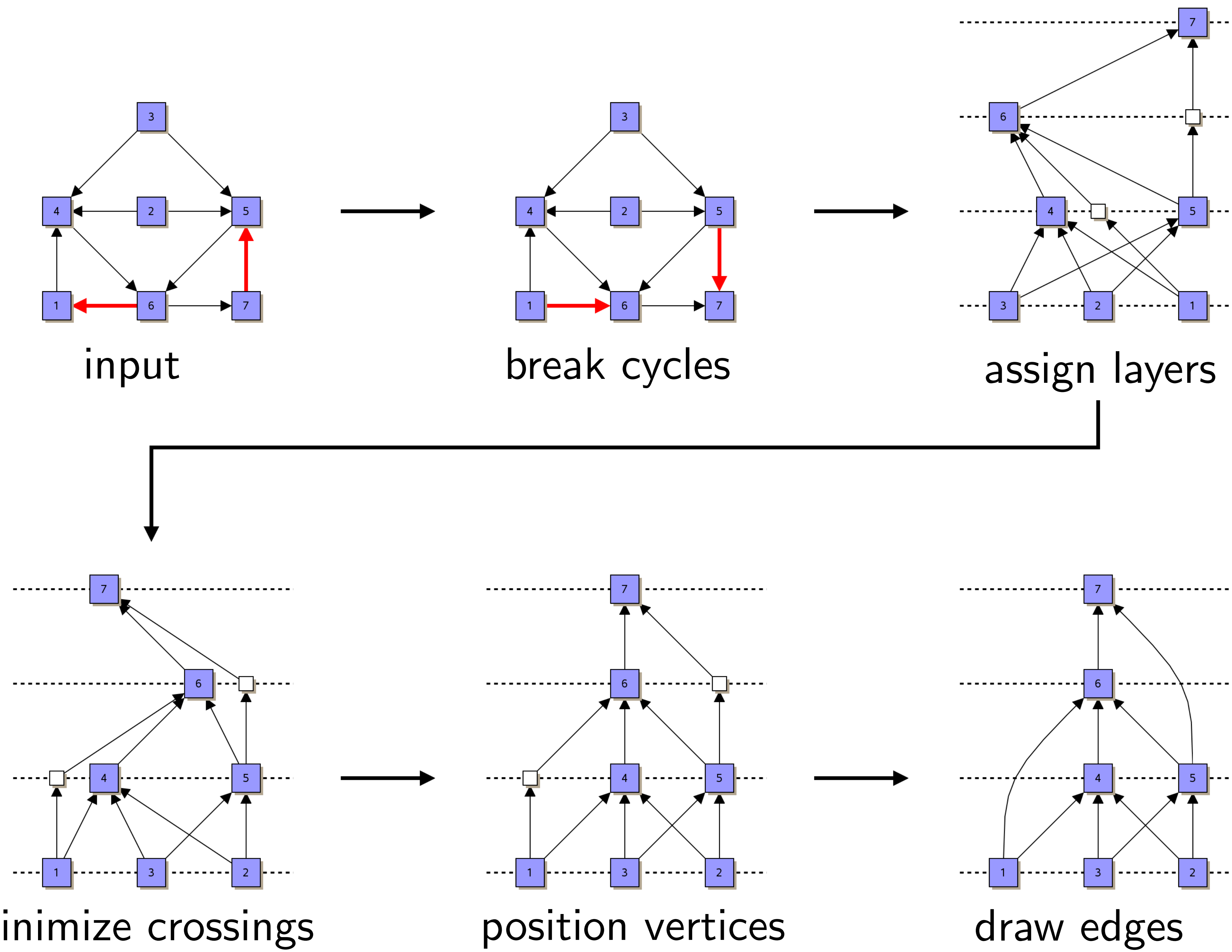
define: m_i = # of idle steps of machine i, observe: $m_i \leq \ell$

w.l.o.g. $m_1 = 0$

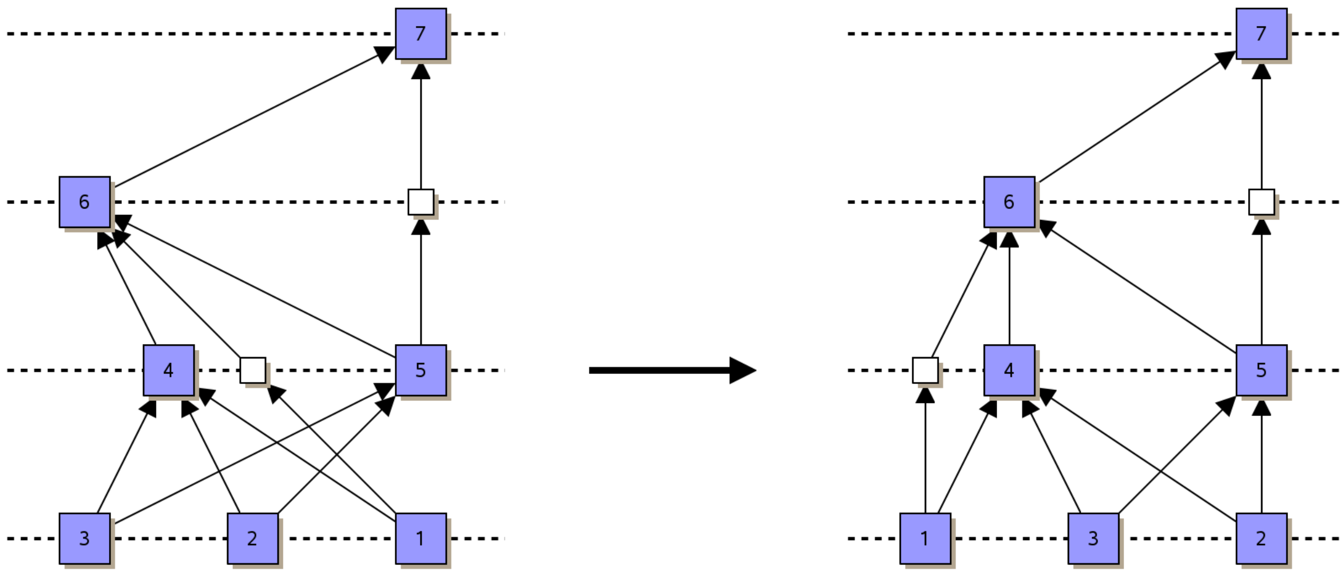
$$T = \frac{1}{B} \cdot \left(n + \sum_{i=2}^{B} m_i\right) \leq \frac{1}{B}\left(n + (B-1)\ell\right) = \frac{n}{B} + \left(1 - \frac{1}{B}\right)\ell$$

$$\leq OPT + \left(1 - \frac{1}{B}\right) OPT$$

Overview


input
break cycles
assign layers
minimize crossings
position vertices
draw edges

Step 3: Crossing Minimization



What would you do?

Problem Statement

Given: DAG $D = (V, A)$, layer assignment of all vertices

Find: Permutation of the vertices on each layer, such that the number of crossing is minimized

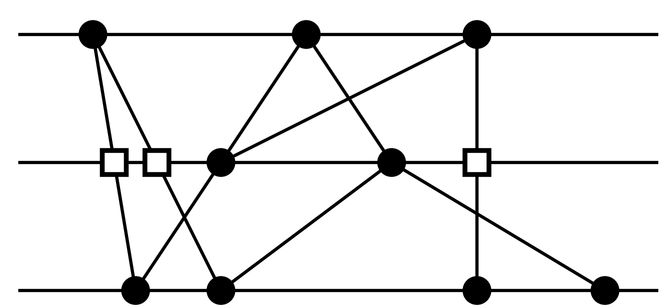
Problem Statement

Given: DAG $D = (V, A)$, layer assignment of all vertices

Find: Permutation of the vertices on each layer, such that the number of crossing is minimized

Properties

- problem is NP-hard even for two layers ($\textsc{Bipartite Crossing Number}$ [Garey, Johnson '83])
- no approaches optimizing several layers simultaneously
- usually iterative optimization for two adjacent layers
  $\rightarrow$ insert dummy vertices in each layer crossed by long edges



One-sided Crossing Minimization (OSCM)

Given: 2-layer graph $G = (L_1, L_2, E)$ and bijective vertex ordering $x_1 \colon L_1 \to \{1, 2, \ldots, |L_1|\}$

Find: vertex ordering $x_2 \colon L_2 \to \{1, 2, \ldots, |L_2|\}$, such that the number of crossing among E is minumum

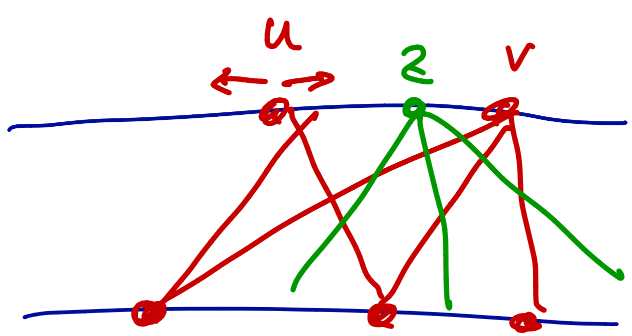
One-sided Crossing Minimization (OSCM)

Given: 2-layer graph $G = (L_1, L_2, E)$ and bijective vertex ordering $x_1 \colon L_1 \to \{1, 2, \ldots, |L_1|\}$

Find: vertex ordering $x_2 \colon L_2 \to \{1, 2, \ldots, |L_2|\}$, such that the number of crossing among E is minumum

Observation:

- number of crossing in 2-layer drawing of G depends only on vertex orderings, not on the exact positions
- for $u, v \in L_2$ the number of crossing among incident edges depends only on $x_2(u) < x_2(v)$ or $x_2(v) < x_2(u)$



One-sided Crossing Minimization (OSCM)

Given: 2-layer graph $G = (L_1, L_2, E)$ and bijective vertex ordering $x_1 \colon L_1 \to \{1, 2, \ldots, |L_1|\}$

Find: vertex ordering $x_2 \colon L_2 \to \{1, 2, \ldots, |L_2|\}$, such that the number of crossing among E is minumum

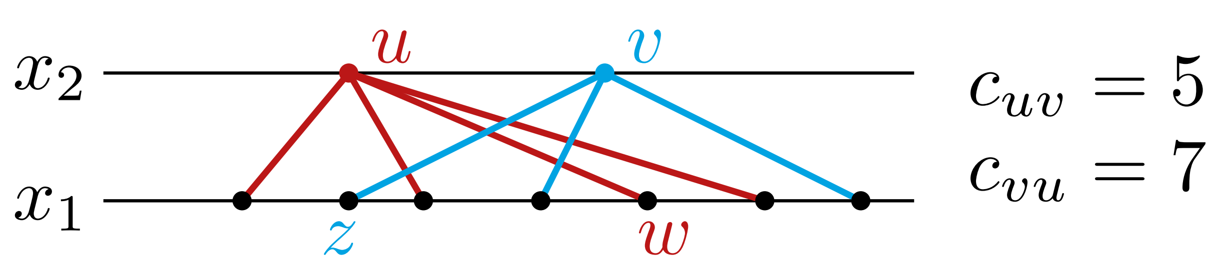
Observation:

- number of crossing in 2-layer drawing of G depends only on vertex orderings, not on the exact positions
- for $u, v \in L_2$ the number of crossing among incident edges depends only on $x_2(u) < x_2(v)$ or $x_2(v) < x_2(u)$

Def: crossing value

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

for $x_2(u) < x_2(v)$



One-sided Crossing Minimization (OSCM)

Given: 2-layer graph $G = (L_1, L_2, E)$ and bijective vertex ordering $x_1 \colon L_1 \to \{1, 2, \ldots, |L_1|\}$

Find: vertex ordering $x_2 \colon L_2 \to \{1, 2, \ldots, |L_2|\}$, such that the number of crossing among E is minumum

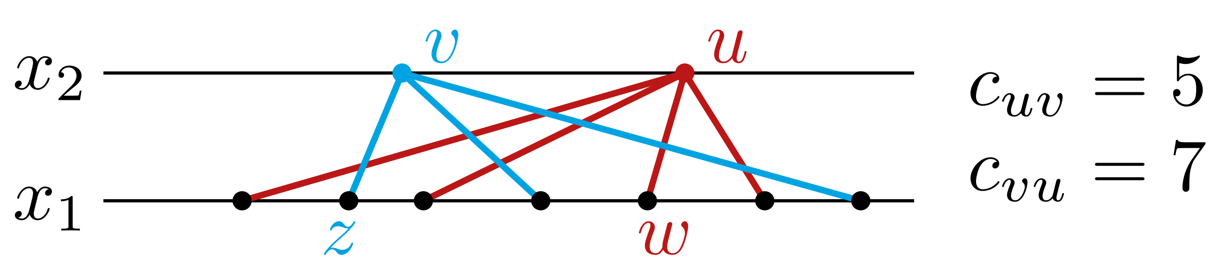
Observation:

- number of crossing in 2-layer drawing of G depends only on vertex orderings, not on the exact positions
- for $u, v \in L_2$ the number of crossing among incident edges depends only on $x_2(u) < x_2(v)$ or $x_2(v) < x_2(u)$

Def: crossing value

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

for $x_2(u) < x_2(v)$



Further Properties

Def: crossing number of G with orders x_1 and x_2 for L_1 and L_2 is denoted by $\mathsf{cr}(G, x_1, x_2)$;
for fixed x_1 let $\mathrm{opt}(G, x_1) = \min_{x_2} \mathsf{cr}(G, x_1, x_2)$

Lemma: The following properties hold:

- $\mathsf{cr}(G, x_1, x_2) = \sum_{x_2(u) < x_2(v)} c_{uv}$
- $\mathrm{opt}(G, x_1) \geq \sum_{u,v \in L_2} \min\{c_{uv}, c_{vu}\}$

Further Properties

Def: crossing number of G with orders x_1 and x_2 for L_1 and L_2 is denoted by $\text{cr}(G, x_1, x_2)$;
for fixed x_1 let $\text{opt}(G, x_1) = \min_{x_2} \text{cr}(G, x_1, x_2)$

Lemma: The following properties hold:

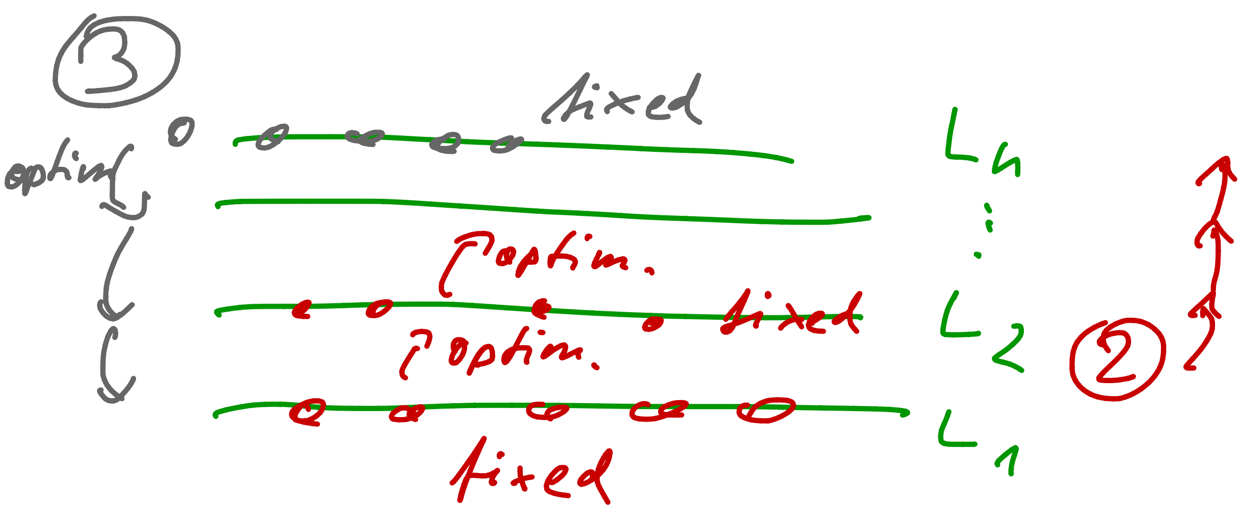
- $\text{cr}(G, x_1, x_2) = \sum_{x_2(u) < x_2(v)} c_{uv}$
- $\text{opt}(G, x_1) \geq \sum_{u,v \in L_2} \min\{c_{uv}, c_{vu}\}$

The value of $\text{cr}(G, x_1, x_2)$ can be computed in $O(n \log n)$ time in a divide-and-conquer algorithm similar to merge sort

Iterative Crossing Minimization

Let $G = (V, E)$ be a DAG with layers $L_1, \dots, L_h$.

1) compute a random ordering x_1 for layer L_1
2) for $i = 1, \dots, h-1$ consider layers L_i and L_{i+1} and minimize $\mathrm{cr}(G, x_i, x_{i+1})$ with fixed x_i ($\rightarrow$ **OSCM**)
3) for $i = h-1, \dots, 1$ consider layers L_{i+1} and L_i and minimize $\mathrm{cr}(G, x_i, x_{i+1})$ with fixed x_{i+1} ($\rightarrow$ **OSCM**)
4) repeat (2) and (3) until no further improvement happens
5) possibly repeat steps (1)–(4) with another x_1
6) return the best found solution



Iterative Crossing Minimization

Let $G = (V, E)$ be a DAG with layers $L_1, \ldots, L_h$.

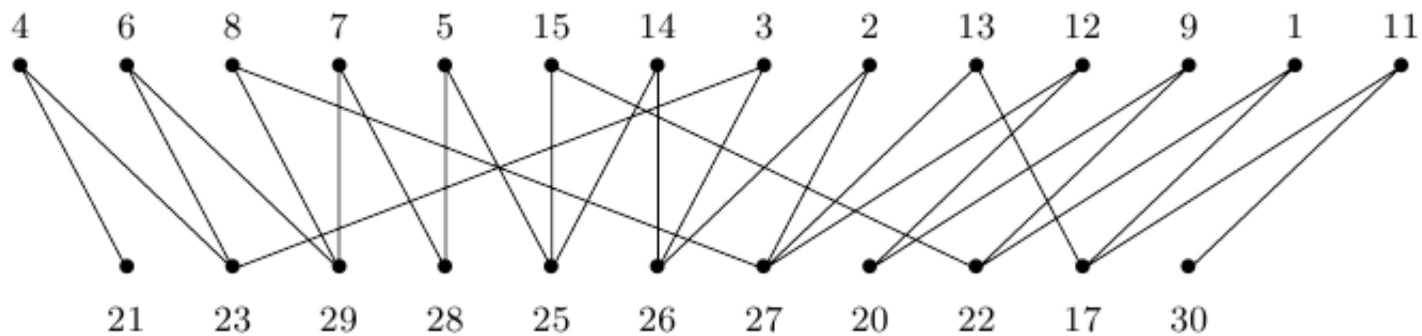
1) compute a random ordering x_1 for layer L_1
2) for $i = 1, \ldots, h-1$ consider layers L_i and L_{i+1} and minimize cr(G, x_i, x_{i+1}) with fixed x_i ($\rightarrow$ **OSCM**)
3) for $i = h-1, \ldots, 1$ consider layers L_{i+1} and L_i and minimize cr(G, x_i, x_{i+1}) with fixed x_{i+1} ($\rightarrow$ **OSCM**)
4) repeat (2) and (3) until no further improvement happens
5) possibly repeat steps (1)–(4) with another x_1
6) return the best found solution

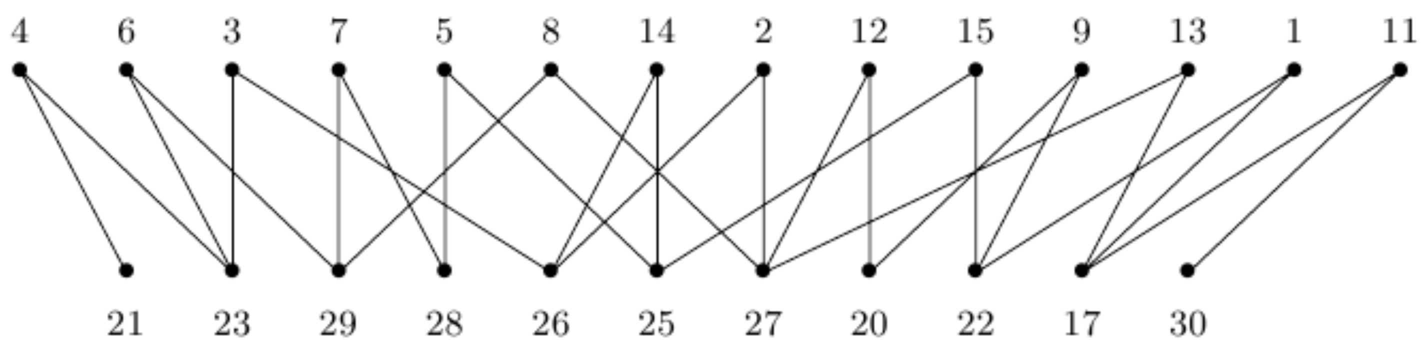
Theorem: The One-Sided Crossing Minimization (OSCM) problem is NP-hard (Eades, Wormald 1994).

Algorithms for OSCM

Heuristics:
- barycenter
- median

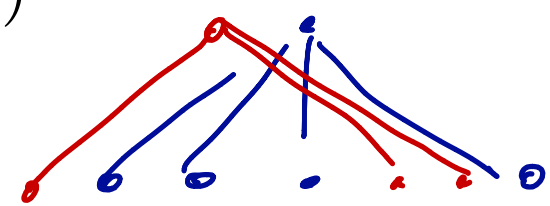
Exact:
- ILP model





Barycenter Heuristic (Sugiyama, Tagawa, Toda 1981)

Idea: few crossings when vertices are close to their neighbors

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


- in case of equality introduce small gap

Barycenter Heuristic (Sugiyama, Tagawa, Toda 1981)

Idea: few crossings when vertices are close to their neighbors

- set

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

- in case of equality introduce small gap

Properties:

- easy to implement
- fast
- usually very good results
- finds optimum if $\mathrm{opt}(G, x_1) = 0$
- may perform $\Theta(\sqrt{n})$ times worse than optimal for some graphs

Barycenter Heuristic (Sugiyama, Tagawa, Toda 1981)

Idea: few crossings when vertices are close to their neighbors

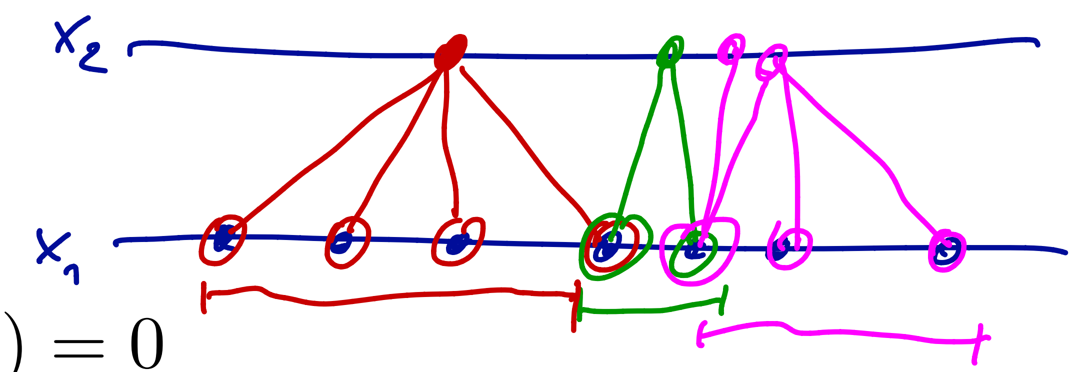
- set

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

- in case of equality introduce small gap

Properties:

- easy to implement
- fast
- usually very good results
- finds optimum if $\mathrm{opt}(G, x_1) = 0$
- may perform $\Theta(\sqrt{n})$ times worse than optimal for some graphs



Why do we find a crossing-free solution if it exists?

Median Heuristic (Eades, Wormald 1994)

Idea: set position to median of the neighbors

- for vertex $v \in L_2$ with neighbors $v_1, \dots, v_k$ set $x_2(v) = \mathrm{med}(v) = x_1(v_{\lceil k/2 \rceil})$ or $x_2(v) = 0$ if $N(v) = \emptyset$
- if $x_2(u) = x_2(v)$ and u, v have different degree parity, place the odd degree vertex to the left
- if $x_2(u) = x_2(v)$ and u, v have the same degree parity, place an arbitrary one to the left
- using linear-time median finding this takes $O(|E|)$ time

Median Heuristic (Eades, Wormald 1994)

Idea: set position to median of the neighbors

- for vertex $v \in L_2$ with neighbors $v_1, \dots, v_k$ set $x_2(v) = \mathrm{med}(v) = x_1(v_{\lceil k/2 \rceil})$ or $x_2(v) = 0$ if $N(v) = \emptyset$
- if $x_2(u) = x_2(v)$ and u, v have different degree parity, place the odd degree vertex to the left
- if $x_2(u) = x_2(v)$ and u, v have the same degree parity, place an arbitrary one to the left
- using linear-time median finding this takes $O(|E|)$ time

Properties:

- easy to implement
- fast
- mostly good performance
- finds optimum when $\mathrm{opt}(G, x_1) = 0$
- **factor-3 approximation**

Median Heuristic: Approximation Factor

Theorem: Let $G = (L_1, L_2, E)$ be a 2-layer graph and x_1 an arbitrary ordering of L_1. Then it holds that

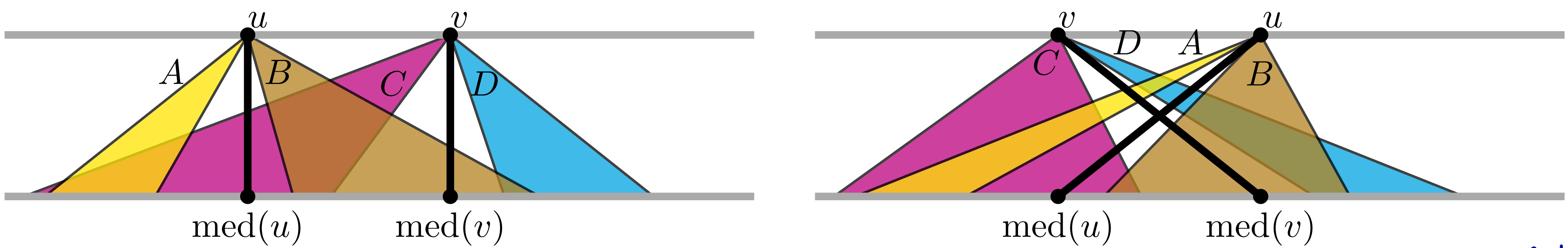
$$\mathrm{med}(G, x_1) \leq 3\,\mathrm{opt}(G, x_1).$$

Median Heuristic: Approximation Factor

Theorem: Let $G = (L_1, L_2, E)$ be a 2-layer graph and x_1 an arbitrary ordering of L_1. Then it holds that

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

Proof: Let $u, v \in L_2$ with $x_2(u) < x_2(v)$



Let $|A| = a$, $|B| = b$, $|C| = c$, $|D| = d$

$$\varepsilon = \begin{cases} 0, & \text{if } \mathrm{med}(u) = \mathrm{med}(v) \\ 1, & \text{otherwise} \end{cases}$$

(1) $c_{vu} \geq \varepsilon + ad + a + d$

(2) $c_{uv} \leq ac + db + bc + c + b$

deg(u) odd: $a = b$ | even: $a + 1 = b$

deg(v) odd: $c = d$ | even: $c + 1 = d$

$\Rightarrow b \leq a + 1$, $c \leq d$

insert into (2)

(3) $c_{uv} \leq ad + ad + d + ad + d + d + a + 1 = 3ad + 3d + a + 1$

Median Heuristic: Approximation Factor

Proof: (cont'd)

1) $c_{vu} \geq ad + a + d + \varepsilon$
2) $c_{uv} \leq ac + bc + bd + c + b$
3) $c_{uv} \leq 3ad + 3d + a + 1$

Show: $c_{uv} \leq 3 c_{vu}$

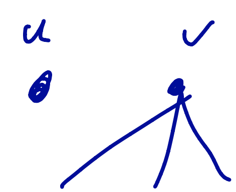
assume $c_{uv} > 3 c_{vu} \Leftrightarrow c_{uv} - 3 c_{vu} > 0$

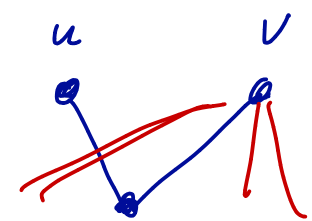
(1),(3) $\Rightarrow \cancel{3ad} + \cancel{3d} + a + 1 - \cancel{3 \cdot (ad} + a + \cancel{d} + \varepsilon) > 0$

$\Leftrightarrow 2a - 1 + 3\varepsilon < 0 \Rightarrow a = \varepsilon = 0 \Rightarrow med(u) = med(v)$

$a = 0 \Rightarrow deg(u) \leq 2$

- if $deg(u) \leq 1$ $\quad c_{uv} - 3 c_{vu} \leq 0$ ✓





$med(u) = med(v)$

- $deg(u) = 2$, $b = 1$ because u is even and left $\Rightarrow deg(v)$ even.

$c + 1 = d$

(1) $a = 0, \varepsilon = 0$
$\Rightarrow c_{vu} \geq d$

(2) $a = 0, b = 1$
$\Rightarrow c_{uv} \leq 2c + d + 1$

$$\left.\begin{array}{l} \\ \end{array}\right\} c_{uv} - 3 c_{vu} \leq 3d - 1 - 3d = -1 \leq 0 \; ↯$$

contradiction

Median Heuristic: Approximation Factor

Proof: (cont'd)

> **Lemma:** The following properties hold:
> - $\mathrm{cr}(G, x_1, x_2) = \sum_{x_2(u)<x_2(v)} c_{uv}$
> - $\mathrm{opt}(G, x_1) \geq \sum_{u,v \in L_2} \min\{c_{uv}, c_{vu}\}$

$$\mathrm{med}(G, x_1) = \mathrm{cr}(G, x_1, x_2) = \sum_{x_2(u)<x_2(v)} c_{uv} \leq 3 \cdot \sum_{x_2(u)<x_2(v)} \min\{c_{uv}, c_{vu}\} \leq 3 \cdot \mathrm{opt}(G, x_1)$$

x_2: computed order

□

Integer Linear Programming

Properties:

- branch-and-cut technique für DAGs of bounded size
- finds optimal solution
- no guarantee to find solution in polynomial time
- suitable for small to medium-size graphs

Integer Linear Programming

Properties:

- branch-and-cut technique für DAGs of bounded size
- finds optimal solution
- no guarantee to find solution in polynomial time
- suitable for small to medium-size graphs

ILP model:

- define arbitrary order $<$ of L_2
- binary variables $x_{uv} = \begin{cases} 1 & \text{if } u \text{ left of } v \\ 0 & \text{otherwise} \end{cases}$

Integer Linear Programming

Properties:

- branch-and-cut technique für DAGs of bounded size
- finds optimal solution
- no guarantee to find solution in polynomial time
- suitable for small to medium-size graphs

ILP model:

- define arbitrary order $<$ of L_2
- binary variables $x_{uv} = \begin{cases} 1 & \text{if } u \text{ left of } v \\ 0 & \text{otherwise} \end{cases}$

$$\mathrm{cr}(G, x_1, x_2) = \sum_{u<v} (c_{uv} x_{uv} + c_{vu}(1 - x_{uv})) = \sum_{u<v} (c_{uv} - c_{vu}) x_{uv} + \sum_{u<v} c_{vu}$$

Integer Linear Programming

Properties:

- branch-and-cut technique für DAGs of bounded size
- finds optimal solution
- no guarantee to find solution in polynomial time
- suitable for small to medium-size graphs

ILP model:

- define arbitrary order $<$ of L_2
- binary variables $x_{uv} = \begin{cases} 1 & \text{if } u \text{ left of } v \\ 0 & \text{otherwise} \end{cases}$

$$\text{cr}(G, x_1, x_2) = \sum_{u<v} (c_{uv} x_{uv} + c_{vu}(1 - x_{uv})) = \sum_{u<v} (c_{uv} - c_{vu}) x_{uv} + \sum_{u<v} c_{vu}$$

minimize $\sum_{u<v} (c_{uv} - c_{vu}) x_{uv}$ s.t.

- $x_{uv} \in \{0, 1\}$ for all $u < v \in L_2$
- transitivity

How to model transitivity in ILP?

Integer Linear Programming

Properties:

- branch-and-cut technique für DAGs of bounded size
- finds optimal solution
- no guarantee to find solution in polynomial time
- suitable for small to medium-size graphs

ILP model:

- define arbitrary order $<$ of L_2
- binary variables $x_{uv} = \begin{cases} 1 & \text{if } u \text{ left of } v \\ 0 & \text{otherwise} \end{cases}$

$$\mathrm{cr}(G, x_1, x_2) = \sum_{u<v} (c_{uv} x_{uv} + c_{vu}(1 - x_{uv})) = \sum_{u<v} (c_{uv} - c_{vu}) x_{uv} + \sum_{u<v} c_{vu}$$

minimize $\sum_{u<v} (c_{uv} - c_{vu}) x_{uv}$ s.t.

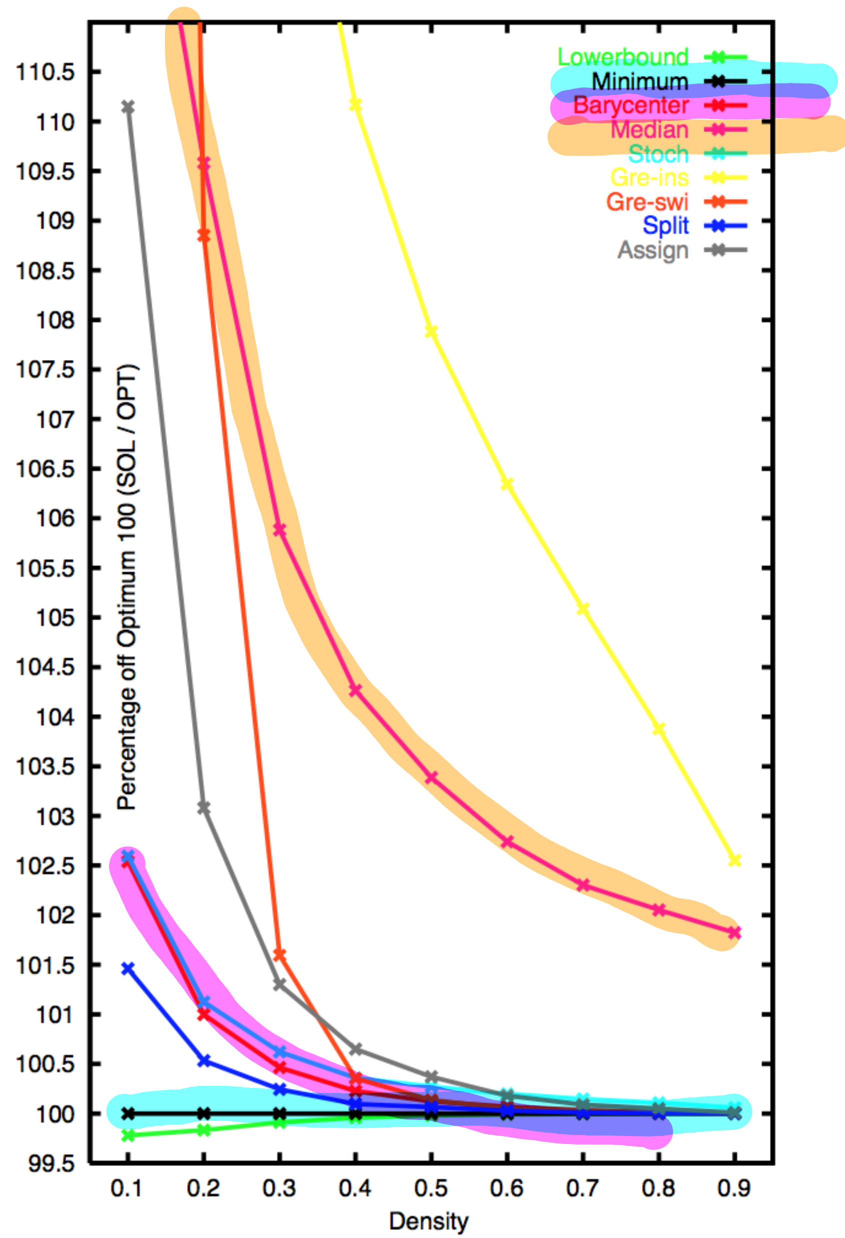
- $x_{uv} \in \{0, 1\}$ for all $u < v \in L_2$
- $0 \le x_{uv} + x_{vw} - x_{uw} \le 1$ for all $u < v < w \in L_2$

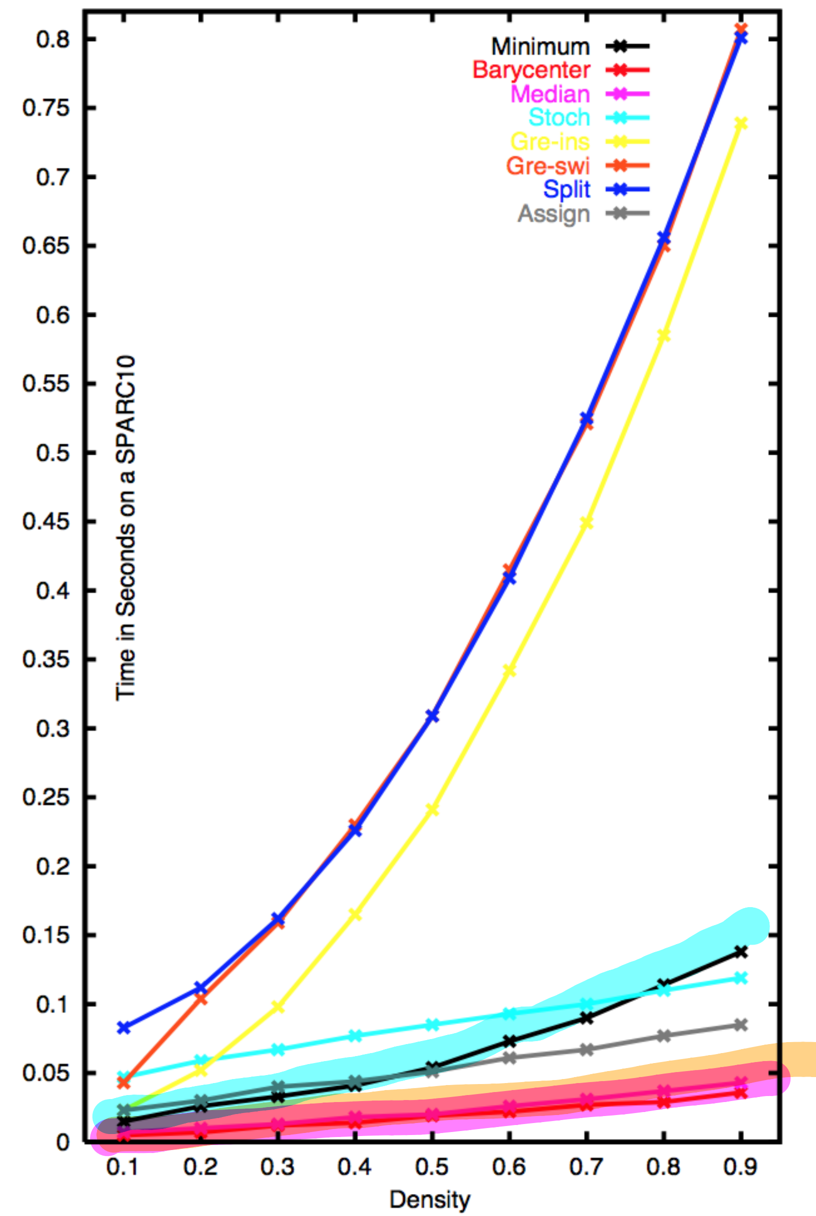
$u \quad v \quad w$

$u < v$ und $v < w$

$\Rightarrow u < w$

Experimental Evaluation (Jünger, Mutzel 1997)

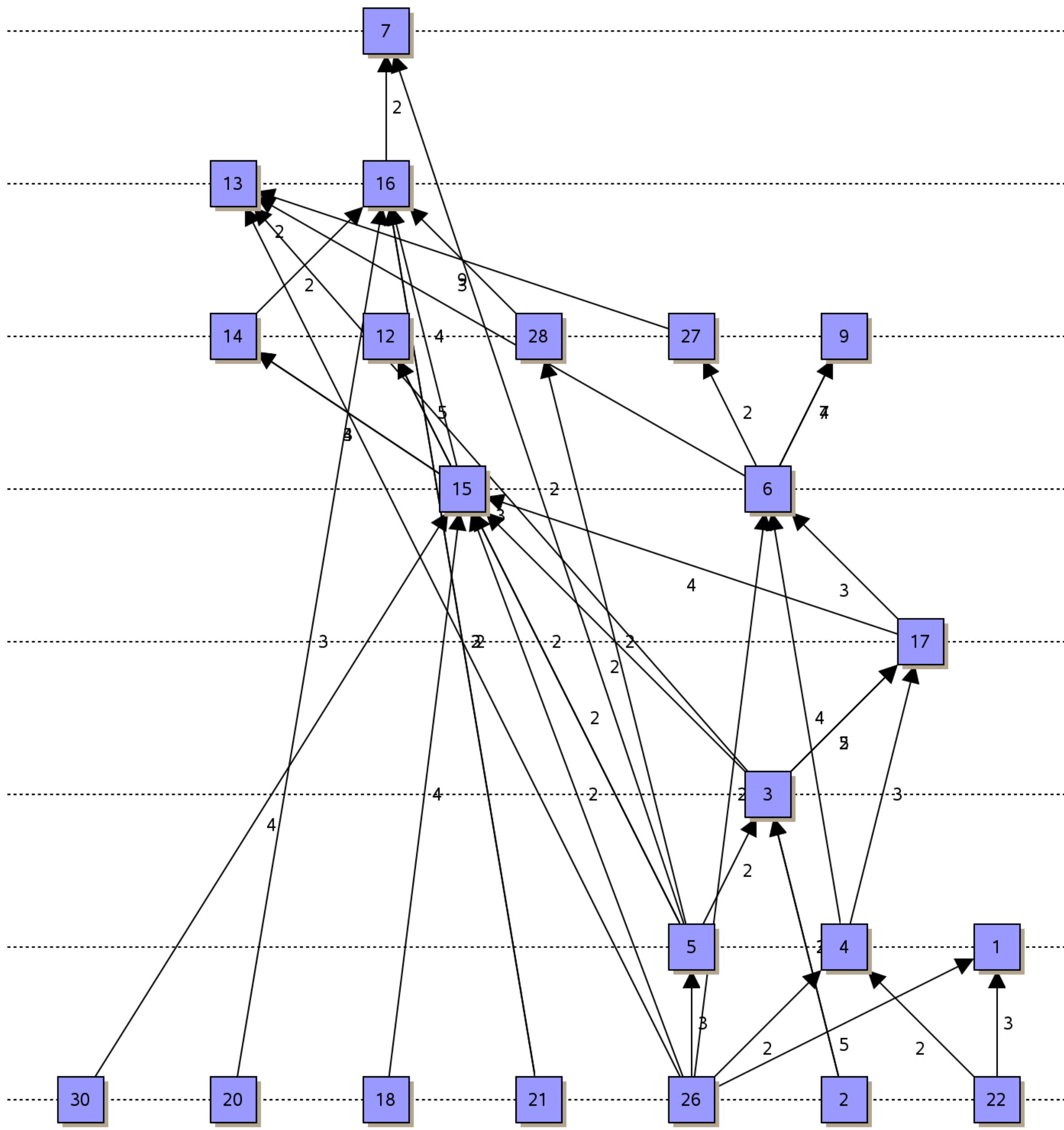




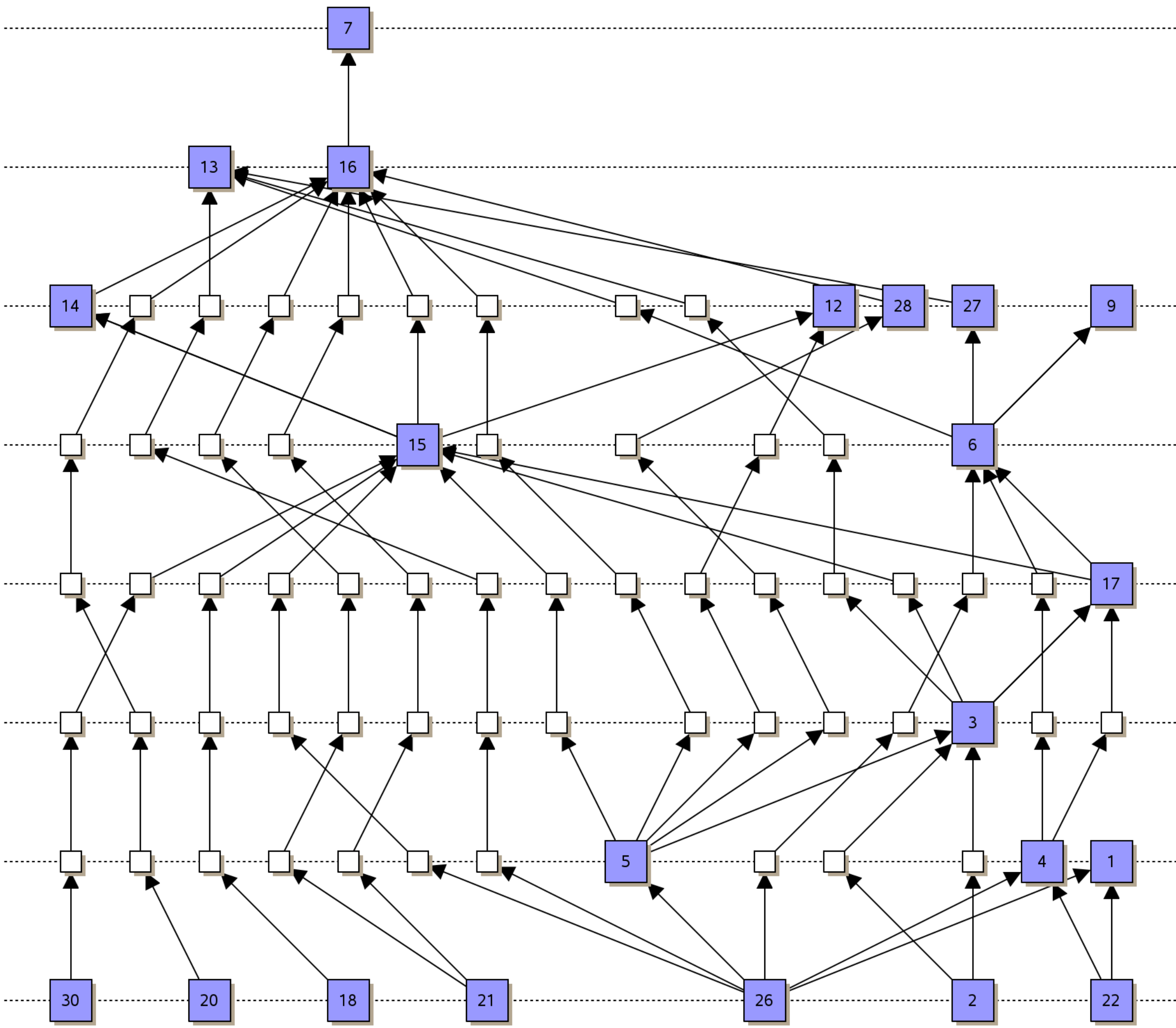
Quality averaged over 100 instances on 20 + 20 vertices bipartite graphs with increasing density

Running times on the same instances

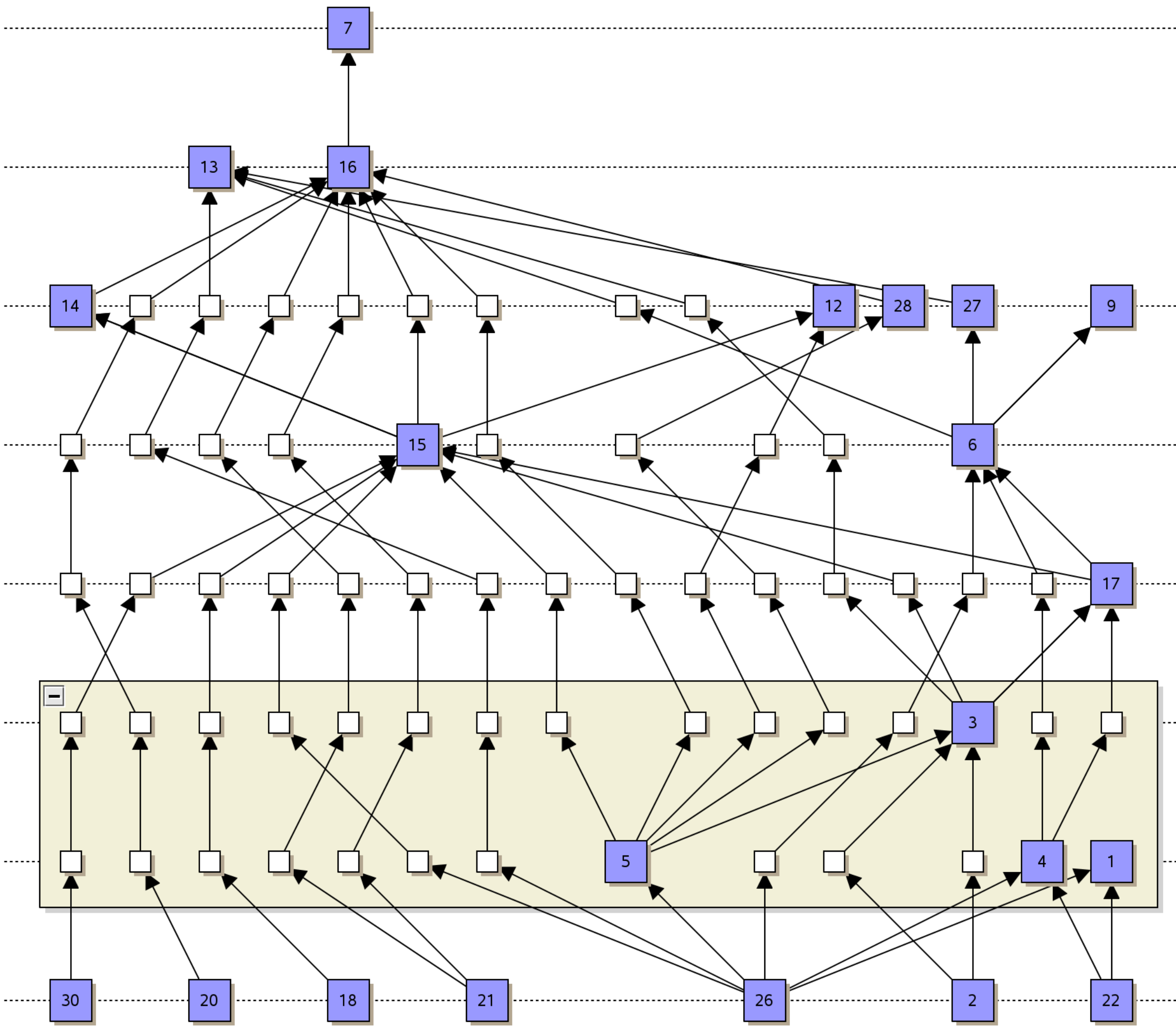
Example



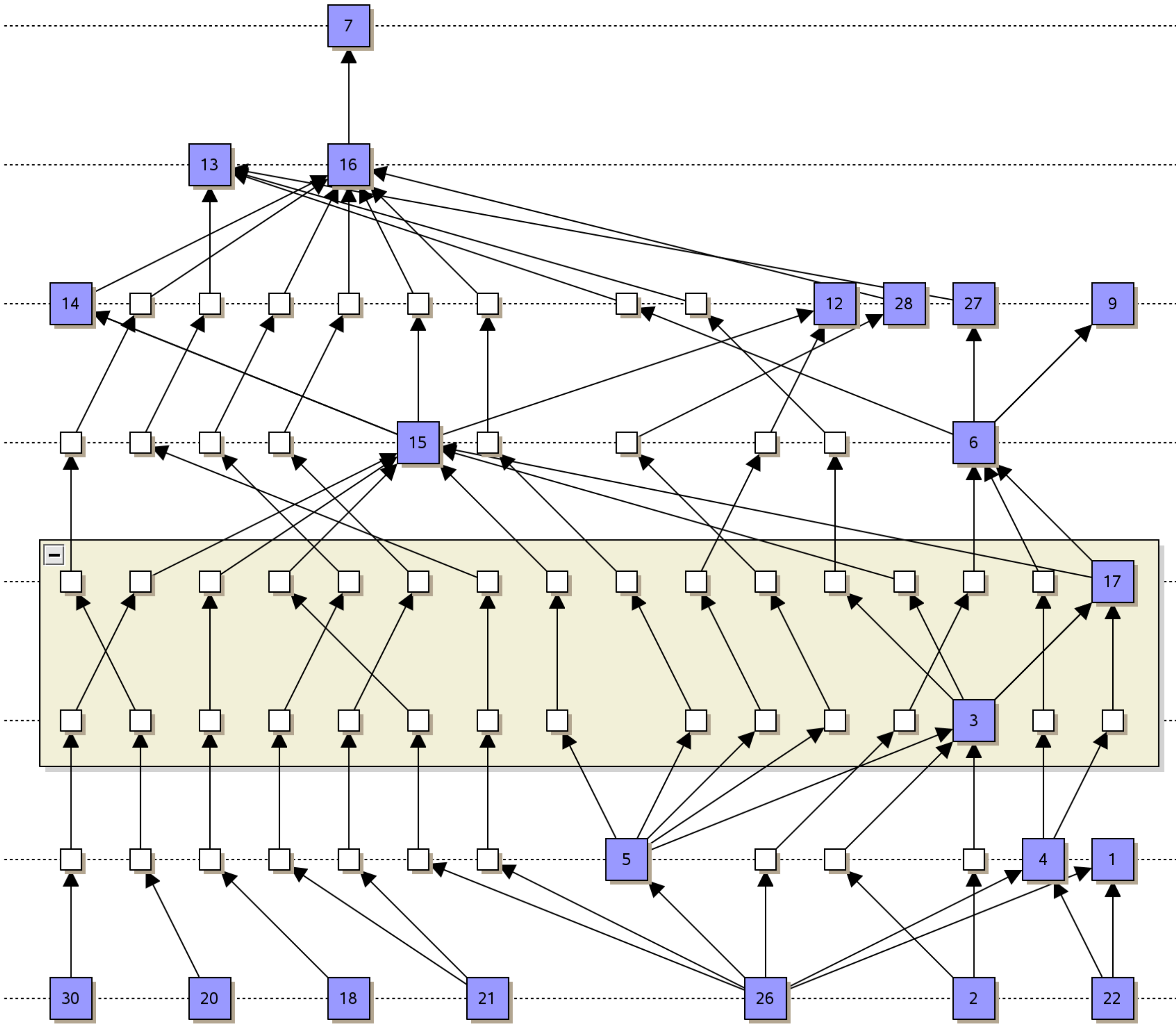
Example



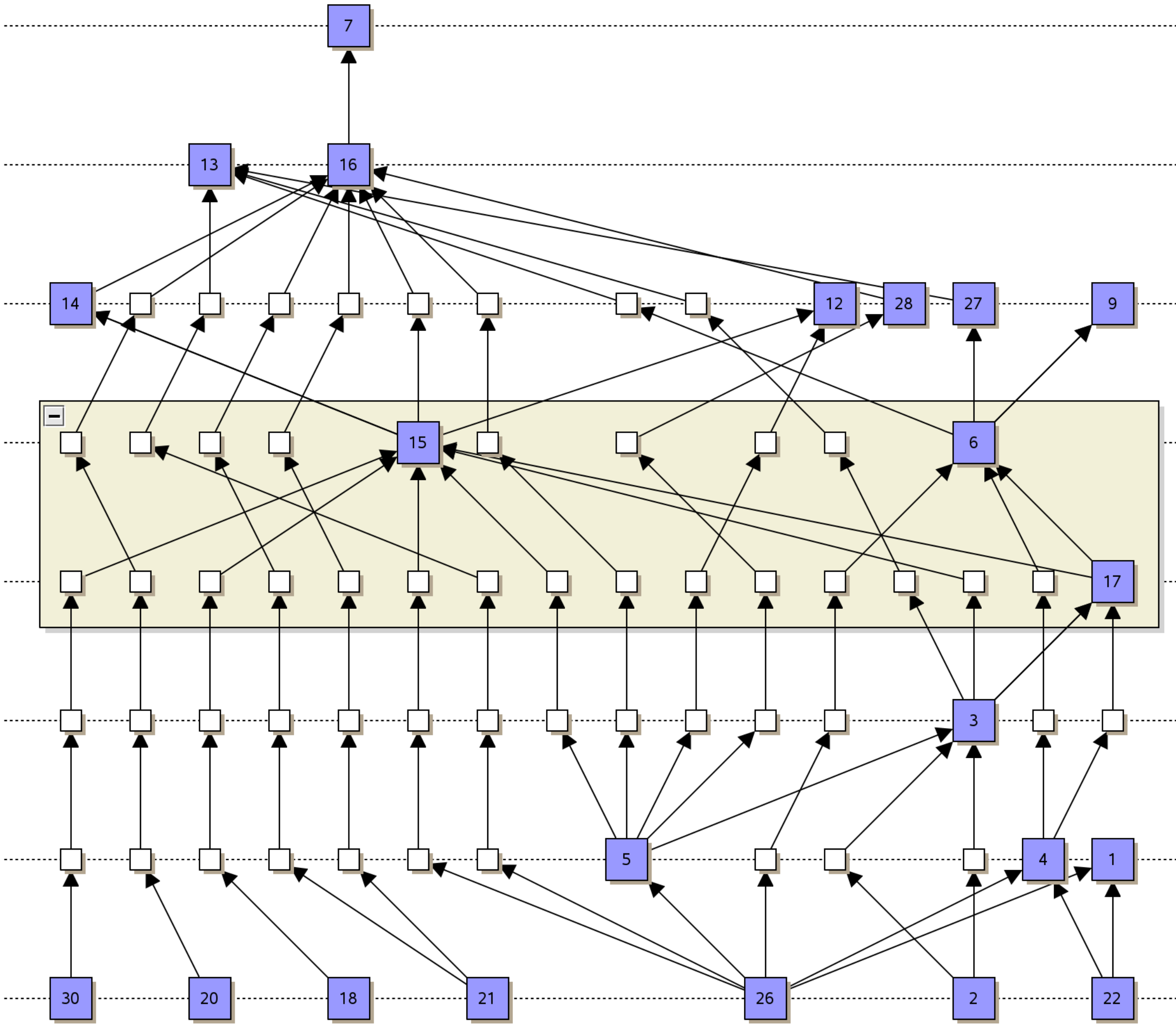
Example



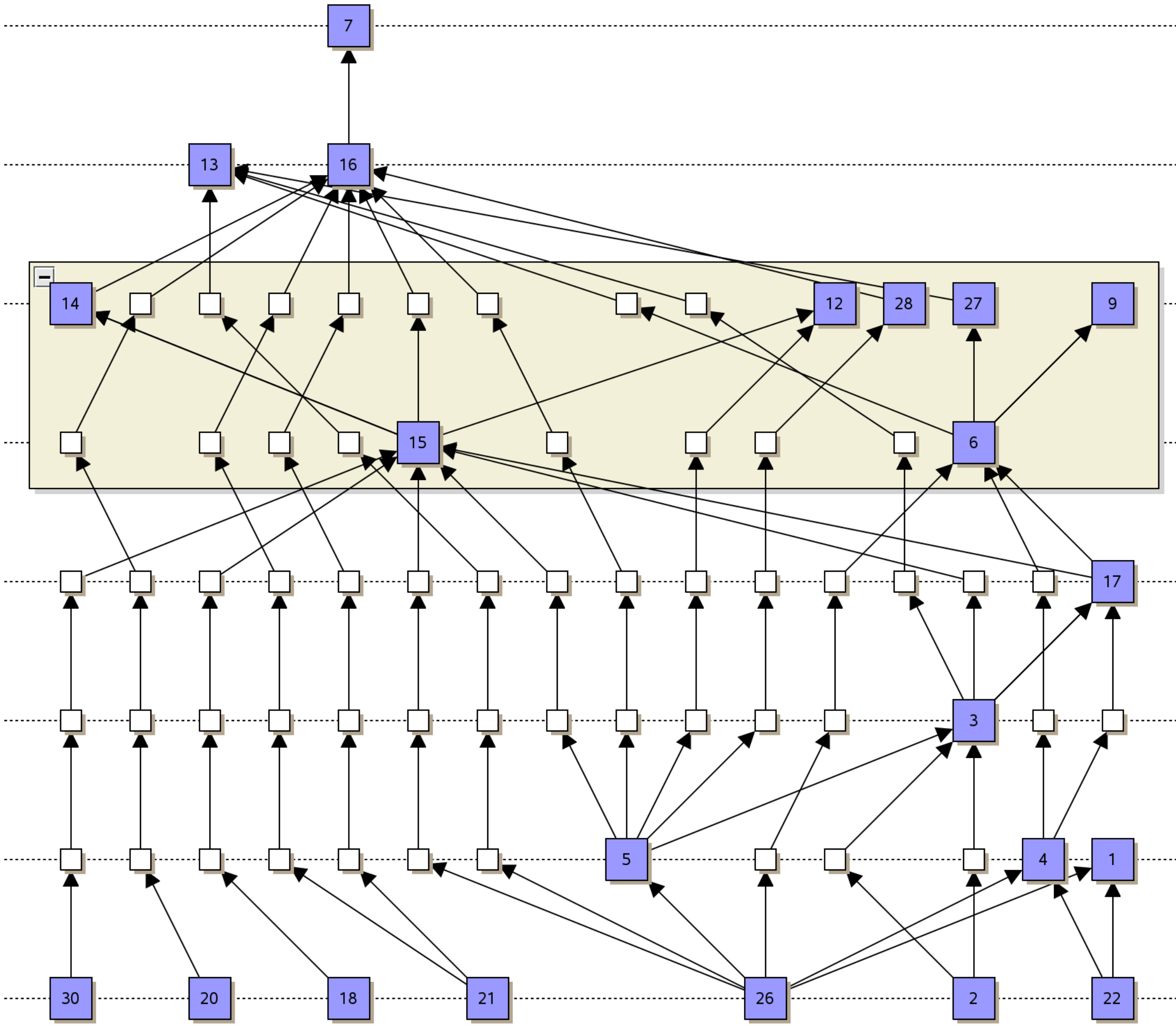
Example



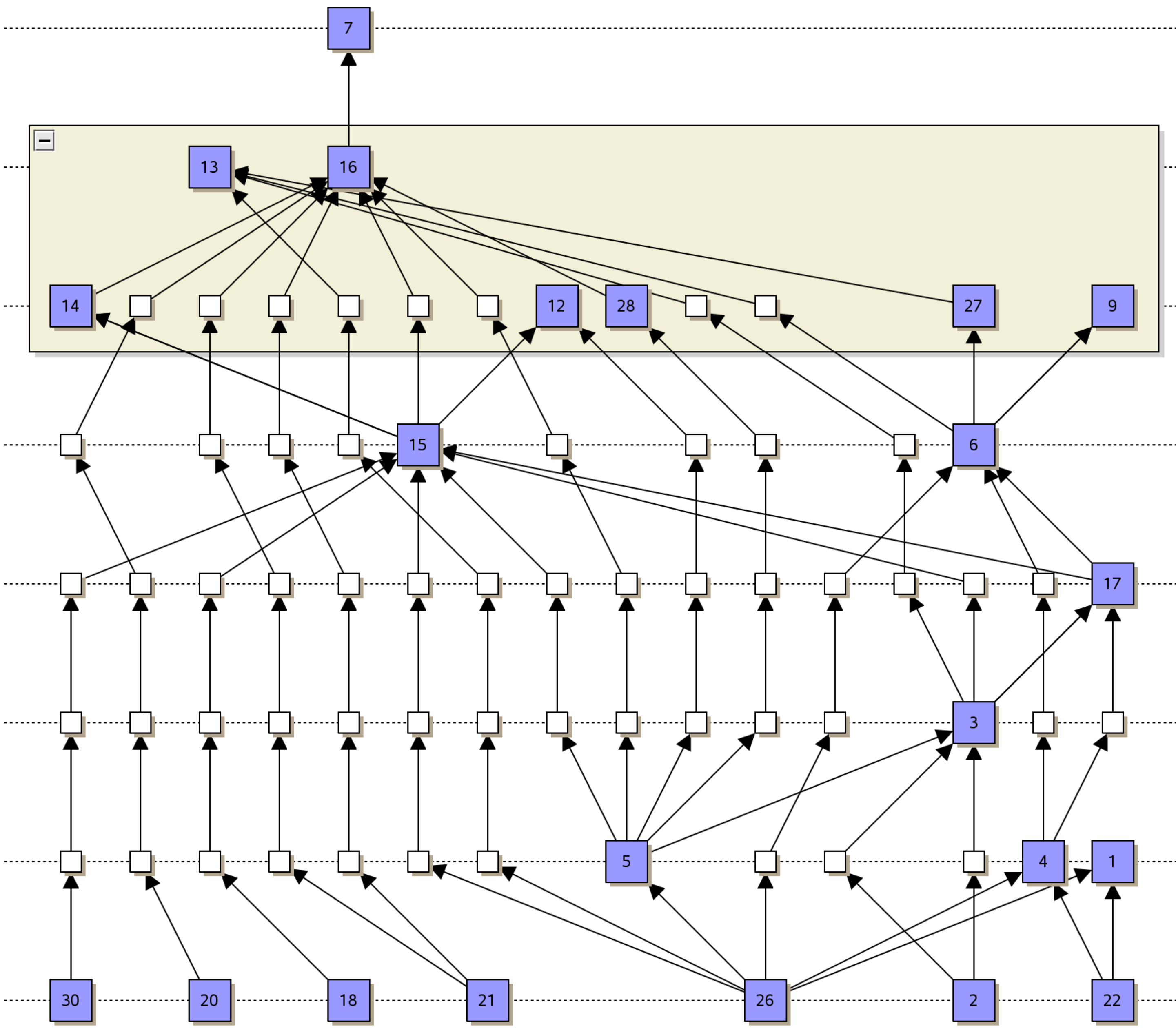
Example



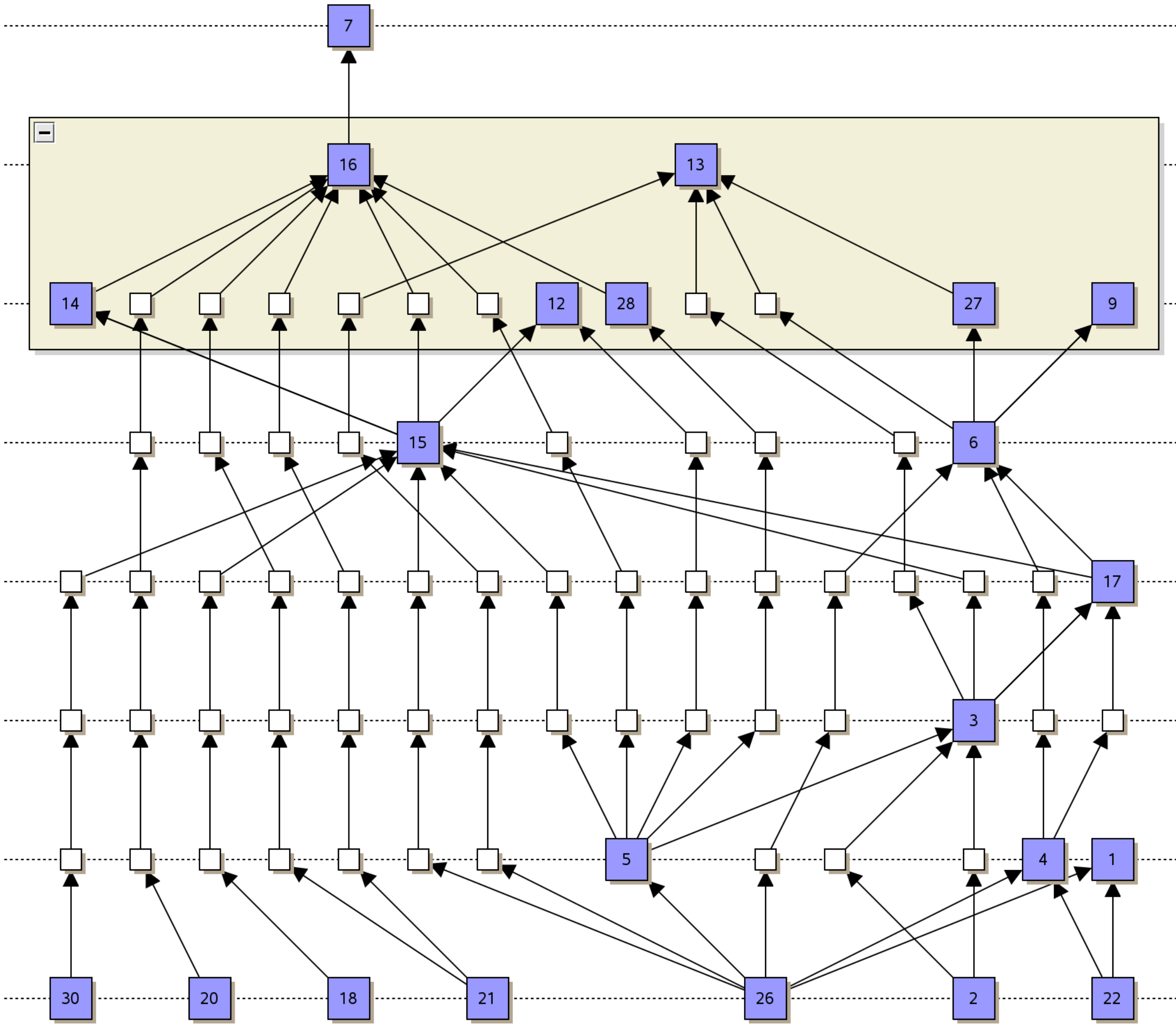
Example



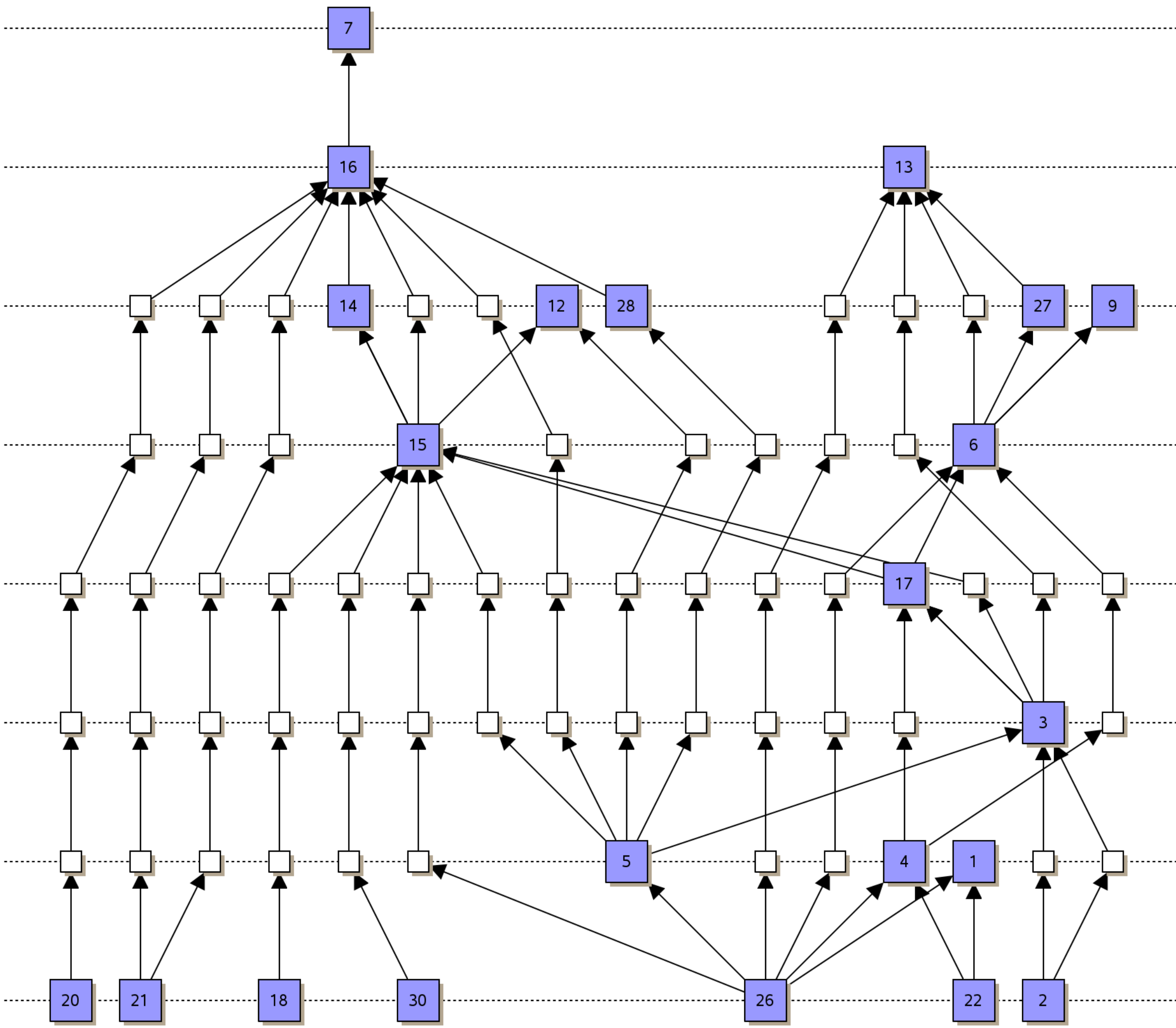
Example



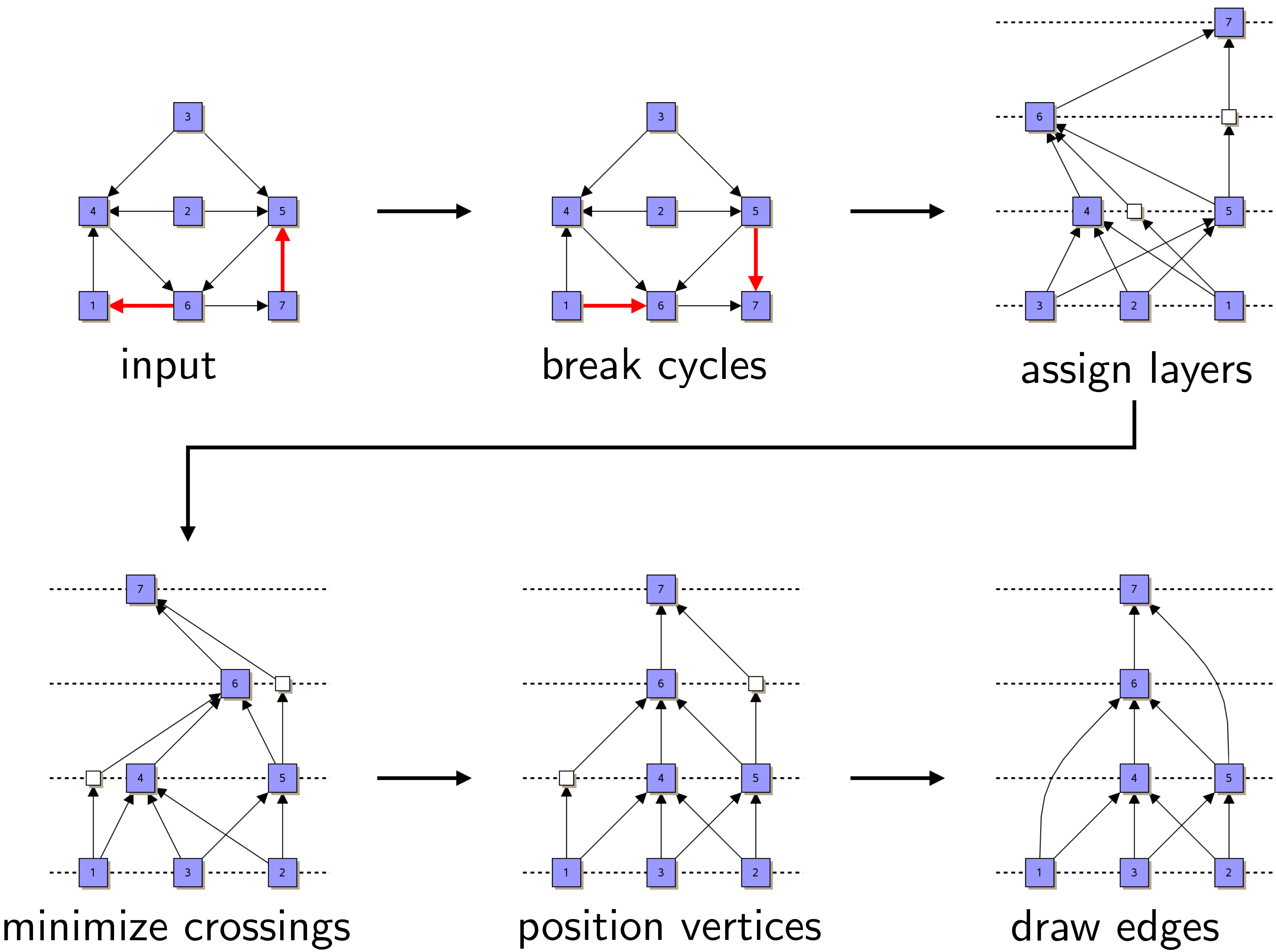
Example



Example

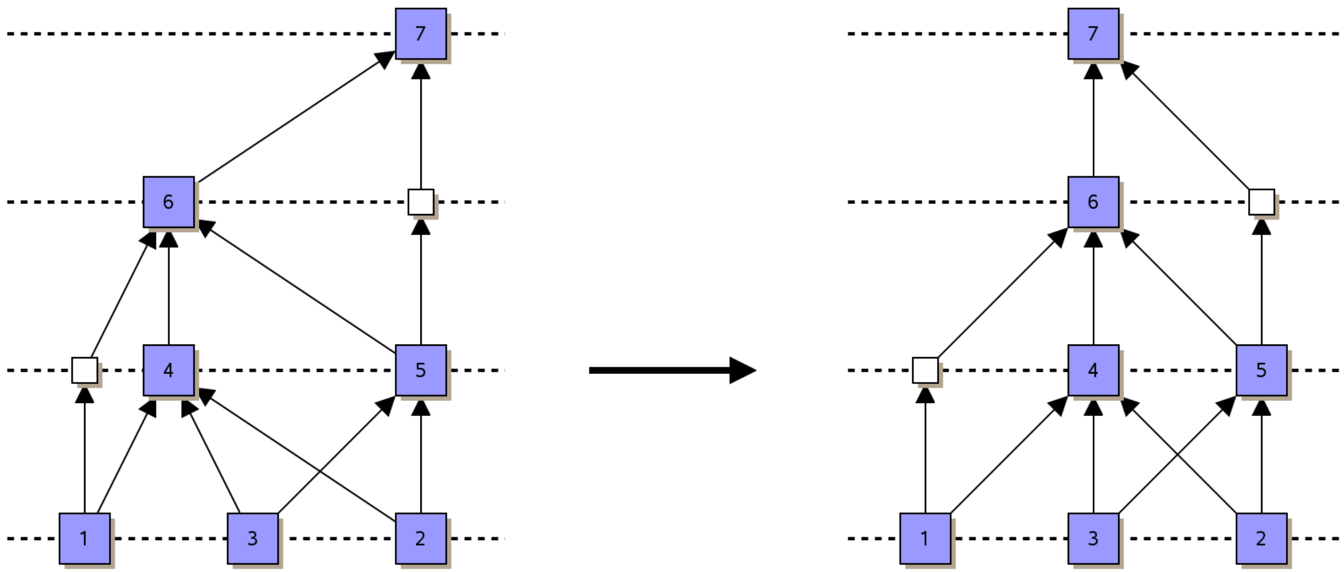


Overview



input
break cycles
assign layers
minimize crossings
position vertices
draw edges

Step 4: Coordinate Assignment



What are the goals?

Edge Straightening

Goal: minimize deviation from straight line for edges with dummy vertices

Idea: quadratic programming

- let $p_{uv} = (u, d_1, \ldots, d_k, v)$ be a path with k dummy vertices between u and v
- let $a_i = x(u) + \frac{i}{k+1}(x(v) - x(u))$ be the x-coordinate of d_i assuming straight line
- minimize $\sum_{i=1}^{k}(x(d_i) - a_i)^2$ over all paths
- constraints: $x(w) - x(z) \geq \delta$ for all vertices on the same layer with w right of z (δ is a spacing parameter)

Edge Straightening

Goal: minimize deviation from straight line for edges with dummy vertices

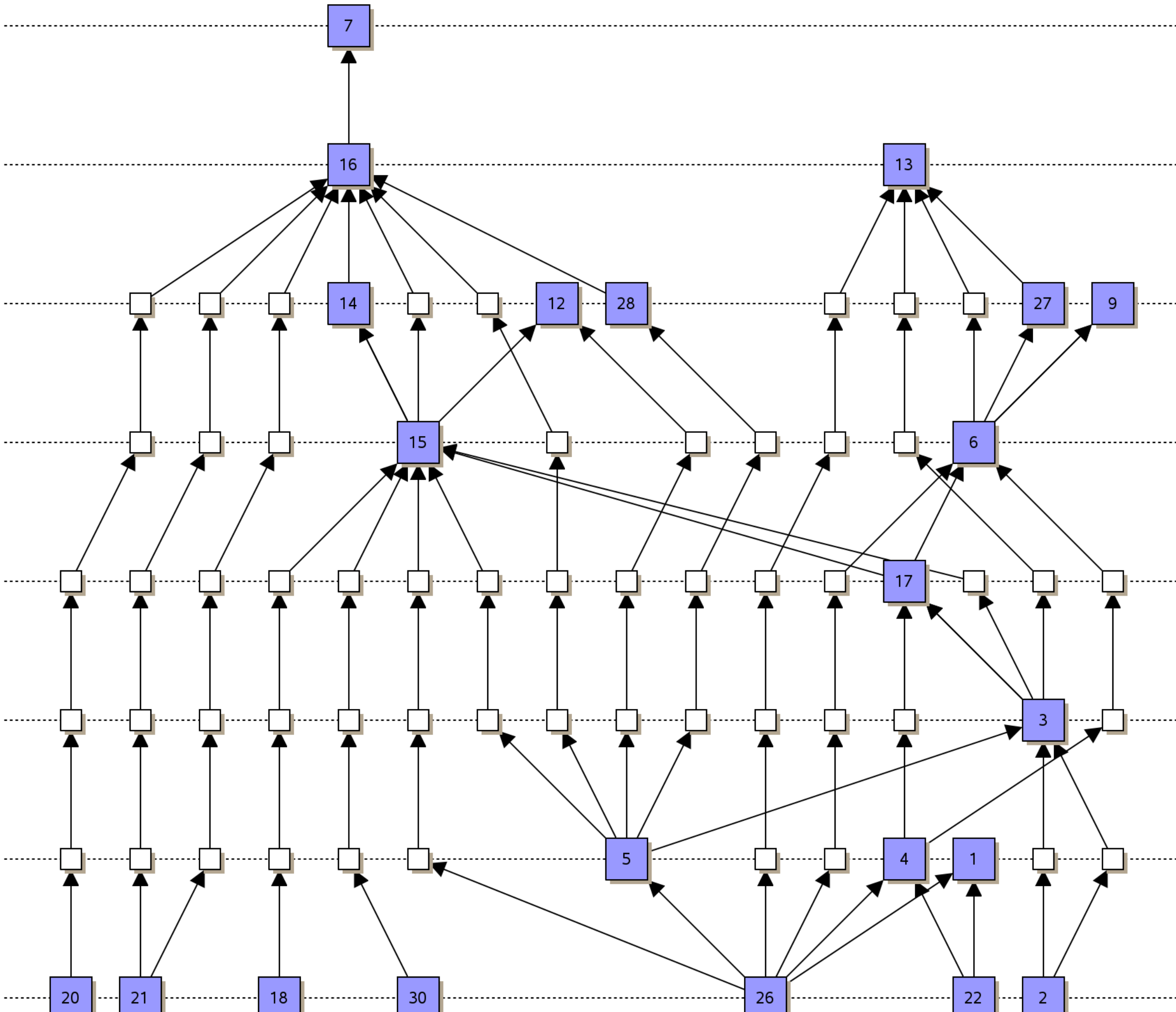
Idea: quadratic programming

- let $p_{uv} = (u, d_1, \ldots, d_k, v)$ be a path with k dummy vertices between u and v
- let $a_i = x(u) + \frac{i}{k+1}(x(v) - x(u))$ be the x-coordinate of d_i assuming straight line
- minimize $\sum_{i=1}^{k}(x(d_i) - a_i)^2$ over all paths
- constraints: $x(w) - x(z) \geq \delta$ for all vertices on the same layer with w right of z (δ is a spacing parameter)

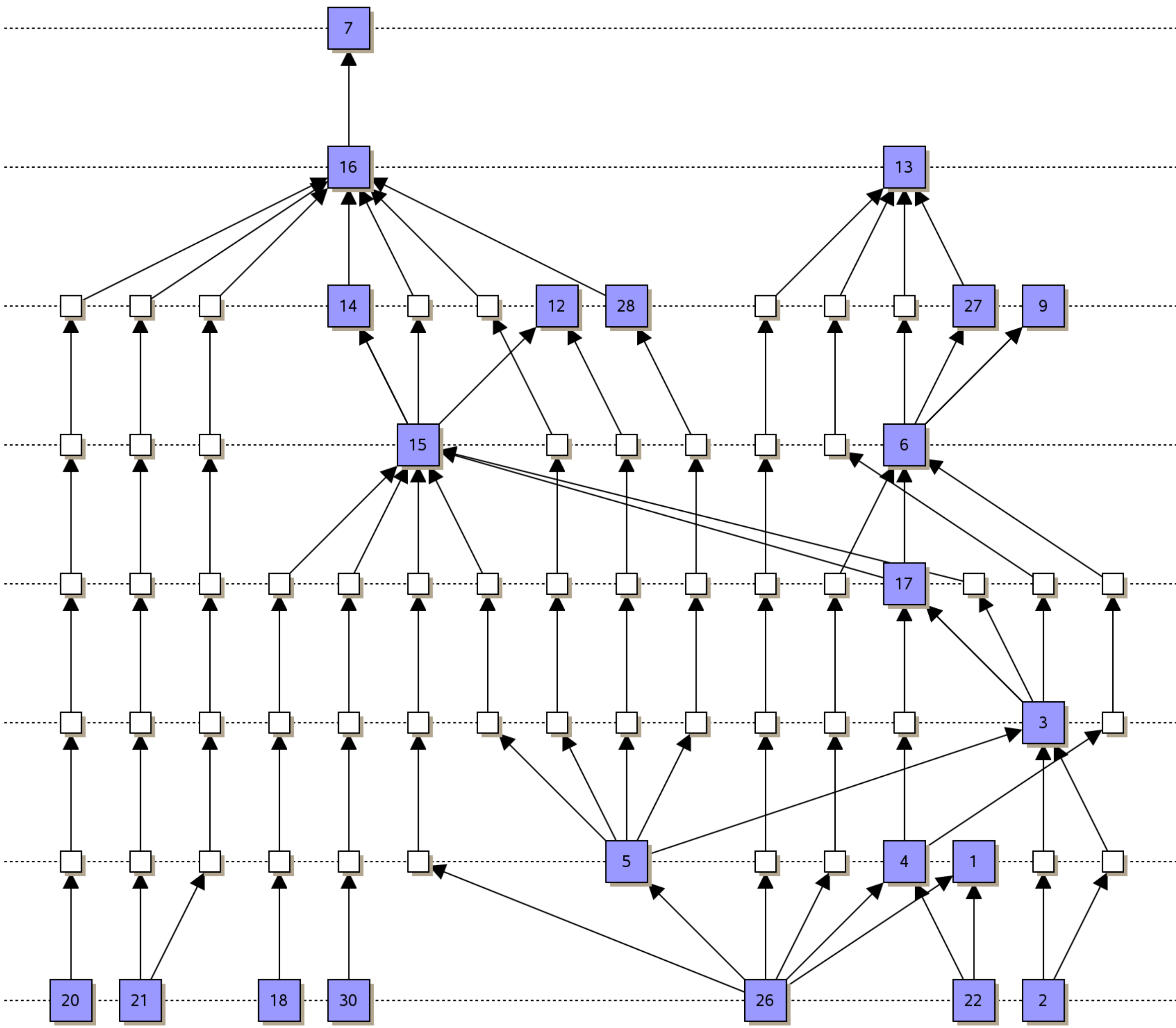
Properties:

- solving quadratic program often time expensive
- width can grow exponentially
- objective function can be modified for optimizing verticality

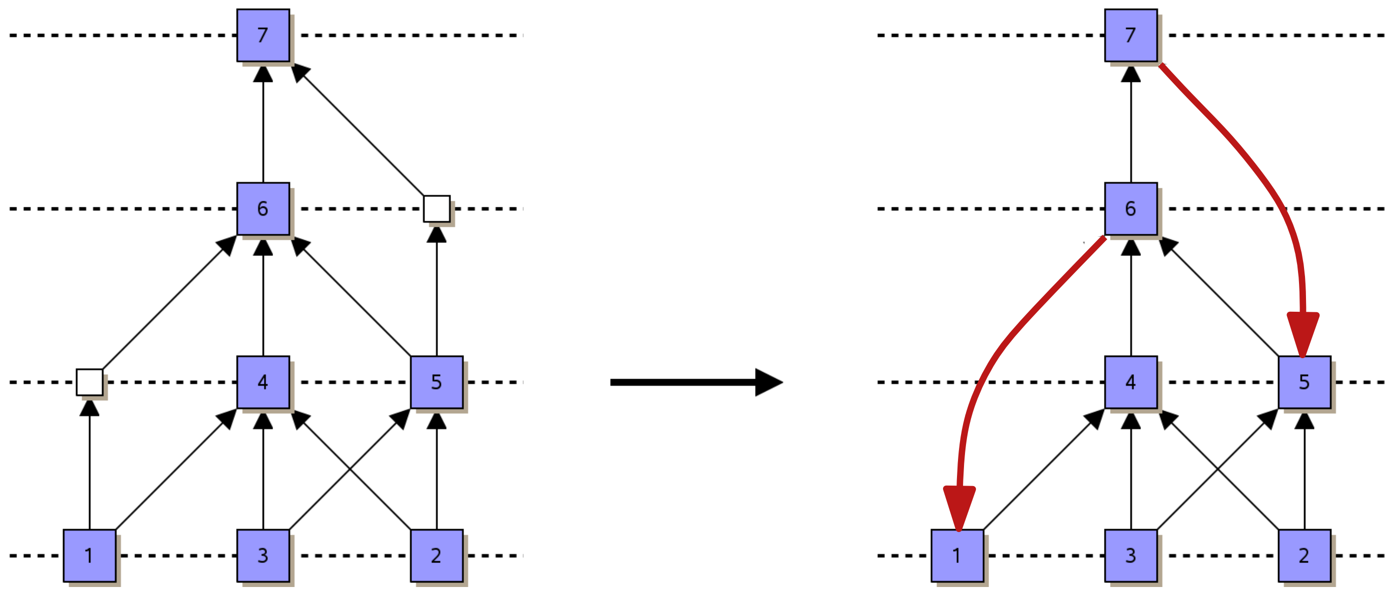
Example



Example

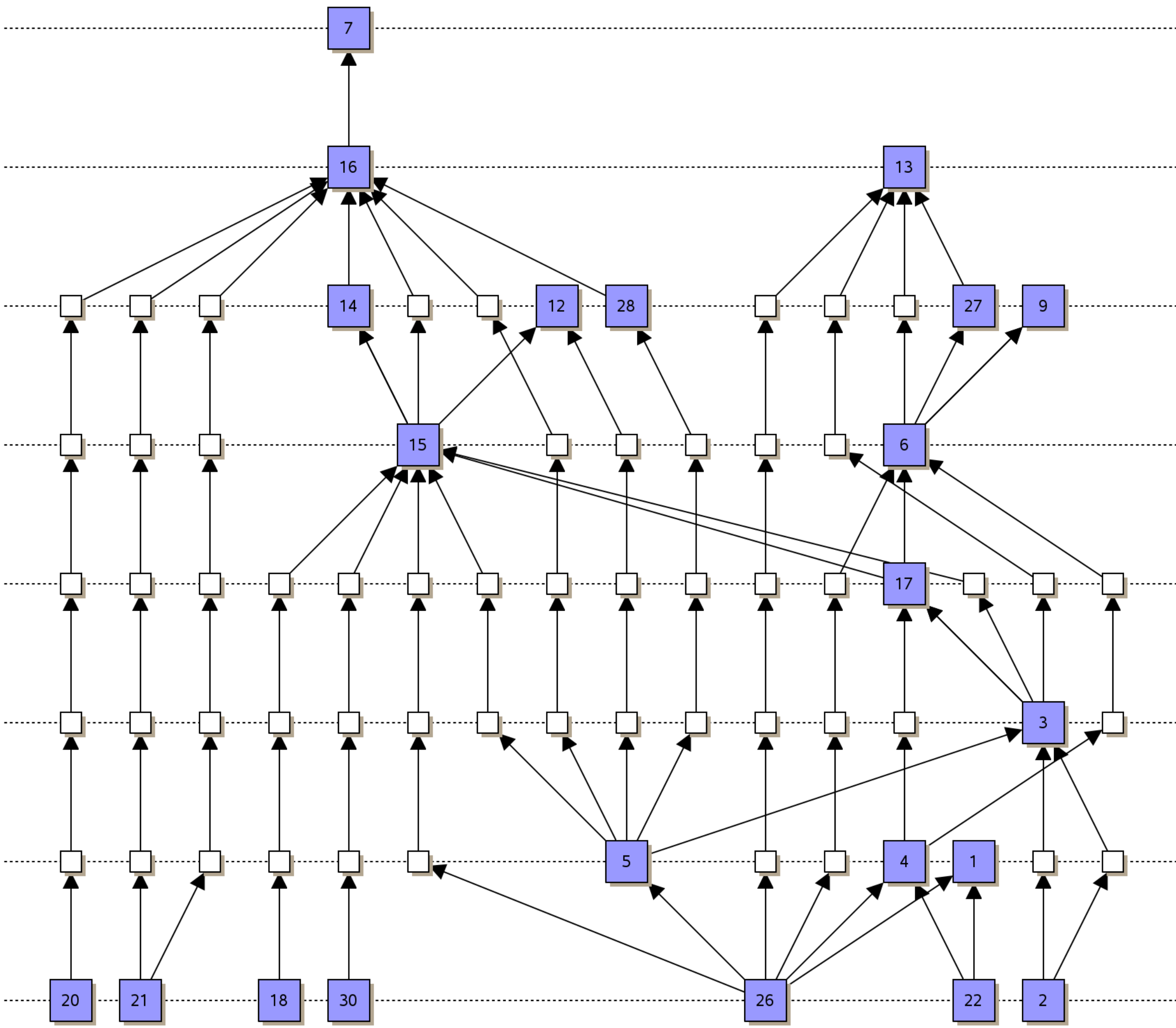


Step 5: Drawing edges

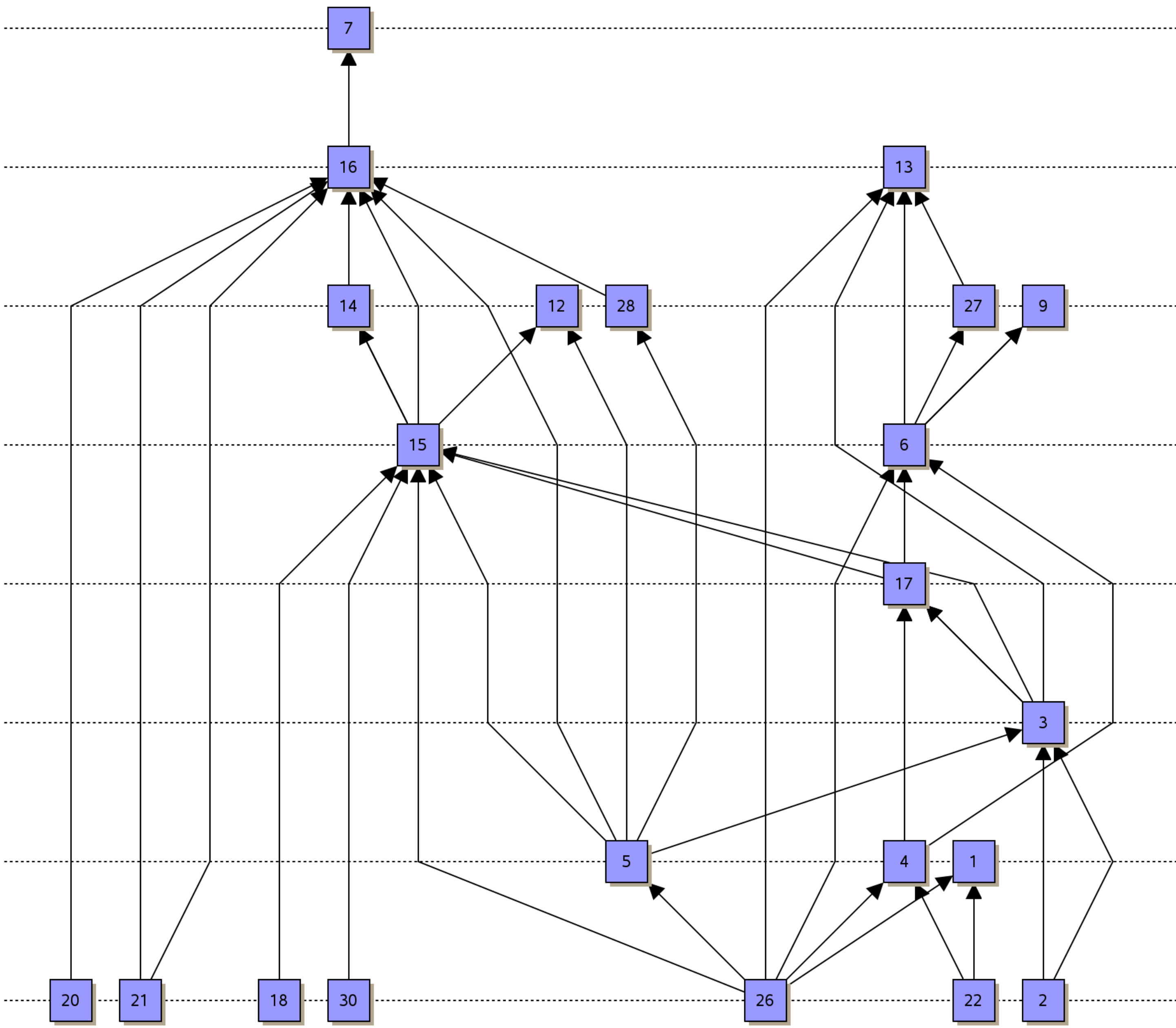


- postprocessing: optionally substitute polylines by Bézier curves
- draw all edges in original orientation

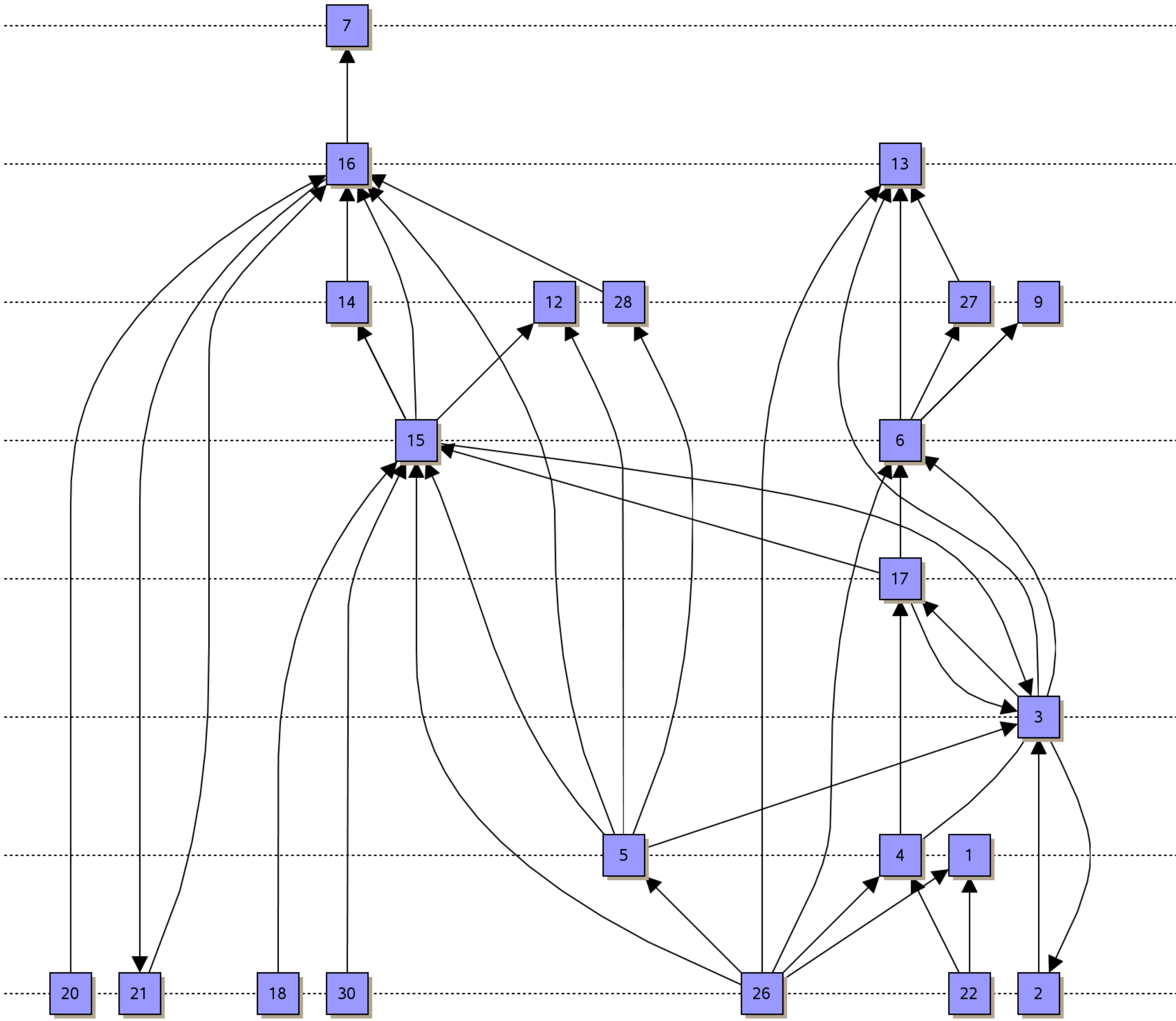
Example



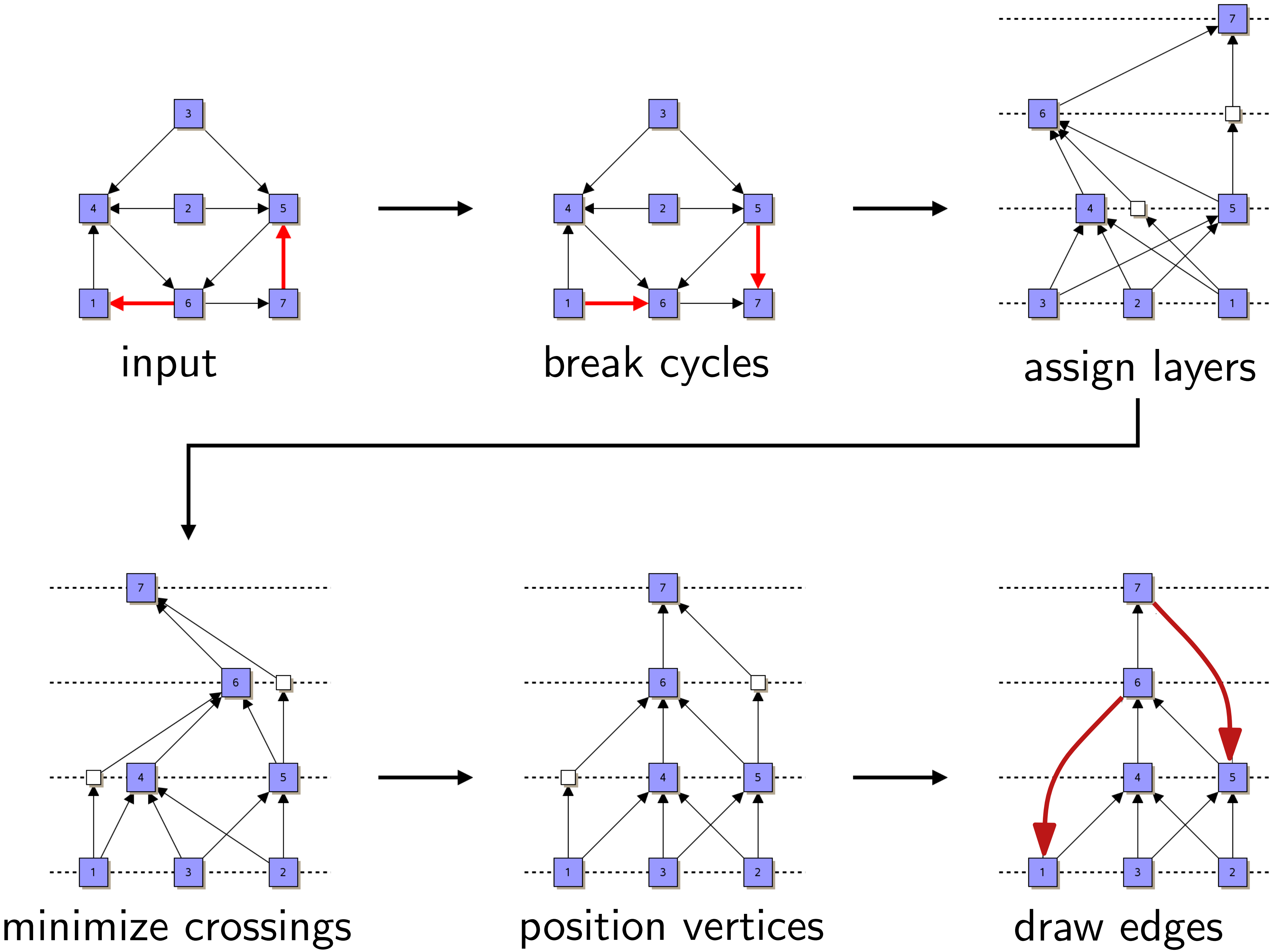
Example



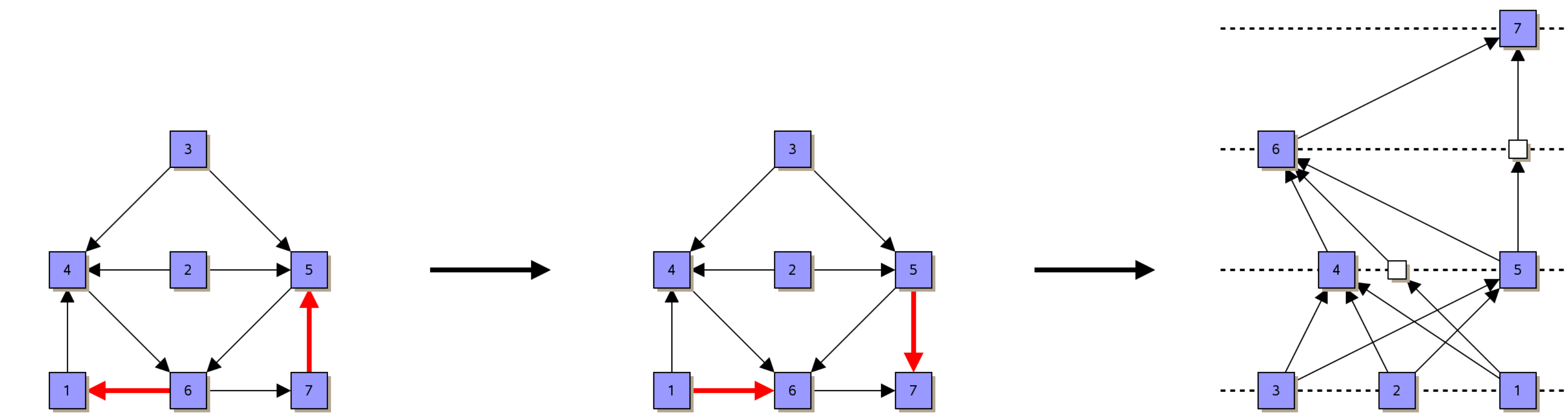
Example



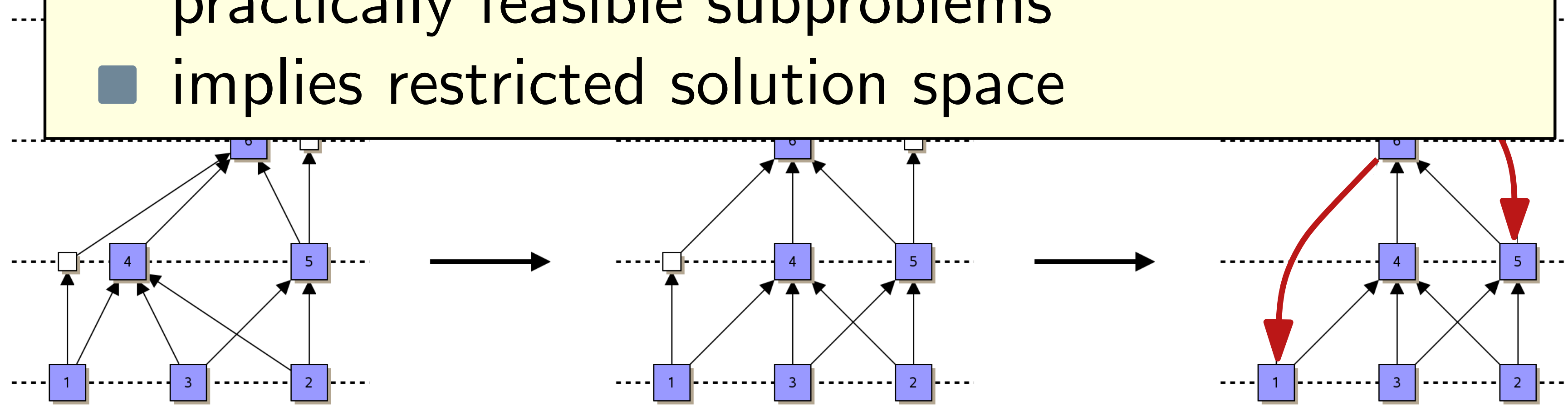
Summary


input
break cycles
assign layers
minimize crossings
position vertices
draw edges

Summary



- flexible framework for drawing directed graphs
- sequential optimization of various criteria
- decomposition into often NP-hard but still practically feasible subproblems
- implies restricted solution space



minimize crossings position vertices draw edges

Announcements

- The class on June 12 is shifted to Monday, June 11 in the same time slot 9:00–11:00 in seminar room 186.
- Student presentations from exercise groups will take place on July 3, 10:00–12:00 and 13:00–15:00. Attendance required.
- Oral exam dates are July 10 and September 18. Registration in TISS will open June 1.