Dual problems in linear programming

Yulia Burkatovskaya

Outline

- Dual LP-problems
- Maximum-weight matching
- Assignment problem
- Minimum-cost flow

Dual LP-problems

Initial problem

Objective function

- $L(X) = \sum_{i=1}^{n} c_i x_i \rightarrow max$

Subject to

- $\sum_{i=1}^{n} a_{j,i} x_i \leq b_j, j = \overline{1,k};$
- $\sum_{i=1}^{n} a_{j,i} x_i = b_j, j = \overline{k+1,m};$
- $x_i \geq 0, i = \overline{1,l}.$

Dual problem

Objective function

- $l(Y) = \sum_{j=1}^{m} b_j y_j \rightarrow min$

Subject to

- $\sum_{j=1}^{m} a_{j,i} y_j \geq c_i, i = \overline{1,l};$
- $\sum_{j=1}^{m} a_{j,i} y_j = c_i, i = \overline{l+1,n};$
- $y_j \geq 0, j = \overline{1,k}.$

Dual LP-problems

- **Complementary slackness conditions**

$$L(X) = \sum_{i=1}^{n} c_i x_i = \sum_{i=1}^{l} c_i x_i + \sum_{i=l+1}^{n} c_i x_i \le$$

$$\le \sum_{i=1}^{l} x_i \sum_{j=1}^{m} a_{i,j} y_j + \sum_{i=l+1}^{n} x_i \sum_{j=1}^{m} a_{i,j} y_j =$$

$$= \sum_{j=1}^{k} y_j \sum_{i=1}^{n} a_{j,i} x_i + \sum_{j=k+1}^{m} y_j \sum_{i=1}^{n} a_{j,i} x_i \le$$

$$\le \sum_{j=1}^{k} b_j y_j + \sum_{j=k+1}^{m} b_j y_j = \sum_{j=1}^{m} b_j y_j = l(Y)$$

So, $L(X) \le l(Y)$ and

$L(X) = l(Y)$ iff (**all summands are non-negative!**):

(a) $\sum_{i=1}^{l} x_i \left(\sum_{j=1}^{m} a_{i,j} y_j - c_i \right) = 0$;

(b) $\sum_{j=1}^{k} y_j \left(b_j - \sum_{i=1}^{n} a_{j,i} x_i \right) = 0$.

Dual LP-problems

- **Complementary slackness conditions**

So, $L(X) = l(Y)$ iff :

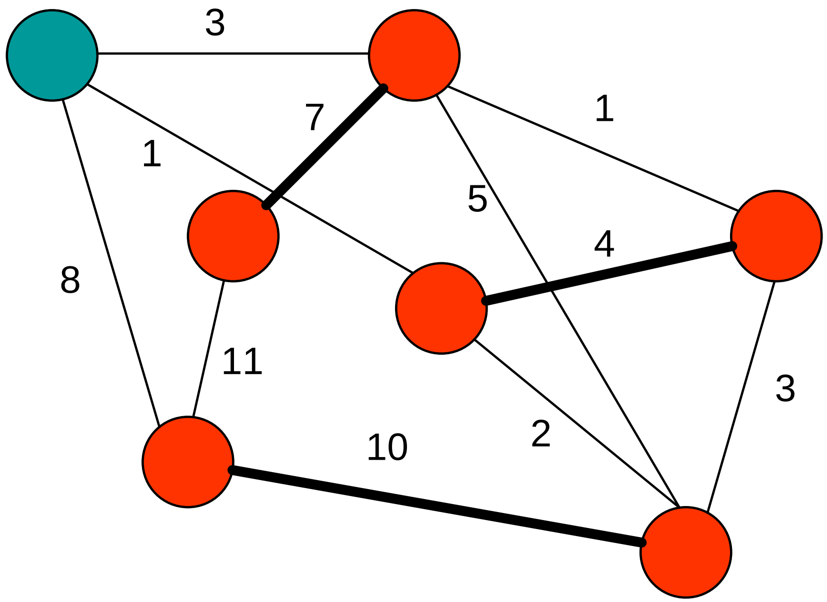
(a) For all $i = \overline{1, l}$, either $x_i = 0$ or the corresponding condition in the dual problem turns into the equality $\sum_{j=1}^{m} a_{j,i} y_j = c_i$;

(b) For all $j = \overline{1, k}$, either $y_j = 0$ or the corresponding condition in the initial problem turns into the equality $\sum_{i=1}^{n} a_{j,i} x_i = b_j$.

To solve the LP problem, we can find values (X, Y), which satisfy the complementary slackness conditions. These values provide the maximum value of the objective function $L(X)$ and the minimum value of the objective function $l(Y)$.

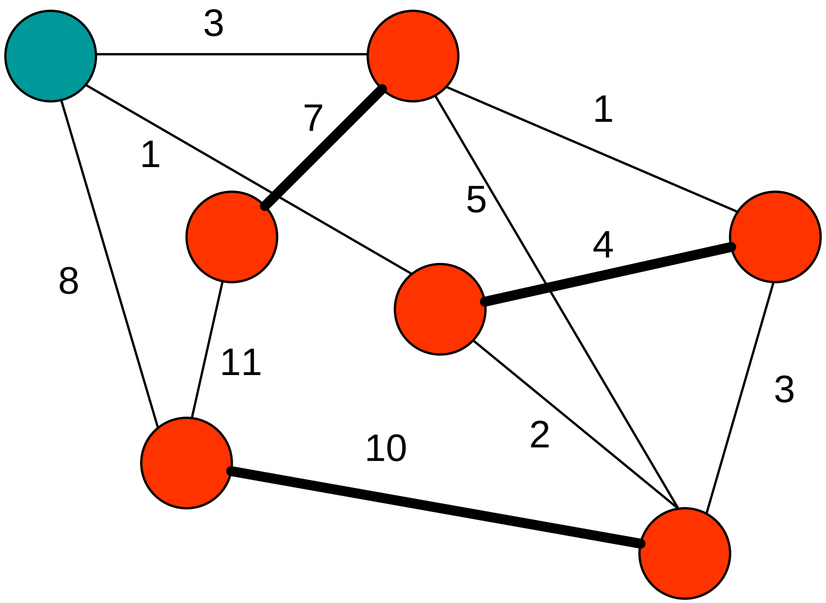
Maximum-weight matching

- **Maximum-weight matching** in a weighted graph G(V,E) is a matching with the maximum sum of the edge weights.



Maximum-weight matching

- **Vertices – people.**
- Two persons, working in pair, produce a certain amount of a product. How to match people, to produce the greatest amount of product?



Maximum-weight matching

- **Problem statement**

$$\sum_{j=1}^{q} c_j \xi_j \to \max;$$

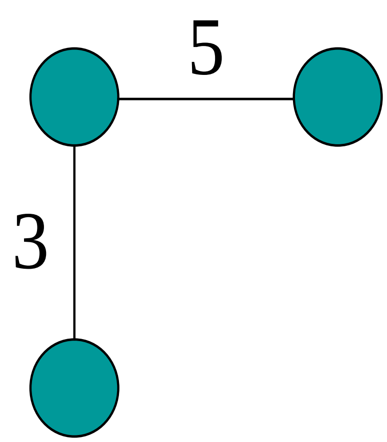
$$\sum_{j=1}^{q} I_{kj} \xi_j \leq 1, \quad \forall k = 1, \ldots, p;$$

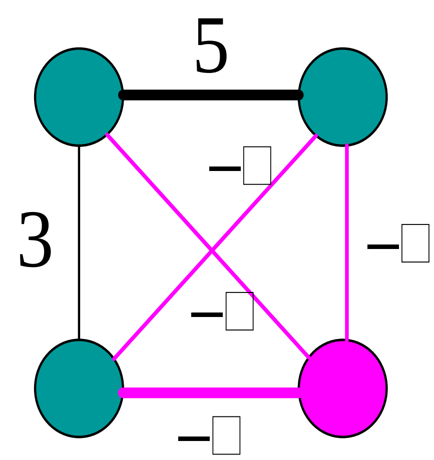
$$\xi_j \in \{0, 1\}.$$

Maximum-weight matching

The problem can be transformed into the perfect maximum-weight matching problem:

- add artificial edges with the weight $-\infty$;
- add an artificial vertex if the number of vertices is odd.





Maximum-weight matching

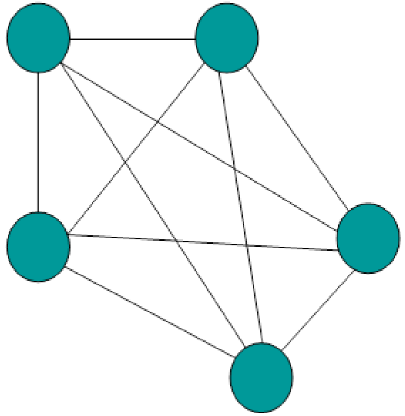
- **Problem statement**
- Solutions: $\xi_j \in \{0,1,1/2\}$

$$\sum_{j=1}^{q} c_j \xi_j \to \max;$$

$$\sum_{j=1}^{q} I_{kj} \xi_j = 1, \quad \forall k = 1, \ldots, p;$$

$$\xi_j \in \{0,1\}.$$

$G(V_r,E_r)$: $V_r=2p_r+1$



Maximum-weight matching

- **Problem statement (non-discrete!)**

$$\sum_{j=1}^{q} c_j \xi_j \longrightarrow max;$$

$$(a) \sum_{j \in T_k} \xi_j = 1, \quad k = \overline{1, p}, \; T_k = \{j = (i, k)\};$$

$$(b) \sum_{j \in E_r} \xi_j \leq p_r, \; V_r \subseteq V, \; |V_r| = 2p_r + 1;$$

$$\xi_j \geq 0, \; j = \overline{1, q}.$$

- (a) every vertex is covered by an edge;
- (b) in every odd-cardinality set of vertices, every vertex is covered by at most one edge.

Maximum-weight matching

Dual problem statement

- π_i – variables associated with vertices
- λ_r – variables associated with odd-cardinality sets of vertices

$$\sum_{j=1}^{p} \pi_j + \sum_{V_r} p_r \lambda_r \to \min;$$

$$\pi_i + \pi_k + \sum_{V_r \in F_j} \lambda_r \geq c_j, \quad \forall\, j = 1, \ldots, q, \quad e_j = (v_i, v_k), \quad F_j = \{V_r | e_j \in E_r\};$$

$$\lambda_j \geq 0, \quad \forall\, V_r.$$

Maximum-weight matching

Complementary slackness conditions

- For every ξ_j, $j = (i, k)$ either $\xi_j = 0$, or the equality holds in the corresponding constraint $\pi_i + \pi_k + \sum_{V_r \in F_j} \lambda_r = c_j$ (the edge $j = (i, k)$ can belong to a matching only if the equality holds);
- For every λ_r, either $\lambda_r = 0$, or the equality holds in the corresponding constraint $\sum_{j \in E_r} \xi_j = p_r$ ($\lambda_r > 0$ only if the odd-cardinality set of vertices has a maximum-cardinality matching).

Maximum-weight matching

- **Start. Choose initial values of the variables.**
- For every λ_r, $\lambda_r = 0$;
- For every edge $j = (i, k)$, $\pi_i + \pi_k \geq c_j$.
- Equality partial graph $G' = (V, E')$: $\mathrm{E}' = \{j = (i, k)\ , \pi_i + \pi_k = c_j\}$.
- **An optimal matching contains only edges from the EPG.**

Maximum-weight matching

- Let the current graph (after a number of blossom shrinkings), be G_{f-1} and its equality partial graph be G'_f. In G'_f, start growing an alternating tree T until the tree either:
- A. blossoms;
- B. augments;
- C. becomes Hungarian.

Maximum-weight matching

- **Case A.** In this case the blossom is shrunk thus obtaining a new graph G_f and its equality partial graph G'_{f+1}.
- The shrinking of a blossom leaves the remaining T with the correct structure of an alternating tree.
- T could be retained and growing of the tree can continue.

Maximum-weight matching

- **Case B.** In this case a better matching with larger cardinality is obtained.
- T must now be discarded and a new tree grown in the same ${G'}_f$ by choosing some remaining unsaturated vertex of ${G'}_f$ as the new root.
- Once T is discarded and a new tree is started to be grown in ${G'}_f$ some of the pseudo-vertices of ${G'}_f$ (formed by the shrinking of earlier blossoms) can now appear labeled inner in the new tree.

Maximum-weight matching

- **Case C**. In this case, the weight vector $[\pi_i, \lambda_r]$ is changed for the current graph G_{f+1} so that a new EPG is obtained. The changes in $[\pi_i, \lambda_r]$ are chosen so that the new G'_f will:
- (a) Continue to satisfy the complementary slackness conditions; and
- (b) Either:
  - the current alternating tree in the old G'_f can be grown further, or it blossoms, or augments—using new links that enter the new G'_f ;
  - Or: some pseudo-vertex of the current alternating tree which was labelled inner is disposed of;
  - Or: we prove that no perfect matching exists in G_0.

Maximum-weight matching

- For the links $j = (i, k)$ in G_{f+1} but not in G'_f, with one terminal vertex in the current alternating tree T and labeled **outer**, and the other terminal vertex **not in $\boldsymbol{T}$** calculate:

$$\Delta_1 = \min_j \{\pi_i + \pi_k - c_j\}$$

- Then,

$$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3\}$$

Maximum-weight matching

- For the links $j = (i, k)$ in G_{f+1} but not in G'_f, with both terminal vertices in T and both labelled **outer** calculate:

$$\Delta_2 = \frac{1}{2} \min_j \{\pi_i + \pi_k - c_j\}$$

Maximum-weight matching

- For the sets S_r of vertices of G forming an outermost pseudo-vertex of T labelled inner calculate:

$$\Delta_3 = \frac{1}{2}\min_{S_r}\{\lambda_r\}$$

Maximum-weight matching

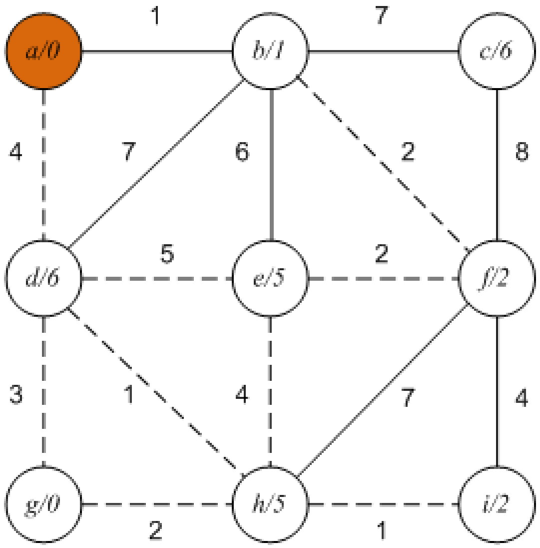
- For every vertex k of G which is an **outer** vertex of T or is contained in a **pseudo-vertex of T labelled outer**, decrease π_k to $\pi_k - \Delta$.

- For every vertex k of G which is an **inner** vertex of T or is contained in a **pseudo-vertex of T labelled inner**, increase π_k to $\pi_k + \Delta$.

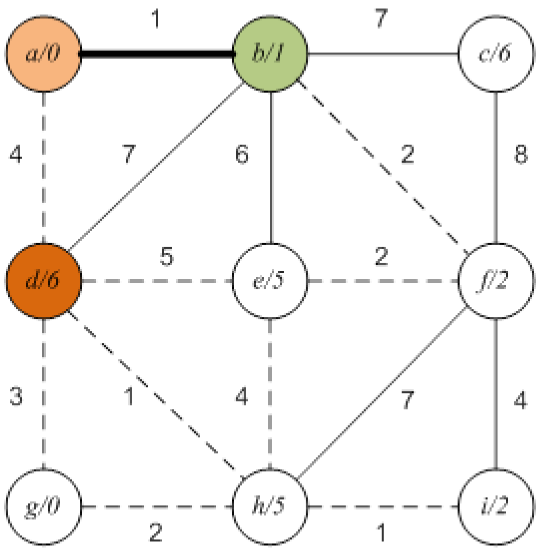
- For every set S_r of vertices of G which forms an **outermost pseudo-vertex of T labelled outer**, increase λ_r to $\lambda_r + 2\Delta$.

- For every set S_r of vertices of G which forms an **outermost pseudo-vertex of T labelled inner** decrease λ_r to $\lambda_r - 2\Delta$.

Maximum-weight matching



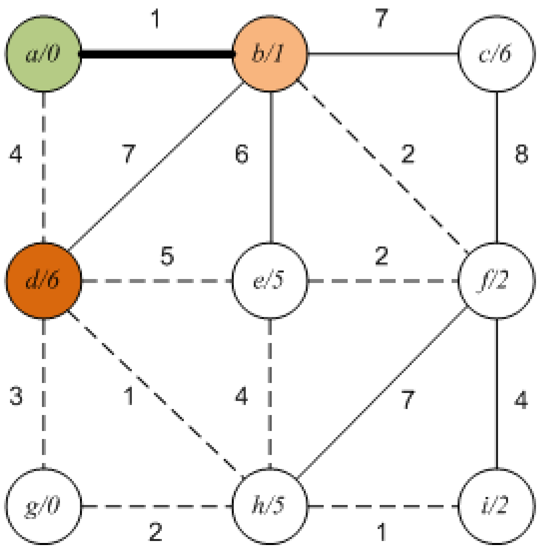
a – root
ab – augmenting path



d - root

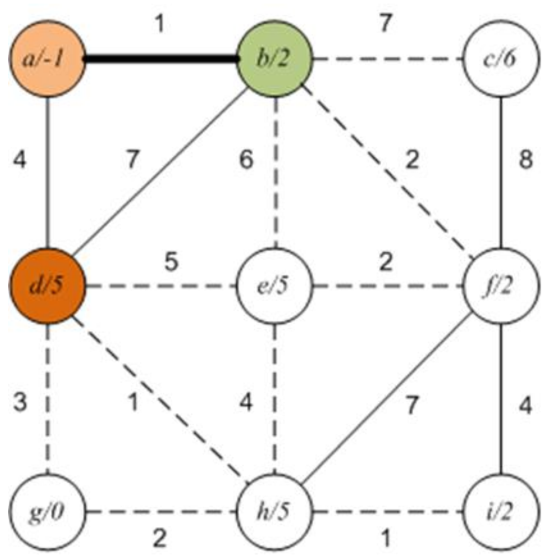
Hungarian tree
D1:
(d,e): 6+5-5=6
(d,h): 6+5-1=10
(d,g): 6+0-3=2 min
D2:
(a,d): (0+6-4)/2=1 min
D3: no
D=1=D2

Maximum-weight matching



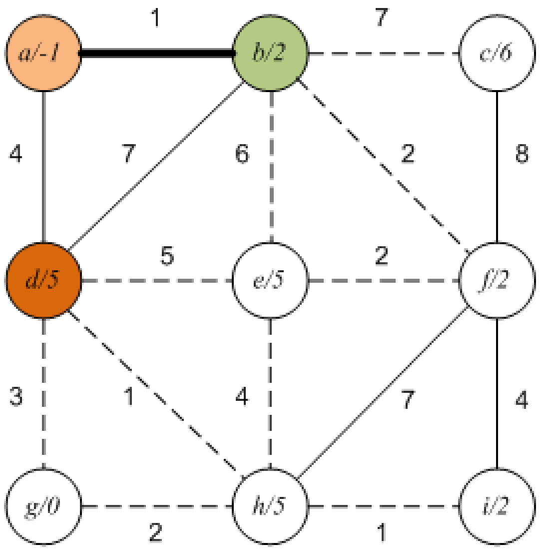
d - root

Hungarian tree
D1:
(d,e): 6+5-5=6
(d,h): 6+5-1=10
(d,g): 6+0-3=2 min
D2:
(a,d): (0+6-4)/2=1 min
D3: no
D=1=D2

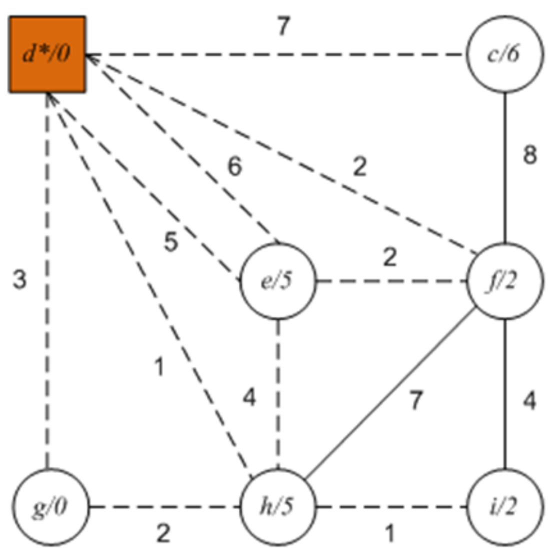


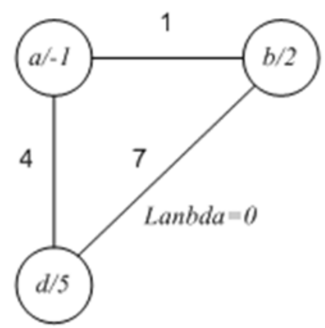
d - root
dba - blossom
Lambda(dba) = 0

Maximum-weight matching



d - root
dba - blossom
Lambda(dba) = 0



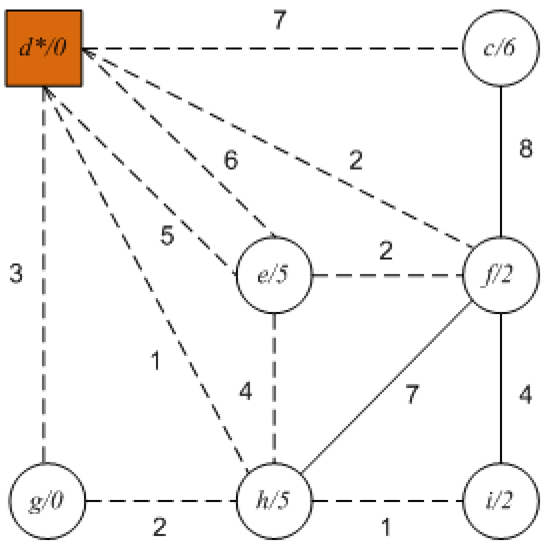


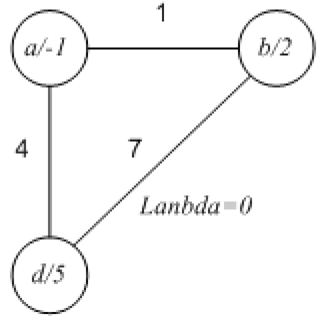
d* - root

Hungarian tree
D1:
- (d,g): 5+0-3=2
- (d,h): 5+5-1=9
- (d,e): 5+5-5=5
- (b,e): 2+5-6=1 min
- (b,f): 2+2-2=2
- (b,c): 2+6-7=1 min

D2: no
D3: no
D=1=D1

Maximum-weight matching





d* - root

Hungarian tree

D1:

(d,g): 5+0-3=2

(d,h): 5+5-1=9

(d,e): 5+5-5=5

(b,e): 2+5-6=1 min

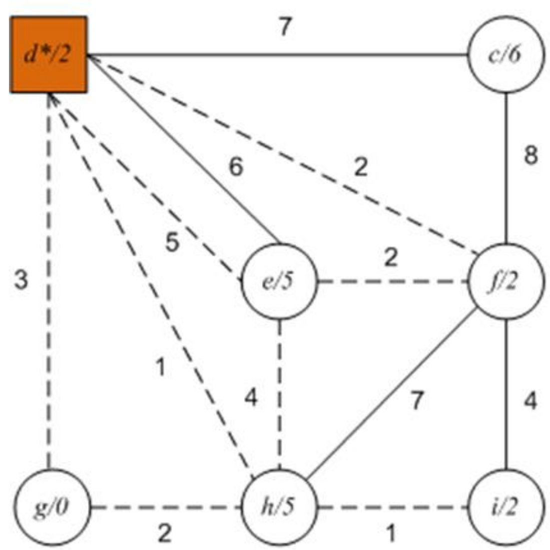
(b,f): 2+2-2=2

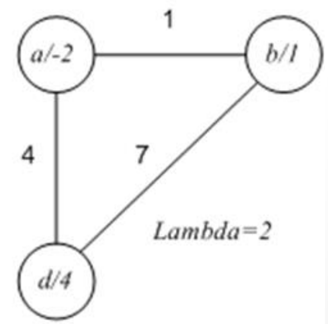
(b,c): 2+6-7=1 min

D2: no

D3: no

D=1=D1

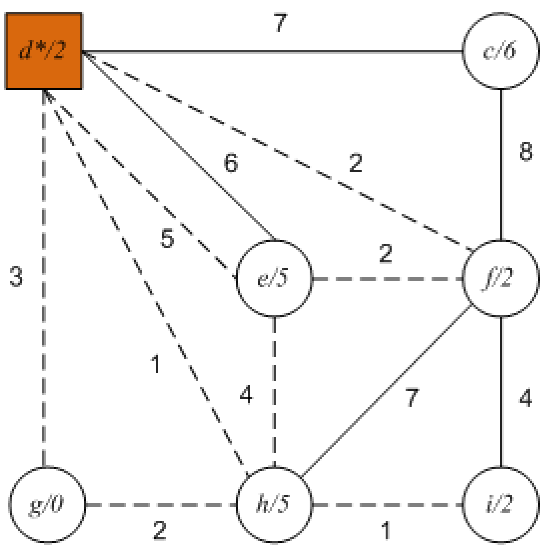


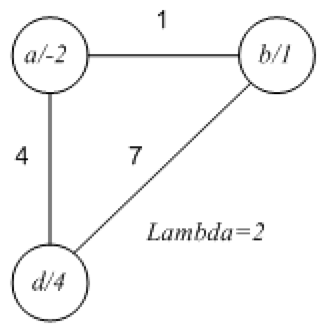


d* - root

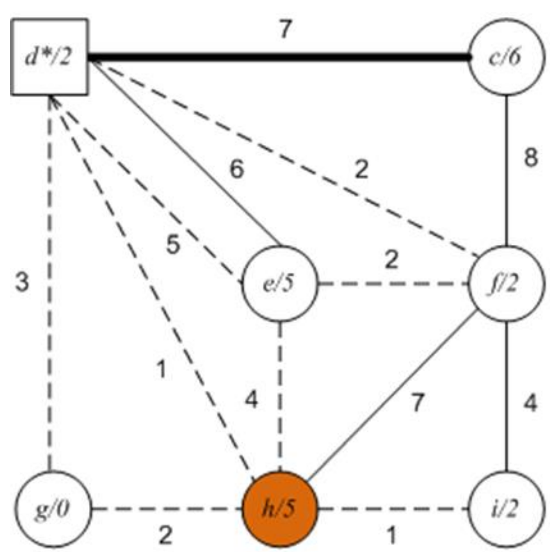
d*c – augmenting path

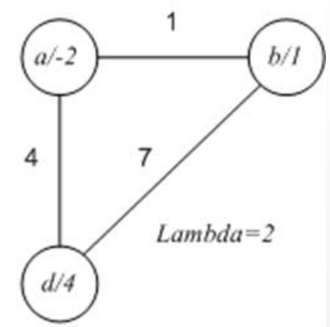
Maximum-weight matching





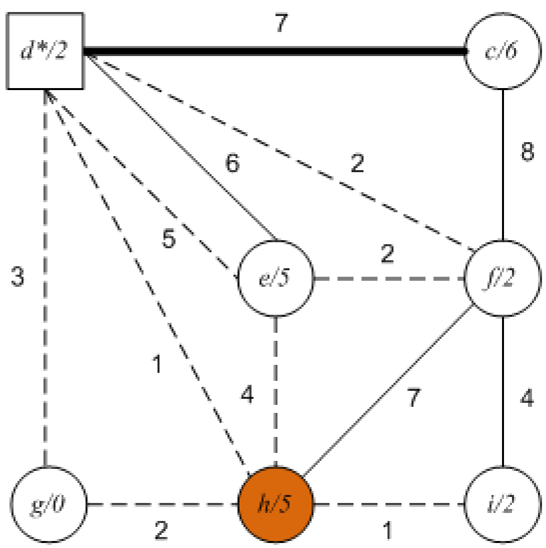
d* - root
d*c – augmenting path

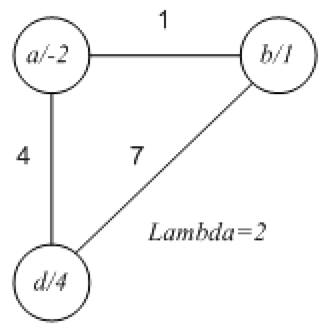




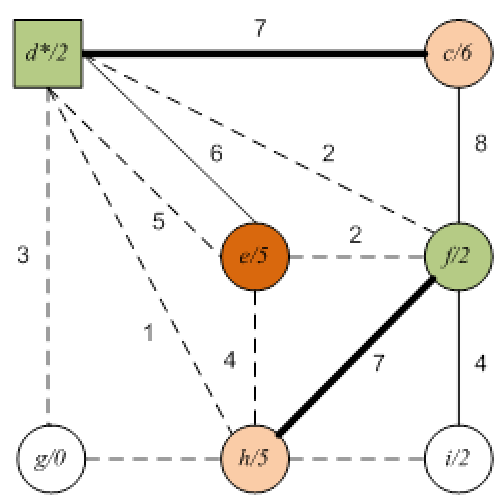
h - root
hf – augmenting path

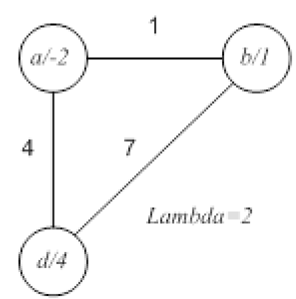
Maximum-weight matching





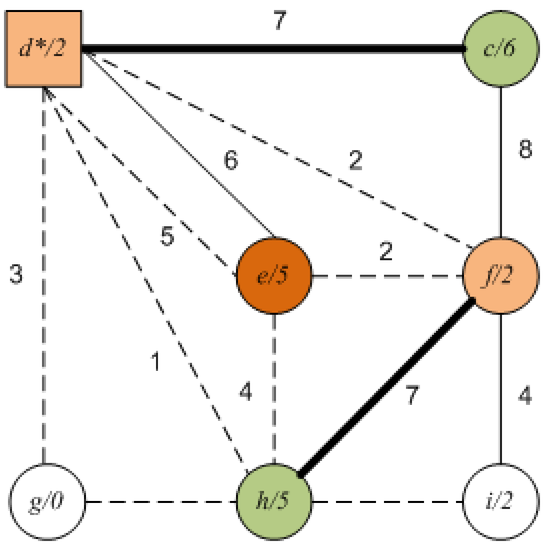
h - root
hf – augmenting path

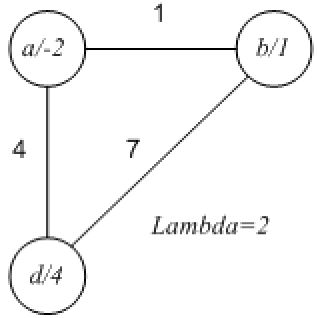




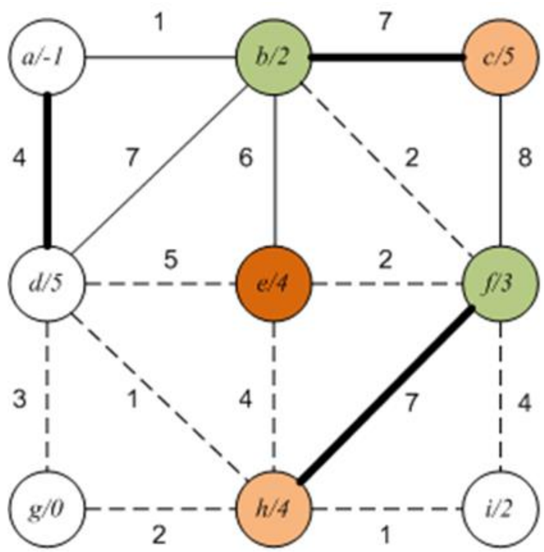
e - root
Hungarian tree
D1:
(h,i): 5+2-1=6
(h,g): 5+0-2=3 min
D2:
(h,e): (5+5-4)/2=3 min
D3:
d*: 2/2=1 min
D=1=D3

Maximum-weight matching



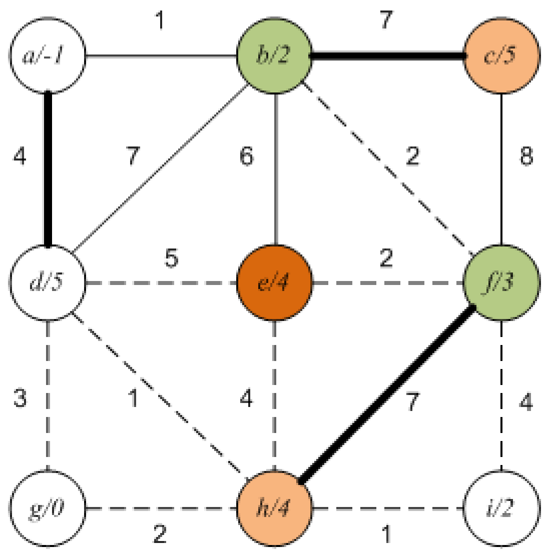


e - root
Hungarian tree
D1:
  (h,i): 5+2-1=6
  (h,g): 5+0-2=3 min
D2:
  (h,e): (5+5-4)/2=3 min
D3:
  d*: 2/2=1 min
D=1=D3

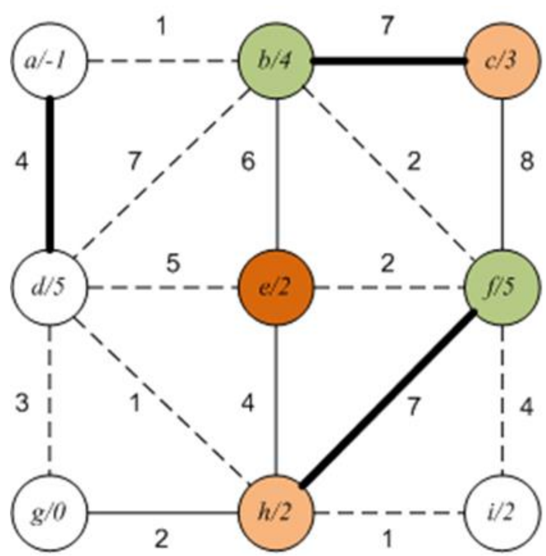


e - root
Hungarian tree
D1:
  (h,i): 4+2-1=5
  (h,g): 4+0-2=2 min
  (e,d): 4+5-5=4
D2:
  (h,e): (4+4-4)/2=2 min
D3: no
D=1=D1=D2

Maximum-weight matching

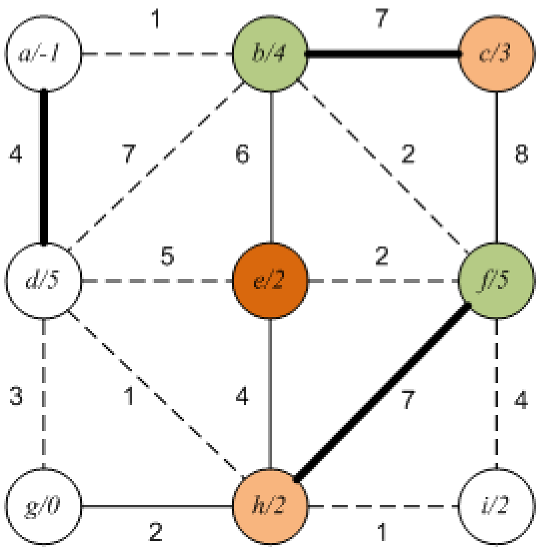


e - root
Hungarian tree
D1:
(h,i): 4+2-1=5
(h,g): 4+0-2=2 min
(e,d): 4+5-5=4
D2:
(h,e): (4+4-4)/2=2 min
D3: no
D=1=D1=D2

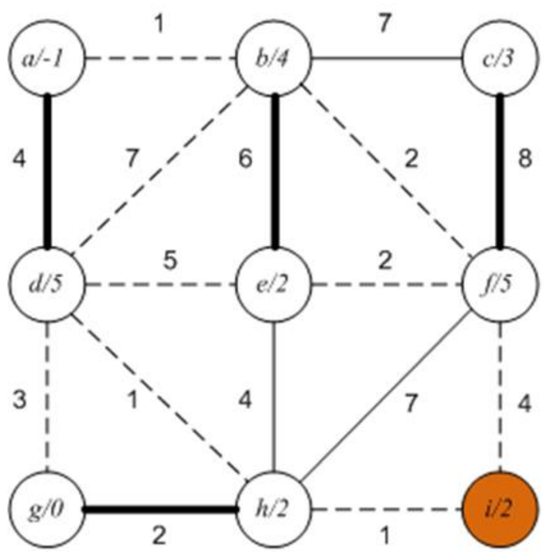


e - root
ebcfhg – augmenting path

Maximum-weight matching

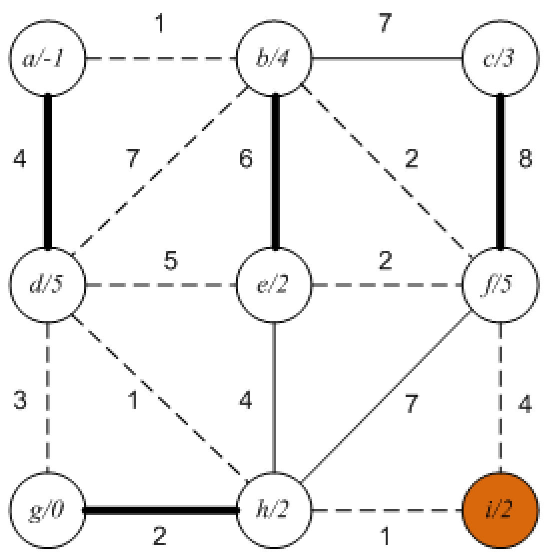


e - root
ebcfhg – augmenting path

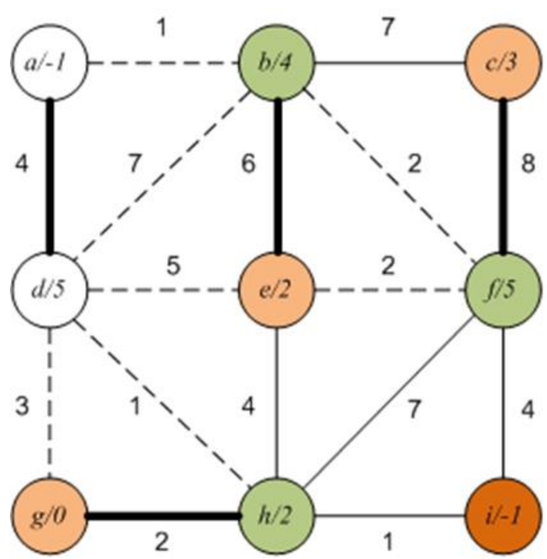


i - root
Hungarian tree
D1:
(i,f): 2+5-4=3 min
(i,h): 2+2-1=3 min
D2: no
D3: no
D=3=D1

Maximum-weight matching

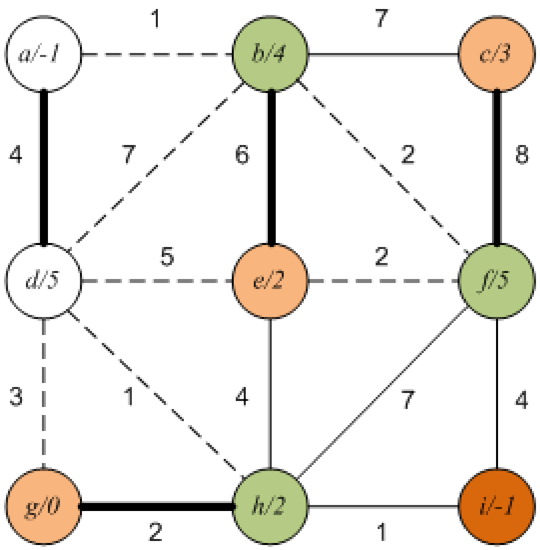


i - root
Hungarian tree
D1:
  (i,f): 2+5-4=3 min
  (i,h): 2+2-1=3 min
D2: no
D3: no
D=3=D1

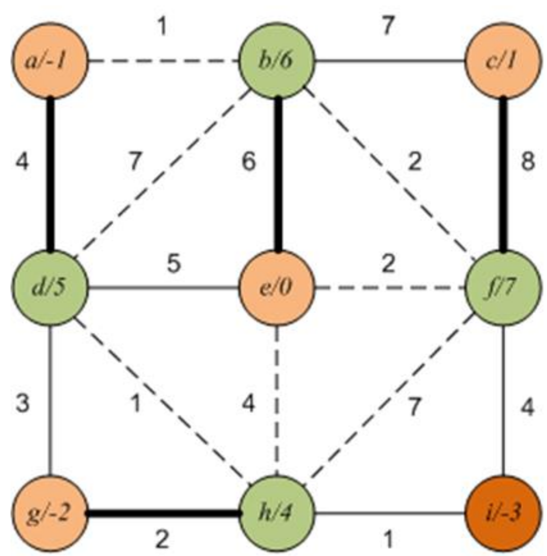


i - root
Hungarian tree
D1:
  (e,d): 2+5-5=2 min
  (g,d): 0+5-3=2 min
D2: no
D3: no
D=1=D1

Maximum-weight matching



i - root
Hungarian tree
D1:
(e,d): 2+5-5=2 min
(g,d): 0+5-3=2 min
D2: no
D3: no
D=1=D1



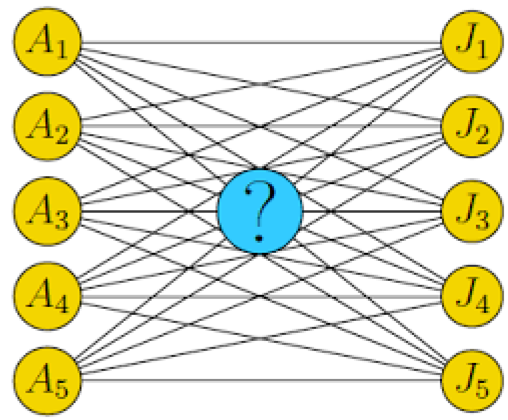
i - root
Hungarian tree
D1: no
D2: no
D3: no

Assignment problem

- To find, in a **weighted bipartite graph**, a matching in which the sum of weights of the edges is as small (large) as possible.

Machines

$$
\begin{array}{cc|ccccc}
 & & \mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{D} & \mathbf{E} \\
\hline
 & \mathbf{1} & 5 & 7 & 11 & 6 & 7 \\
 & \mathbf{2} & 8 & 5 & 5 & 6 & 5 \\
\mathbf{Job} & \mathbf{3} & 6 & 7 & 10 & 7 & 3 \\
 & \mathbf{4} & 10 & 4 & 8 & 2 & 4
\end{array}
$$



Assignment problem

Bipartite graph:

- No odd-length cycles;
- **No blossoms!**
- Hungarian algorithm.

Assignment problem

Initial problem

Objective function:

$$\sum_{j=1}^{q} c_j \xi_j \longrightarrow min;$$

Subject to:

$$\sum_{j \in T_k} \xi_j = 1,$$

$$k = \overline{1,p},\ T_k = \{j = (i,k)\};$$

$$\xi_j \geq 0,\ j = \overline{1,q}.$$

Dual problem

Objective function:

$$\sum_{i=1}^{p} \pi_j \longrightarrow max;$$

Subject to:

$$\pi_i + \pi_k \leq c_j,$$

$$\forall j = (i,k),\ j = \overline{1,q}.$$

Minimum-cost flow

- Residual network