Models and Algorithms for Matching and Assignment Problems

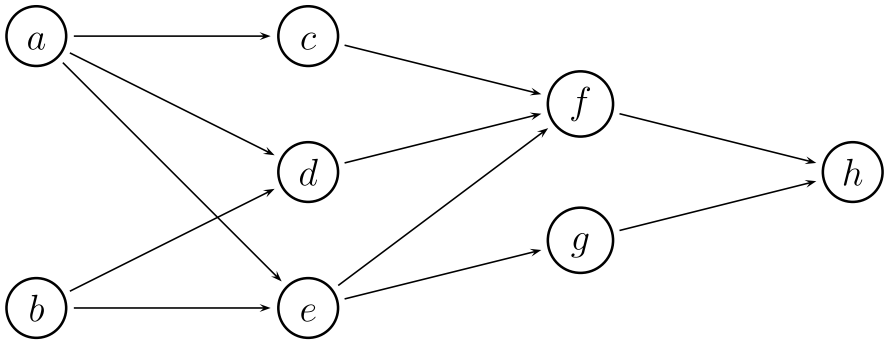
Silvano Martello
DEI "Guglielmo Marconi", Università di Bologna, Italy

3. Maximum matching applications

Applications of maximum matching: 1. Vehicle scheduling problem

- In an **airline network** (or railway network, or bus network) a set of trips must be served.
- What is the **minimum number of vehicles** (aircrafts in this case) needed to serve all trips?
- **Graph theory model:**
  - Network (oriented graph) $\mathcal{N} = (N, A)$ with *vertex set* N and *arc set* A;
  - usually, one has a daily or weekly schedule;
  - every trip (e.g., Monday morning trip from X to Y) is modeled as a vertex in $\mathcal{N}$;
  - two vertices i and j are joined by an arc (i, j) if it is possible to serve trip j (e.g., from Y to Z) immediately after trip i (e.g, from X to Y) by the same vehicle:

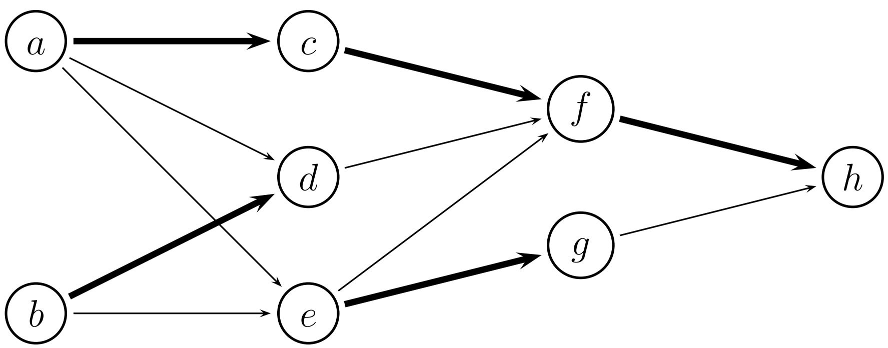
  - The trips which are served by one vehicle form therefore a directed path in the graph.
  - **Solution:** find a minimum number of vertex-disjoint paths in $\mathcal{N}$ which cover all vertices.



Vertices a, b, c, ... $=$ trips;

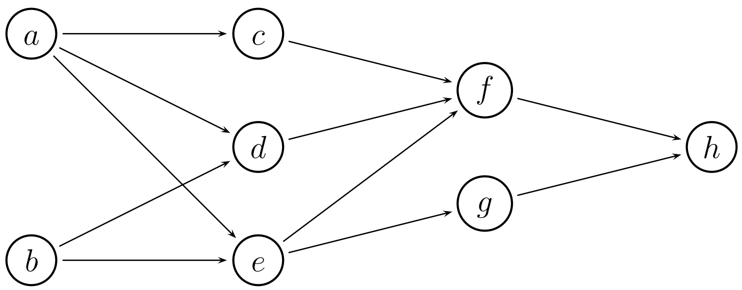
Arc $(i, j) \Leftrightarrow$ it is possible to serve trip j immediately after trip i

by the same vehicle.



Minimum number of node disjoint paths (bold) which cover all nodes.

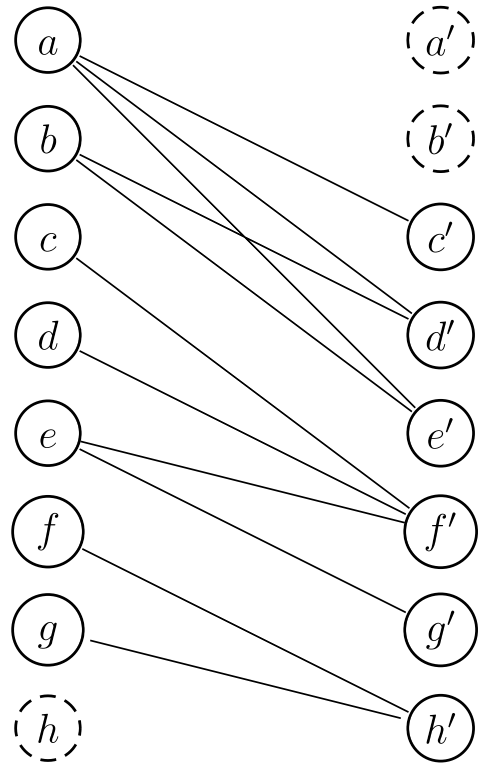
How to determine it?

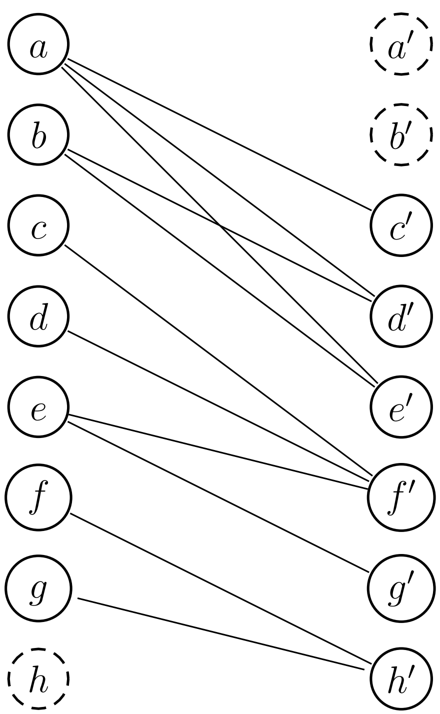


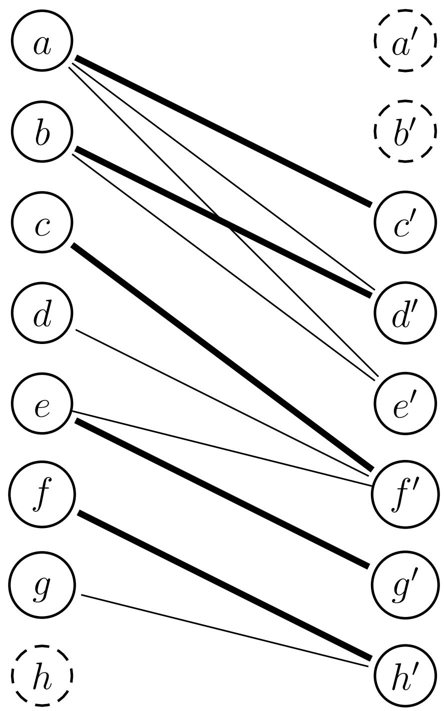
1) Define a bipartite graph $G = (U, V; E)$ as:

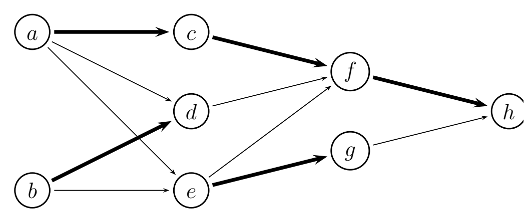
- $\forall \;\; i \in N$ in which some arc starts, $\exists$ vertex $i \in U$;
- $\forall \;\; j \in N$ in which some arc ends, $\exists$ vertex $j \in V$;
- every arc $(i, j) \in A$ leads to an edge $[i, j'] \in E$.

2) A matching in G leads to arcs in $\mathcal{N}$ which form node disjoint paths, since the matched vertices correspond to nodes of $\mathcal{N}$ whose indegree and outdegree is at most 1.









3) Every system of node disjoint paths in $\mathcal{N}$

corresponds to a matching in G and vice versa.

4) $\forall$ path from i to j in $\mathcal{N}$, $\exists$ two vertices of G unmatched (i' and j) $\Rightarrow$

5) Maximum cardinality matching in $G \Leftrightarrow$

Minimum number of node disjoint paths in $\mathcal{N}$.

Applications of maximum matching: 2. Time slot assignment problem

- **Telecommunication systems using satellites:** the data, buffered in ground stations, are remitted to the satellite where they are sent back to earth;
- onboard the satellite n transponders connect the sending stations with the receiving stations.
- In **Time Division Multiple Access** technique these $n \times n$ connections change in very short time intervals of variable length λ;
- the **switch mode** describes, for each $n \times n$ connection, which sending stations are momentarily connected with which receiving stations.
- We can model the switch mode by an $n \times n$ binary **permutation matrix** P with exactly one 1-entry in every row and column:

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

- $p_{ij} = 1 \iff$ the i-th earth station remits data to the j-th earth station:
- 1 transmits to 3, 2 transmits to 1, 3 transmits to 2.

- An $n \times n$ **traffic matrix** T contains the information about the times needed to transfer the required data:

$$T = \begin{pmatrix} 0 & 4 & 7 \\ 2 & 0 & 0 \\ 2 & 5 & 0 \end{pmatrix}.$$

- t_{ij} = total time needed to transfer the data from station i to station j.
- After applying switch mode P_k for λ_k time units, T is changed to $T - \lambda_k P_k$:

$$P_k = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad \lambda_k = 2 \Longrightarrow T = \begin{pmatrix} 0 & 4 & 5 \\ 0 & 0 & 0 \\ 2 & 3 & 0 \end{pmatrix}.$$

- **Problem:** Given a traffic matrix T, determine switch modes P_k and times λ_k such that all data are remitted in minimum time:

$$\begin{array}{lll} \min & \sum_k \lambda_k & \\ \text{s.t.} & \sum_k \lambda_k P_k \geq T & \text{elementwise} \\ & \lambda_k \geq 0 & \end{array}$$

- **Observation:**
  - Let $t^* = \max(\,(\max \text{ row sum}), (\max \text{ column sum})\,)$.
  - No two elements of the same row or column can be remitted at the same time, so
  - t^* is a lower bound for $\min \sum_k \lambda_k$.
  - We can add dummy elements to the matrix so that all row and column sums are equal to t^*:
  - if we can find a solution of value t^*, such solution will be optimal.
- **Algorithm Equalize_sums:**

for $i := 1$ **to** n **do** $R_i := \sum_{j=1}^{n} t_{ij}$;
for $j := 1$ **to** n **do** $C_j := \sum_{i=1}^{n} t_{ij}$;
$t^* := \max\ (\max_i R_i,\ \ \max_j C_j)$;
for $i := 1$ **to** n **do** $a_i := t^* - R_i$;
for $j := 1$ **to** n **do** $b_j := t^* - C_j$;
for $i := 1$ **to** n **do**
    for $j := 1$ **to** n **do**
        $s_{ij} := \min(a_i, b_j)$;
        $a_i := a_i - s_{ij}$;
        $b_j := b_j - s_{ij}$
    endfor
endfor
$T := T + S$

- **Example:** $T = \begin{pmatrix} 4 & 7 & 1 \\ 3 & 1 & 0 \\ 2 & 1 & 1 \end{pmatrix}$;
- maximum row and column sum: $t^* = 12$;
- $a = (0, 8, 8)$, $b = (3, 3, 10)$;
- $i = 1$, $S = \begin{pmatrix} 0 & 0 & 0 \\ - & - & - \\ - & - & - \end{pmatrix}$, $a = (0, 8, 8)$, $b = (3, 3, 10)$;
- $i = 2$, $S = \begin{pmatrix} 0 & 0 & 0 \\ 3 & 3 & 2 \\ - & - & - \end{pmatrix}$, $a = (0, 0, 8)$, $b = (0, 0, 8)$;
- $i = 3$, $S = \begin{pmatrix} 0 & 0 & 0 \\ 3 & 3 & 2 \\ 0 & 0 & 8 \end{pmatrix}$, $a = (0, 0, 0)$, $b = (0, 0, 0)$;
- $T := T + S = \begin{pmatrix} 4 & 7 & 1 \\ 6 & 4 & 2 \\ 2 & 1 & 9 \end{pmatrix}$.

- Once matrix T has equal row and column sums, it is decomposed:
- **Algorithm Decompose:**

  $k := 1$;

  while $T \neq 0$ **do**

  construct a bipartite graph G with $|U| = |V| = n$ and an edge $[i, j]$ iff $t_{ij} > 0$;

  find a perfect matching φ_k in G, corresponding to a switch mode P_k;

  $\lambda_k := \min\{t_{i\varphi_k(i)} \ : \ i = 1, 2, \ldots, n\}$;

  $T := T - \lambda_k P_k$;

  $k := k + 1$

  endwhile
- At every iteration we have a matrix T with constant row and column sums.
- Hence there exists a perfect matching, that can be found by a maximum matching algorithm.

- **Example (resumed):** $T = \begin{pmatrix} 4 & 7 & 1 \\ 6 & 4 & 2 \\ 2 & 1 & 9 \end{pmatrix}$;
- $k = 1$: $\varphi_1 = (1,2,3)$, $P_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, $\lambda_1 = 4$, $T = \begin{pmatrix} 0 & 7 & 1 \\ 6 & 0 & 2 \\ 2 & 1 & 5 \end{pmatrix}$;
- $k = 2$: $\varphi_2 = (2,3,1)$, $P_2 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$, $\lambda_2 = 2$, $T = \begin{pmatrix} 0 & 5 & 1 \\ 6 & 0 & 0 \\ 0 & 1 & 5 \end{pmatrix}$;
- $k = 3$: $\varphi_3 = (2,1,3)$, $P_3 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, $\lambda_3 = 5$, $T = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$;
- $k = 4$: $\varphi_4 = (3,1,2)$, $P_4 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$, $\lambda_4 = 1$, $T = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$.
- $T = \begin{pmatrix} 4 & 7 & 1 \\ 6 & 4 & 2 \\ 2 & 1 & 9 \end{pmatrix}$

$$= 4 \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + 2 \cdot \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} + 5 \cdot \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} + 1 \cdot \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

A problem in scheduling theory

- We are given m machines and n jobs (assume, by simplicity, that $m = n$);
- each job requires processing on every machine (in any order);
- each machine can process at most one job at a time;
- no job can be processed simultaneously on two machines;
- a non-negative integer **matrix** T gives the total amount of time, t_{ij}, **job** j **must be processed on machine** i **(**$i, j = 1, 2, \ldots, n$**)**;
- each processing can be interrupted at any time and resumed later.
- **Preemptive Open Shop Scheduling Problem:** find a feasible schedule such that the completion time of the latest job is as small as possible.
- **Question:** Are you able to solve this problem?
- **Answer: Yes**, this problem is equivalent to the **Time Division Multiple Access** problem.
- Algorithm **(Equalize + Decompose)** was invented by Inukai (*IEEE Trans. Comm.*, 1979).
- Algorithm **(Equalize + Decompose)** was invented by Gonzalez and S. Sahni (*J. ACM*, 1976).
- **Question:** Who invented algorithm **(Equalize + Decompose)**? The answer will come later.
- (Note: if no processing can be interrupted, the problem is strongly $\mathcal{NP}$-hard.)

Exercise 2

An airline company wants to plan 8 routes, denoted as $a, b \ldots, h$, using the minimum number of aircrafts. An arc (x, y) in the graph below indicates that route y can be covered by the same aircraft that covers route x.

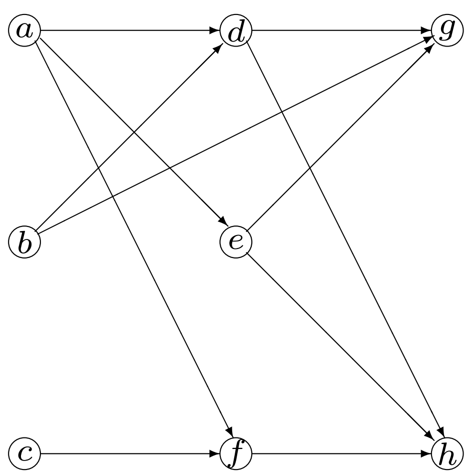
1. define the corresponding bipartite graph using the vertices provided on the right;

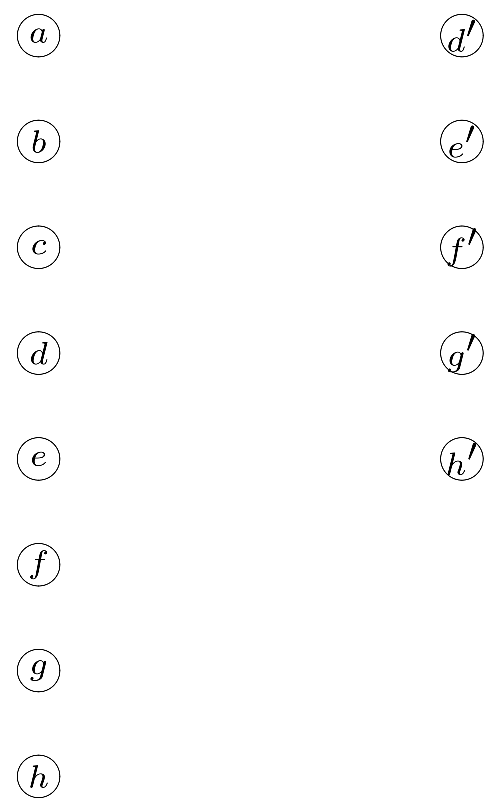
2. find a solution through the **Greedy algorithm**;

3. highlight the selected edges;

4. is this matching a maximal one? ____ why? ____________________

5. provide the corresponding plan on the bottom





Aircraft number **1** covers routes ________________

Aircraft number **2** covers routes ________________

Aircraft number **3** covers routes ________________

Aircraft number **4** covers routes ________________

Aircraft number **5** covers routes ________________

Exercise 3

Consider the *TDMA* system defined by traffic matrix $T = \begin{pmatrix} 4 & 0 & 3 \\ 2 & 2 & 5 \\ 4 & 5 & 2 \end{pmatrix}$. Determine the *switch modes*, and the corresponding transmission times, to perform all transmissions in minimal time.

In the decomposition algorithm, select the switch modes according to the following rule: Select the element of maximum value in the first row of current matrix T; select the element of maximum value in the second row of current matrix T among those that are in a non-covered column; ...

Algorithm Equalize_sums:

$t^* = \quad ; (a) = (\quad , \quad , \quad); \quad (b) = (\quad , \quad , \quad);$

$i = 1: \quad (a) = (\quad , \quad , \quad); \quad (b) = (\quad , \quad , \quad);$

$i = 2: \quad (a) = (\quad , \quad , \quad); \quad (b) = (\quad , \quad , \quad);$

$i = 3: \quad (a) = (\quad , \quad , \quad); \quad (b) = (\quad , \quad , \quad);$

$$S = \left(\begin{array}{c|c|c} \quad & \quad & \quad \\ \hline \quad & \quad & \quad \\ \hline \quad & \quad & \quad \end{array}\right).$$

$$T = T + S = \left(\begin{array}{c|c|c} \quad & \quad & \quad \\ \hline \quad & \quad & \quad \\ \hline \quad & \quad & \quad \end{array}\right).$$

Algorithm Decompose:

$$k = 1\text{:}\ P_1 = \left(\begin{array}{c|c|c} \quad & \quad & \quad \\ \hline \quad & \quad & \quad \\ \hline \quad & \quad & \quad \end{array}\right);\ \lambda_1 = \qquad ;\ T = \left(\begin{array}{c|c|c} \quad & \quad & \quad \\ \hline \quad & \quad & \quad \\ \hline \quad & \quad & \quad \end{array}\right).$$

$$k = 2\text{:}\ P_2 = \left(\begin{array}{c|c|c} \quad & \quad & \quad \\ \hline \quad & \quad & \quad \\ \hline \quad & \quad & \quad \end{array}\right);\ \lambda_2 = \qquad ;\ T = \left(\begin{array}{c|c|c} \quad & \quad & \quad \\ \hline \quad & \quad & \quad \\ \hline \quad & \quad & \quad \end{array}\right).$$

$$k = 3\text{:}\ P_3 = \left(\begin{array}{c|c|c} \quad & \quad & \quad \\ \hline \quad & \quad & \quad \\ \hline \quad & \quad & \quad \end{array}\right);\ \lambda_3 = \qquad ;\ T = \left(\begin{array}{c|c|c} \quad & \quad & \quad \\ \hline \quad & \quad & \quad \\ \hline \quad & \quad & \quad \end{array}\right).$$

$$k = 4\text{:}\ P_4 = \left(\begin{array}{c|c|c} \quad & \quad & \quad \\ \hline \quad & \quad & \quad \\ \hline \quad & \quad & \quad \end{array}\right);\ \lambda_4 = \qquad ;\ T = \left(\begin{array}{c|c|c} \quad & \quad & \quad \\ \hline \quad & \quad & \quad \\ \hline \quad & \quad & \quad \end{array}\right).$$

4. Linear sum assignment problem

Assignment Problem $\equiv$ Weighted Matching Problem

- $G = (U, V; E)$ = bipartite graph with $|U| = |V| = n$, and c_{ij} = cost of edge $[i, j]$;
- determine a minimum cost perfect matching in G.

- Mathematical model

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1, 2, \ldots, n), \\
& \sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, 2, \ldots, n), \\
& x_{ij} \in \{0, 1\} \qquad (i, j = 1, 2, \ldots, n).
\end{aligned}
$$

with $x_{ij} = 1$ iff vertex $i \in U$ is matched to vertex $j \in V$, **or, equivalently,**

with $x_{ij} = 1$ iff row i of $[c_{ij}]$ is assigned to column j of $[c_{ij}]$.

Applications of the Assignment Problem

- **Subproblem in many important combinatorial optimization problems:**
  - quadratic assignment problem;
  - traveling salesman problem;
  - vehicle routing problems.
- **Personnel assignment.** (classical)
- **Railway systems:** assigning engines to trains due to traffic constraints.
- **Military operations:** assign interceptors to attacking missiles.
- **Scheduling:** optimal assignment of jobs to machines.
- **Depletion of inventory** of items having a known age a, and a value $f(a)$.
- ...

Assignment Problem (AP)

- Mathematical model

$$\min \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1, 2, \ldots, n), \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, 2, \ldots, n), \tag{3}$$

$$x_{ij} \in \{0, 1\} \qquad (i, j = 1, 2, \ldots, n). \tag{4}$$

with $x_{ij} = 1$ iff row i is assigned to column j.

Let us define a vector x' obtained by concatenating the rows of the x matrix and a vector c' obtained by concatenating the rows of the cost matrix.

The assignment constraints (2) and (3) define a $2n \times n^2$ matrix:

Constraint matrix:

$$
\begin{array}{l}
x' = (x_{11} \quad \cdots \quad x_{1n} \;\; x_{21} \quad \cdots \quad x_{2n} \quad \cdots \quad x_{n1} \quad \cdots \quad x_{nn})\\
c' = (c_{11} \quad \cdots \quad c_{1n} \;\; c_{21} \quad \cdots \quad c_{2n} \quad \cdots \quad c_{n1} \quad \cdots \quad c_{nn})
\end{array}
$$

$$
\begin{array}{r}
1\\ \\ \\ n\\ n+1\\ \\ \\ 2n
\end{array}
\left(\begin{array}{cccc|cccc|c|cccc}
1&1&\cdots&1&&&&&&&&&\\
&&&&1&1&\cdots&1&&&&&\\
&&&&&&&&\cdots&&&&\\
&&&&&&&&&1&1&\cdots&1\\
\hline
1&&&&1&&&&&1&&&\\
&1&&&&1&&&&&1&&\\
&&\cdots&&&&\cdots&&\cdots&&&\cdots&\\
&&&1&&&&1&&&&&1
\end{array}\right)
=
\begin{array}{c}1\\1\\\cdots\\1\\1\\1\\\cdots\\1\end{array}
$$

$$A$$

The AP is the Integer (binary) Linear Program (ILP):

$$
\begin{array}{ll}
\min & c'x\\
\text{s.t.} & Ax = \underline{\mathbf{1}}\\
& x \in \{0,1\}
\end{array}
$$

A general ILP is $\mathcal{NP}$-hard hence it cannot be solved in polynomial time (unless $\mathcal{P} = \mathcal{NP}$).
An LP (e.g., this ILP above with $x \in \{0,1\}$ replaced by $x \geq 0$) can be solved in polynomial time.

Properties of the constraint matrix

$$
\begin{array}{r}
1 \\ \\ \\ n \\ n+1 \\ \\ \\ 2n
\end{array}
\left(
\begin{array}{cccc|cccc|c|cccc}
1 & 1 & \cdots & 1 & & & & & & & & & \\
 & & & & 1 & 1 & \cdots & 1 & & & & & \\
 & & & & & & & & \cdots & & & & \\
 & & & & & & & & & 1 & 1 & \cdots & 1 \\
\hline
1 & & & & 1 & & & & & 1 & & & \\
 & 1 & & & & 1 & & & & & 1 & & \\
 & & \cdots & & & & \cdots & & \cdots & & & \cdots & \\
 & & & 1 & & & & 1 & & & & & 1
\end{array}
\right)
=
\begin{array}{c}
1 \\ 1 \\ \cdots \\ 1 \\ 1 \\ 1 \\ \cdots \\ 1
\end{array}
$$

- Remind the vertex-edge incidence matrix of a graph:
  - one row per vertex;
  - one column per edge;
  - $M_{ij} = \begin{cases} 1 & \text{if edge } j \text{ is incident to vertex } i \\ 0 & \text{otherwise} \end{cases}$

1. **The constraint matrix is the vertex-edge incidence matrix of the bipartite graph** $G = (U, V; E)$ whose **minimum cost perfect matching** solves the AP.

Properties of the constraint matrix (cont'd)

$$
\begin{array}{r}
1 \\ \\ \\ n \\ n+1 \\ \\ \\ 2n
\end{array}
\left(\begin{array}{cccc|cccc|c|cccc}
1 & 1 & \cdots & 1 & & & & & & & & & \\
 & & & & 1 & 1 & \cdots & 1 & & & & & \\
 & & & & & & & & \cdots & & & & \\
 & & & & & & & & & 1 & 1 & \cdots & 1 \\
\hline
1 & & & & 1 & & & & & 1 & & & \\
 & 1 & & & & 1 & & & & & 1 & & \\
 & & \cdots & & & & \cdots & & \cdots & & & \cdots & \\
 & & & 1 & & & & 1 & & & & & 1
\end{array}\right)
=
\begin{array}{c} 1 \\ 1 \\ \cdots \\ 1 \\ 1 \\ 1 \\ \cdots \\ 1 \end{array}
$$

- **Definition:** An $m \times n$ integer matrix A is **totally unimodular (TUM)** iff each square submatrix B satisfies $\det(B) \in \{0, +1, -1\}$.
- **Theorem:** If an ILP $\{\min c'x : Ax = b, x \geq 0, x \text{ integer}\}$ has a TUM matrix A then its LP relaxation $\{\min c'x : Ax = b, x \geq 0\}$ has integer (hence optimal) solutions $\forall$ integer b.
- **Theorem:** An integer matrix A with $a_{ij} \in \{0, +1, -1\}$ $\forall\ i, j$ is TUM if
  (1) no column has more than two nonzero elements, and
  (2) the rows can be partitioned in two sets I_1, I_2 such that
  - if a column has two elements of the same sign, their rows are in different sets;
  - if a column has two elements of different sign, their rows are in the same set.

2. **The constraint matrix is TUM** (Proof: $I_1 = \{1, \ldots, n\}$, $I_2 = \{n+1, \ldots, 2n\}$)
   $\Rightarrow$ **The continuous relaxation of its ILP model provides the optimal solution.**

Duality

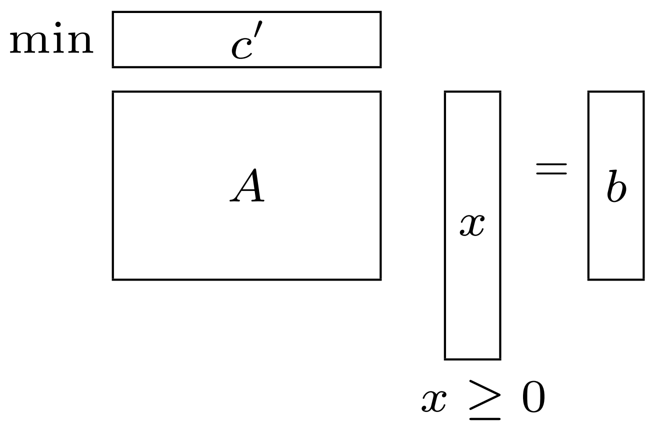
Recall the Duality theory of Linear Programming (discovered in the Fifties):

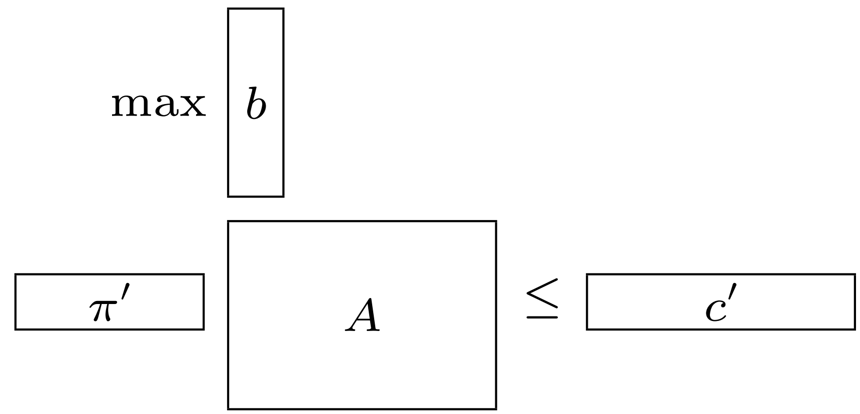
(Primal) $\min c'x$

$$\begin{aligned} Ax &= b \\ x &\geq 0 \end{aligned}$$

(Dual) $\max \pi' b$

$$\pi' A \leq c'$$





	Primal	Dual
• **# variables**	n	m $(m < n)$
• **objective function**	coefficients c'	coefficients b'
• **# constraints**	m	n $(n > m)$
• **constraints**	coefficients a'_i (ith row)	coefficients A_j (jth column)
• **right hand side**	b	c

Duality (cont'd)

(Primal) $\min c'x$

$$\begin{aligned} Ax &= b \\ x &\geq 0 \end{aligned}$$

(Dual) $\max \pi' b$

$$\pi' A \leq c'$$

Note: This primal LP is in **standard form** (only equality constraints). In general, it can also have inequalities (similar dual, with $\pi_i \geq 0$ constraints for the corresponding dual variables).

Duality theorem: If a (primal) Linear Program has a finite optimal solution, then

1. its dual has a finite optimal solution;

2. the two optimal solutions have the same value.

Complementary slackness: Given a general (primal) linear program (with equality and inequality constraints), two solutions, x and π, respectively **feasible** for the primal and its dual, are optimal **if and only if**

$$u_i = \pi_i(a_i' x - b_i) = 0 \text{ for } i = 1, \ldots, m \quad \textbf{AND}$$

$$v_j = (c_j - \pi' A_j) x_j = 0 \text{ for } j = 1, \ldots, n.$$

Observation: If the primal is in **standard form** (like for the AP)

any feasible primal solution x satisfies the first group of slackness constraints.

Duality of the Assignment problem

- Distinct dual variables u_i and v_j for the two sets of assignment constraints (2) and (3).
- **Dual problem:**

$$\max \quad \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \tag{5}$$

$$\text{s.t.} \quad u_i + v_j \le c_{ij} \qquad (i, j = 1, 2, \ldots, n). \tag{6}$$

- **Complementary slackness:** Two feasible solutions x, (u, v) are optimal if and only if

$$x_{ij}(c_{ij} - u_i - v_j) \quad = \quad 0 \qquad (i, j = 1, 2, \ldots, n). \tag{7}$$

- **Duality: if (7) holds then**

$$\max \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \quad = \quad \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{8}$$

- The Duality theory of linear programming was formulated by **John von Neumann in 1947**
- **but anticipated in 1931 by Jenö Egerváry**.

Who was Jenö Egerváry?

Jenö Egerváry was born in Debrecen (Hungary) in 1891. He had a wide scientific production, ranging from analysis and function theory to theoretical physics and geometry.

In the Fifties he was head of the Dept. of Mathematics of the Polytechnic University of Budapest.

In 1956 the Hungarian Revolution was suppressed by the invasion of the Soviet Army, which re-established Communist authority. In the subsequent repression period, he was forced to retire.

In 1958, fearing to be imprisoned for specious accusations, Egerváry committed suicide.

Egerváry's Theorem

- Covering system = set of *lines* (rows and columns) that contain the ith row of C with multiplicity λ_i and the jth column with multiplicity μ_j, and satisfy

$$\lambda_i + \mu_j \geq c_{ij} \quad (i, j = 1, \dots, n). \tag{9}$$

- A covering system of minimum value

$$\sum_{k=1}^{n} (\lambda_k + \mu_k) \tag{10}$$

  is called a *minimal covering system*.

- **Minimizing (10) subject to (9) is the dual problem of a maximization AP!!!**.
- **Egerváry's Theorem (1931)**:
  If (c_{ij}) is an $n \times n$ matrix of non-negative integers then, subject to condition (9), we have

$$\min \sum_{k=1}^{n} (\lambda_k + \mu_k) = \max_{\varphi} \sum_{i=1}^{n} c_{i\varphi(i)}. \tag{11}$$

- In other words, **the primal and the dual problem have the same solution value !!!**.

Initialization

- The AP has been solved with a variety of approaches: primal, dual, primal-dual and others.
- Most algorithms for the AP have a **preprocessing phase** to determine

  (i) a feasible dual solution, and

  (ii) a partial primal solution (where less than n rows are assigned)

  satisfying the complementary slackness conditions.
- **Notation** (in addition to $X = \{x_{ij}\}$)

$$row(j) = \begin{cases} i & \text{if column } j \text{ is assigned to row } i, \\ 0 & \text{if column } j \text{ is not assigned} \end{cases} \qquad (j = 1, 2, \ldots, n).$$

$$\varphi(i) = \begin{cases} j & \text{if row } i \text{ is assigned to column } j, \\ 0 & \text{if row } i \text{ is not assigned} \end{cases} \qquad (i = 1, 2, \ldots, n).$$

Initialization algorithm

Procedure Basic_preprocessing

comment: remind, compl. slackness $x_{ij}(c_{ij} - u_i - v_j) = 0$, dual constraints $u_i + v_j \le c_{ij}$;

for $i := 1$ **to** n **do** $u_i := \min\{c_{ij} : j = 1, 2, \ldots, n\}$;

for $j := 1$ **to** n **do** $v_j := \min\{c_{ij} - u_i : i = 1, 2, \ldots, n\}$;

comment: find a partial feasible solution;

for $i := 1$ **to** n **do for** $j := 1$ **to** n **do** $x_{ij} := 0$;

for $j := 1$ **to** n **do** $row(j) := 0$;

for $i := 1$ **to** n **do**

    for $j := 1$ **to** n **do**

        if $(row(j) = 0$ **and** $c_{ij} - u_i - v_j = 0)$ **then** $x_{ij} := 1$, $row(j) := i$ and **break**

    endfor

endfor

- The u_i and v_j values satisfy the dual constraints $c_{ij} - u_i - v_j \ge 0 \; \forall \; i, j$;
- the x_{ij} values satisfy

(i) the complementary slackness conditions $x_{ij}(c_{ij} - u_i - v_j) = 0 \; \forall \; i, j$;

(ii) the primal constraints $\sum_{j=1}^{n} x_{ij} = 1 \; \forall \; i$, $\sum_{i=1}^{n} x_{ij} = 1 \; \forall \; j$ **with '$\le$' instead of '$=$'**;

- the solution is **dual feasible** and **primal infeasible**.

Example

$$\underbrace{\begin{pmatrix} 7 & 9 & 8 & 9 \\ 2 & 8 & 5 & 7 \\ 1 & 6 & 6 & 9 \\ 3 & 6 & 2 & 2 \end{pmatrix}}_{C} \qquad \begin{array}{c} 7 \\ 2 \\ 1 \\ 2 \end{array} \underbrace{\begin{pmatrix} 7 & 9 & 8 & 9 \\ 2 & 8 & 5 & 7 \\ 1 & 6 & 6 & 9 \\ 3 & 6 & 2 & 2 \end{pmatrix}}_{C} \qquad \begin{array}{cc} & \begin{array}{cccc} 0 & 2 & 0 & 0 \end{array} \\ \begin{array}{c} 7 \\ 2 \\ 1 \\ 2 \end{array} & \underbrace{\begin{pmatrix} 7 & 9 & 8 & 9 \\ 2 & 8 & 5 & 7 \\ 1 & 6 & 6 & 9 \\ 3 & 6 & 2 & 2 \end{pmatrix}}_{C} \end{array}$$

- $\overline{C} = (\overline{c}_{ij}) = (c_{ij} - u_i - v_j)$;
- $\underbrace{\begin{pmatrix} \underline{\mathbf{0}} & 0 & 1 & 2 \\ 0 & 4 & 3 & 5 \\ 0 & 3 & 5 & 8 \\ 1 & 2 & \underline{\mathbf{0}} & 0 \end{pmatrix}}_{\overline{C}} \quad \underbrace{\begin{pmatrix} \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \end{pmatrix}}_{X}$ **initial solution (only 0 reduced costs c_{ij})**
- $row = (1,\ 0,\ 4,\ 0)$ ($\Leftrightarrow \varphi = (1,\ 0,\ 0,\ 3)$);
- It can be shown that the $\overline{c}_{ij}$ values are the **reduced costs** of the associated LP.

Observation: C and $\overline{C}$ are equivalent

- Consider any feasible primal solution $X = (x_{ij})$:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij} - u_i - v_j)x_{ij}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} - \sum_{i=1}^{n} u_i \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} v_j \sum_{i=1}^{n} x_{ij}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} - \sum_{i=1}^{n} u_i - \sum_{j=1}^{n} v_j$$

- The objective function values induced by C and $\overline{C}$ differ by the same constant

$$\sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j .$$

We now have all the elements for understanding the **Hungarian Algorithm**, invented in 1953 by **Harold W. Kuhn**.