Models and Algorithms for Matching and Assignment Problems

Silvano Martello

DEI "Guglielmo Marconi", Università di Bologna, Italy

This work by is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

1. Introduction

This work by is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

The Assignment Problem

- Assign in the best way n jobs to n candidates (one each), knowing the (presumed) time, c_{ij}, that each candidate i would need for each job j;
- **Example:** $n = 3$:

$$\begin{array}{ll|ccc|}
 & & \multicolumn{3}{c}{\text{Job}} \\
 & & 1 & 2 & 3 \\
\hline
 & 1 & 20 & 60 & 30 \\
\text{Candidate} & 2 & 80 & 40 & 90 \\
 & 3 & 50 & 70 & 80 \\
\hline
\end{array}$$

By enumeration ($3! = 6$ possible solutions): $1 \leftrightarrow 3$, $2 \leftrightarrow 2$, $3 \leftrightarrow 1$; time $= 120$.

- **Mathematical model:** $x_{ij} = \begin{cases} 1 & \text{if candidate } i \text{ performs job } j \\ 0 & \text{otherwise} \end{cases}$

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} && \text{total time needed} \\
& \sum_{i=1}^{n} x_{ij} = 1 \quad \text{for } j = 1, \dots, n && \text{one candidate per job} \\
& \sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 1, \dots, n && \text{one job per candidate} \\
& x_{ij} \in \{0, 1\} \quad \text{for } i, j = 1, \dots, n
\end{aligned}$$

The Assignment Problem (cont'd)

The Assignment Problem (AP) can be formulated in a mathematically simpler way.
As we want to assign to each row i a different column, say $\varphi(i)$, we can state it as:

Given an $n \times n$ matrix $C = (c_{ij})$, find a permutation φ of $\{1, 2, \ldots, n\}$ that minimizes

$$\sum_{i=1}^{n} c_{i\varphi(i)},$$

In spite of its simplicity, in the last sixty years AP attracted hundreds of researchers, accompanying and sometimes anticipating the development of Combinatorial Optimization.

Although its origins date back to the Eighteenth and Nineteenth century (Monge, Jacobi),

the study of its combinatorial structure to the Twenties (Frobenius, König, Egerváry),

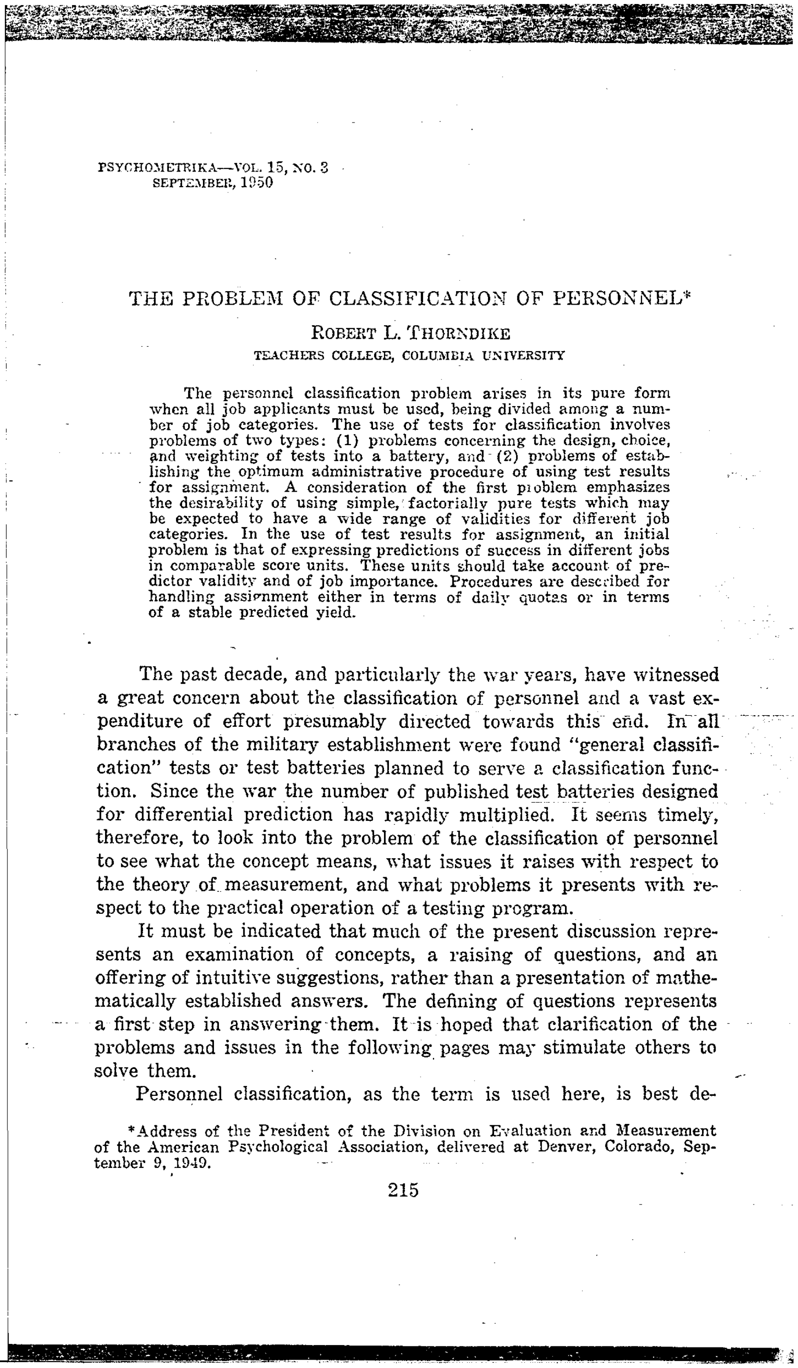
and the first (exponential) implicit enumeration algorithm to the Forties (Easterfield),

the problem was formulated in modern way in the Fifties by a psychologist (!!!), R. L. Thorndike, President of the Division on Evaluation and Measurement, American Psychological Association:

"Given: A set of ... N vacancies to be filled, and N individuals to be used in filling them.

Required: To assign the individuals to the jobs in such a way that the average success of all the individuals in all the jobs to which they are assigned will be a maximum."

("The problem of classification of personnel", *Psychometrica*, 1950).



PSYCHOMETRIKA—VOL. 15, NO. 3
SEPTEMBER, 1950

THE PROBLEM OF CLASSIFICATION OF PERSONNEL*

ROBERT L. THORNDIKE
TEACHERS COLLEGE, COLUMBIA UNIVERSITY

The personnel classification problem arises in its pure form when all job applicants must be used, being divided among a number of job categories. The use of tests for classification involves problems of two types: (1) problems concerning the design, choice, and weighting of tests into a battery, and (2) problems of establishing the optimum administrative procedure of using test results for assignment. A consideration of the first problem emphasizes the desirability of using simple, factorially pure tests which may be expected to have a wide range of validities for different job categories. In the use of test results for assignment, an initial problem is that of expressing predictions of success in different jobs in comparable score units. These units should take account of predictor validity and of job importance. Procedures are described for handling assignment either in terms of daily quotas or in terms of a stable predicted yield.

The past decade, and particularly the war years, have witnessed a great concern about the classification of personnel and a vast expenditure of effort presumably directed towards this end. In all branches of the military establishment were found "general classification" tests or test batteries planned to serve a classification function. Since the war the number of published test batteries designed for differential prediction has rapidly multiplied. It seems timely, therefore, to look into the problem of the classification of personnel to see what the concept means, what issues it raises with respect to the theory of measurement, and what problems it presents with respect to the practical operation of a testing program.

It must be indicated that much of the present discussion represents an examination of concepts, a raising of questions, and an offering of intuitive suggestions, rather than a presentation of mathematically established answers. The defining of questions represents a first step in answering them. It is hoped that clarification of the problems and issues in the following pages may stimulate others to solve them.

Personnel classification, as the term is used here, is best de-

*Address of the President of the Division on Evaluation and Measurement of the American Psychological Association, delivered at Denver, Colorado, September 9, 1949.

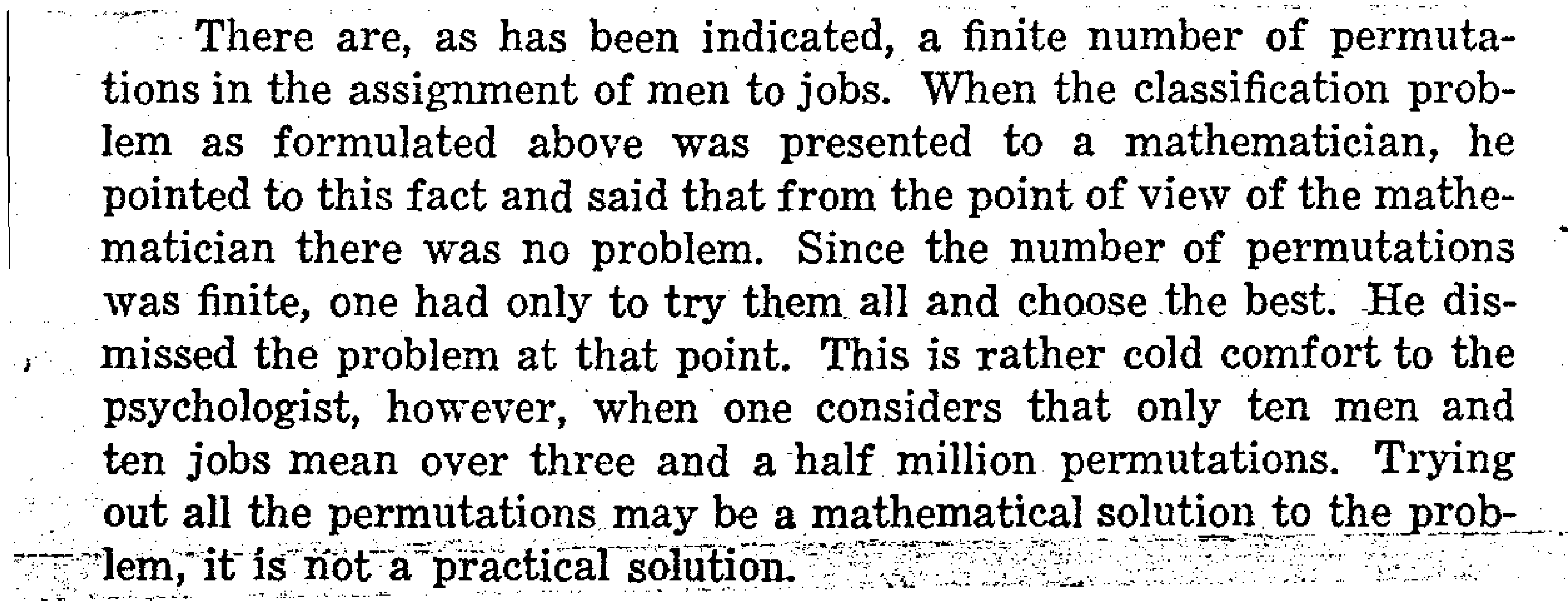
215

Preliminary, let's consider a strange question:

Could a psychologist of the Fifties apprehend complexity issues

better than a mathematician?

To answer, let's read a paragraph in the Thorndike paper.



There are, as has been indicated, a finite number of permutations in the assignment of men to jobs. When the classification problem as formulated above was presented to a mathematician, he pointed to this fact and said that from the point of view of the mathematician there was no problem. Since the number of permutations was finite, one had only to try them all and choose the best. He dismissed the problem at that point. This is rather cold comfort to the psychologist, however, when one considers that only ten men and ten jobs mean over three and a half million permutations. Trying out all the permutations may be a mathematical solution to the problem, it is not a practical solution.

Question:

Could a psychologist of the Fifties apprehend complexity issues better than a mathematician?

Answer: YES

Mathematicians will understand complexity 15 years later! (Edmonds, 1965)

Note:

- a **10** $\times$ **10** AP **can** be solved by a modern **PC** through enumeration;
- a **20** $\times$ **20** AP **cannot** be solved by a modern **PC** through enumeration in less than one century, **but today**
- a **20** $\times$ **20** AP **can** be solved by the **fastest supercomputers** through enumeration.
- Fastest supercomputers: **Cray XT system Jaguar**, **IBM Blue Gene/P Intrepid**.
- **Jaguar:** **Speed:** $\sim$ 1 Petaflop (10^{15} calculations per second);
  Processors: 182,000 Opterons, running at 2.3 gigahertz;
  Memory: 362 TB (Tera bytes, 1 TB $= 10^{12}$ bytes) of high speed memory.

  However
- a **24** $\times$ **24** AP **cannot** be solved by a Jaguar/Blue Gene through enumeration in less than one century.

The beginning

In his reminiscences "On the origin of the Hungarian method" (1991), **H.W. Kuhn** reminds that he was attracted to the problem in 1953, when C. B. Tompskins (1912–1971), a pioneer in numerical analysis and computing, was unsuccessfully trying to program a SWAC (Standards Western Automatic Computer) to solve 10×10 AP instances.

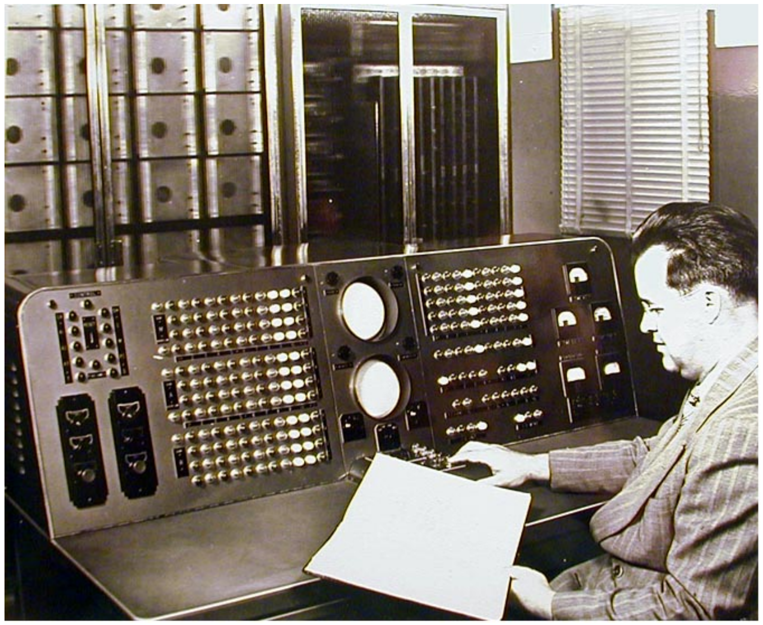
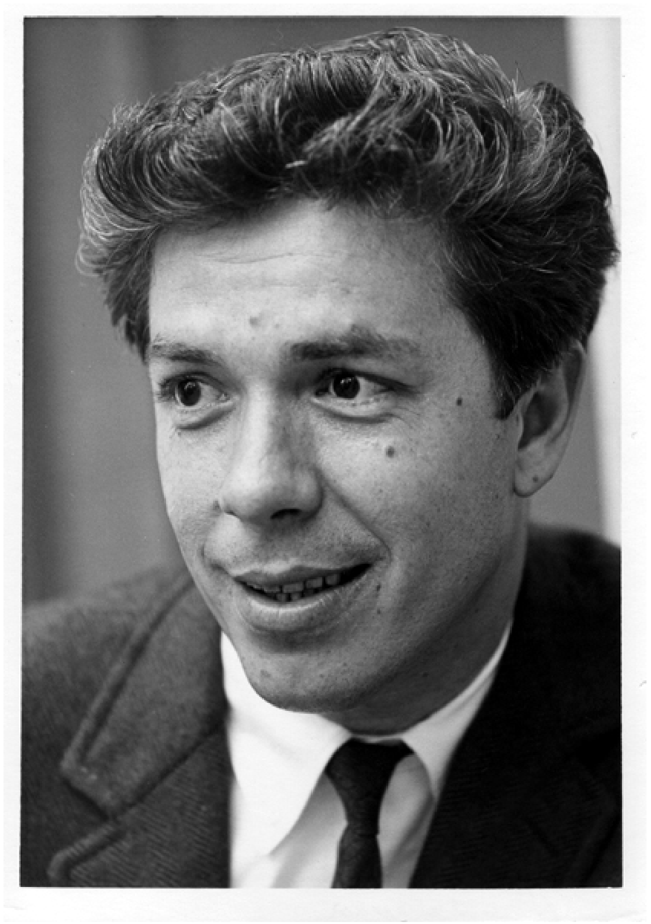


Figure 1: SWAC

Who is H.W. Kuhn?



Harold W. Kuhn (b. 1925) is a giant in the history of optimization. His **Hungarian method** for the AP is one of the fundamental results that gave birth to Combinatorial Optimization.

The **Kuhn-Tucker conditions** (necessary optimality conditions for nonlinear programming) are a cornerstone of Mathematical Programming.

His book with Sylvia Nasar *The Essential John Nash* (remind the movie A beautiful mind) extended his fame to a wider audience.

The beginning (cont'd)

In that period Kuhn was reading the classical graph theory book by the Hungarian mathematician **Dénes König**,

Theorie der Endlichen und Unendlichen Graphen (1936),

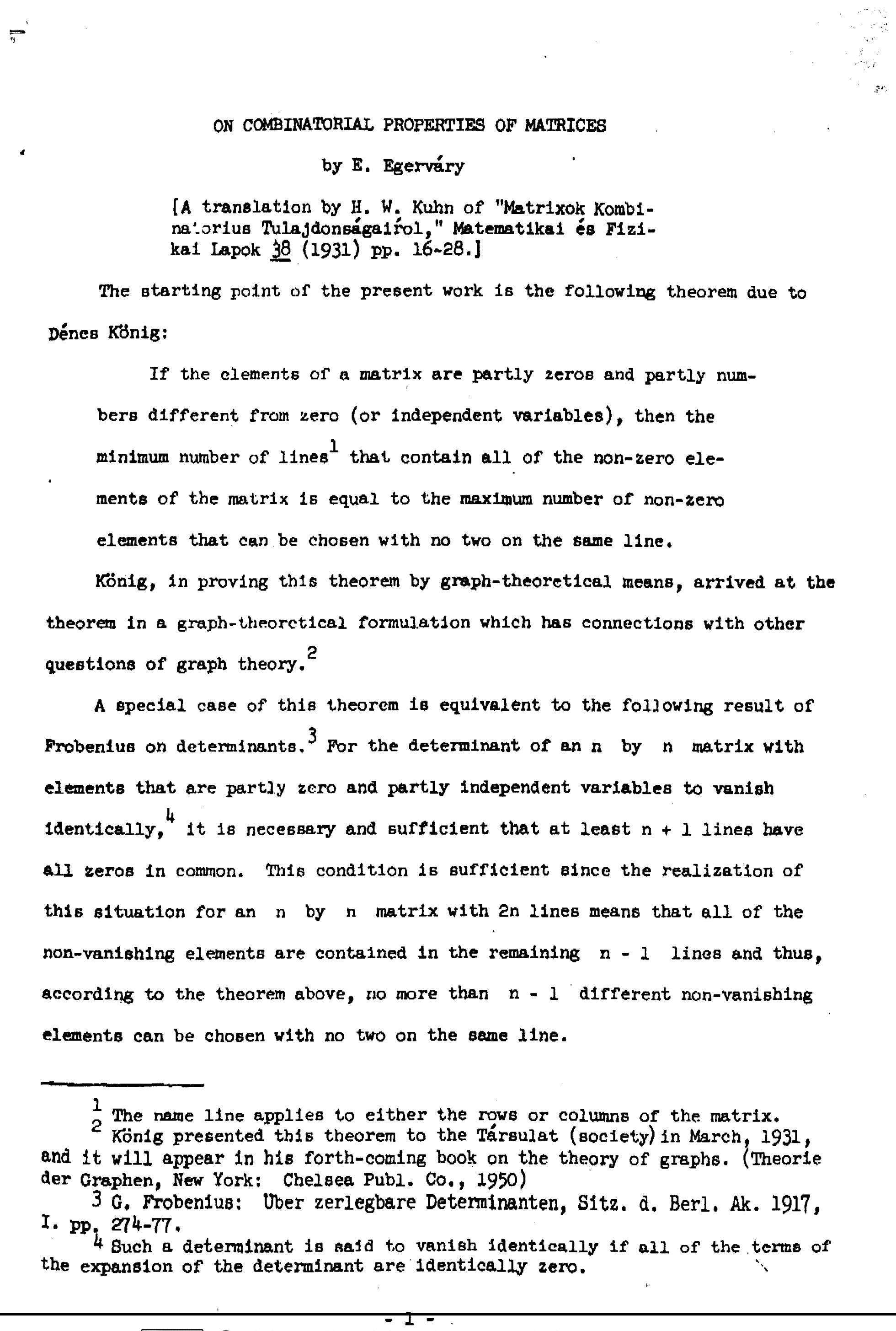
the first book ever published on **Graph Theory**, and encountered his augmenting path algorithm for the matching problem on bipartite graphs. (a fundamental algorithm that we will introduce in the following).

As we will see, the matching problem on a bipartite graph is a special case of assignment problem on a 0-1 matrix. Hence the question arose:

How can the König algorithm be extended to the general assignment problem?

A footnote in the book pointed to a 1931 paper by **Jenö Egerváry**, published in Hungarian on the journal *Matmatikai és Fizikai Lapok*.

Kuhn then translated Egerváry's paper with the help of a Hungarian dictionary and grammar, and published the translation as a Research Report of the George Washington University.



ON COMBINATORIAL PROPERTIES OF MATRICES

by E. Egerváry

[A translation by H. W. Kuhn of "Matrixok Kombinatorius Tulajdonságairól," Matematikai és Fizikai Lapok 38 (1931) pp. 16-28.]

The starting point of the present work is the following theorem due to Dénes König:

> If the elements of a matrix are partly zeros and partly numbers different from zero (or independent variables), then the minimum number of lines[1] that contain all of the non-zero elements of the matrix is equal to the maximum number of non-zero elements that can be chosen with no two on the same line.

König, in proving this theorem by graph-theoretical means, arrived at the theorem in a graph-theoretical formulation which has connections with other questions of graph theory.[2]

A special case of this theorem is equivalent to the following result of Frobenius on determinants.[3] For the determinant of an n by n matrix with elements that are partly zero and partly independent variables to vanish identically,[4] it is necessary and sufficient that at least n + 1 lines have all zeros in common. This condition is sufficient since the realization of this situation for an n by n matrix with 2n lines means that all of the non-vanishing elements are contained in the remaining n - 1 lines and thus, according to the theorem above, no more than n - 1 different non-vanishing elements can be chosen with no two on the same line.

---

[1] The name line applies to either the rows or columns of the matrix.

[2] König presented this theorem to the Tarsulat (society) in March, 1931, and it will appear in his forth-coming book on the theory of graphs. (Theorie der Graphen, New York: Chelsea Publ. Co., 1950)

[3] G. Frobenius: Über zerlegbare Determinanten, Sitz. d. Berl. Ak. 1917, I. pp. 274-77.

[4] Such a determinant is said to vanish identically if all of the terms of the expansion of the determinant are identically zero.

- 1 -

The beginning (cont'd)

Using Egervárys reduction and Königs maximum matching algorithm, in the fall of 1953 Kuhn solved several 12×12 assignment problems **by hand**.

Each of these examples took under two hours to solve. *This must have been one of the last times when pencil and paper could beat the largest and fastest electronic computer in the world.*

The algorithm was christened the Hungarian algorithm in honor of these two mathematicians and published in two famous papers on *Naval Research Logistics Quarterly*:

"The Hungarian method for the assignment problem" (1955)

"Variants of the Hungarian method for the assignment problem" (1956)

Reprinted in 2005: "The Hungarian method for the assignment problem"

Over 2500 citations on Google Scholar.

Contents

This presentation is based on the book **Assignment Problems**

by R. Burkard, M. Dell'Amico and S. Martello, SIAM, 2009 (2nd Edition 2012).

Home page (excerpts, applets, freeware, ...): `www.assignmentproblems.com`

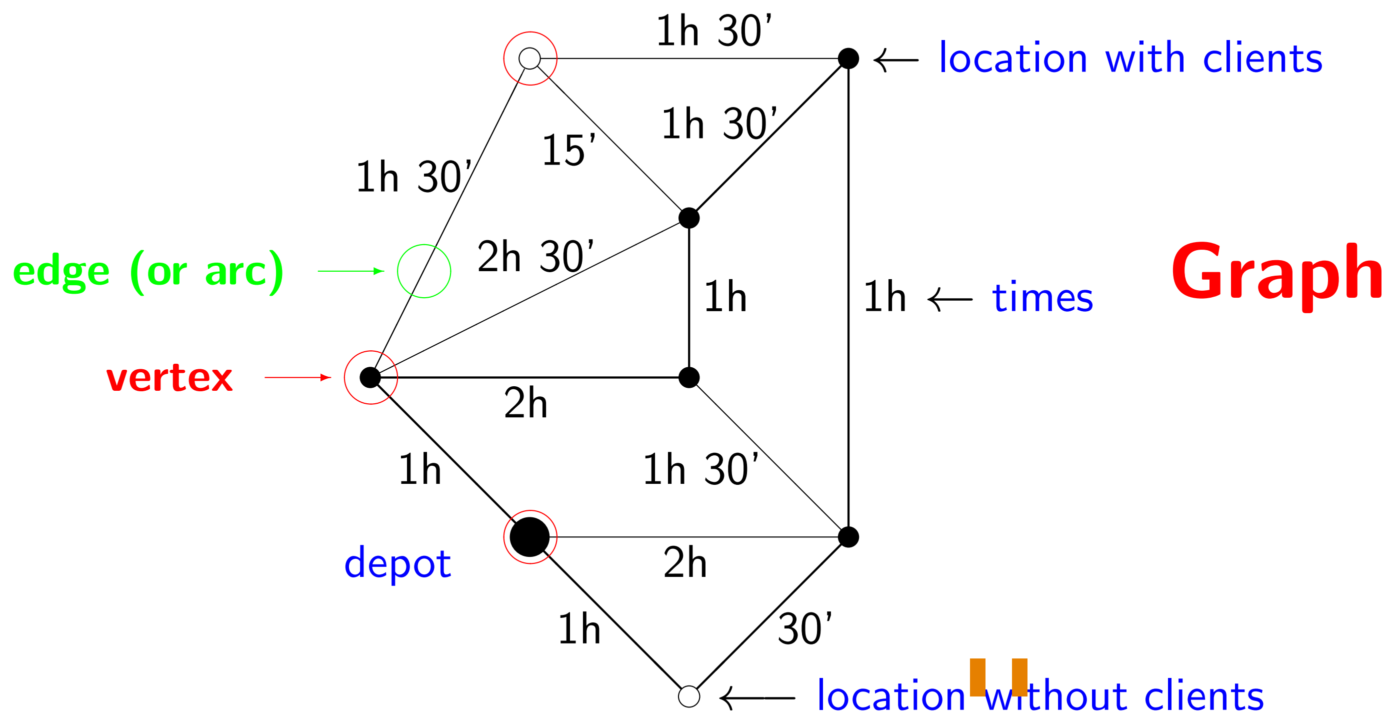
2. Theoretical foundations: matching problems

A tour of Combinatorial Optimization from 1916 to 1949

This work by is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

Graph theory: definitions and notations

- What is a graph? Let's start with an easy example.
  The depot of a factory and its clients are modeled by a road map with presumed traveling times for each road: this is a **Graph**:



- Graphs

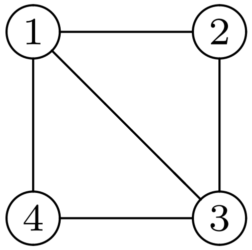
$\begin{cases} \textbf{simple}: & \text{at most one edge between two vertices. We will alway consider simple graphs;} \\ \textbf{multiple}: & \text{more edges between two vertices ar possible (usually transformed to simple graphs);} \end{cases}$

undirected graphs

- $V = \{1, \ldots, n\}$ (*vertices*)
- $E = \{e_1, \ldots, e_m\}$ (*edges*)

  $e_i = [j, k] \equiv [k, j]$

  traveling allowed in both directions
- $G = (V, E)$



$$V = \{1, 2, 3, 4\}$$
$$\begin{aligned} E &= \{e_1, e_2, e_3, e_4, e_5\} \\ &= \{[1, 2], [2, 3], [3, 4], \\ &\quad [4, 1], [1, 3]\} \end{aligned}$$

Degree of a vertex: # incident edges.

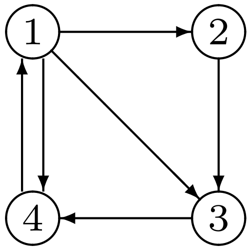
Spanning tree: $G' = (V, E')$ $(E' \subseteq E)$
which has no circuit and st $|E'| = n - 1$,
e.g., $E' = \{[1, 2], [1, 3], [1, 4]\}$

directed graphs

- $V = \{1, \ldots, n\}$ (*vertices*)
- $A = \{a_1, \ldots, a_m\}$ (*arcs*)

  $a_i = (j, k) \not\equiv (k, j)$

  traveling only allowed from j to k
- $G = (V, A)$



$$V = \{1, 2, 3, 4\}$$
$$\begin{aligned} A &= \{a_1, a_2, a_3, a_4, a_5, a_6\} \\ &= \{(1, 2), (2, 3), (3, 4), \\ &\quad (4, 1), (1, 4), (1, 3)\} \end{aligned}$$

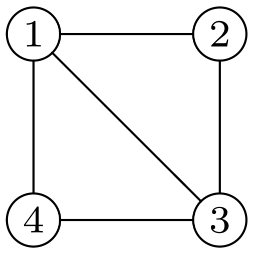
In-degree of a vertex = # entering arcs.
Out-degree of a vertex = # leaving arcs.

- **Weighted graphs** (both directed and undirected): each arc/edge has an associated *weight* (or *cost*, *length*, ...)

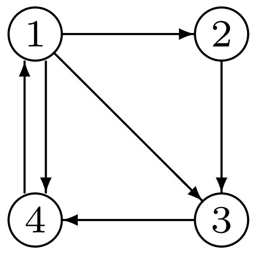
$$\begin{tikzcd} j \arrow[r, "41"] & k \end{tikzcd} \quad w(j,k) = w(a_i) = w_{jk} = 41$$

- *Path* = sequence of consecutive arcs/edges

$$\{j_1, j_2, \dots, j_\ell \ : (j_i, j_{i+1}) \in A \ \ ([j_i, j_{i+1}] \in E) \ \forall\ i \text{ and } j_i \not\equiv j_k \ \forall i \neq k\}.$$



$\{1,4,3\}$ = path from 1 to 3
$\equiv$ path from 3 to 1



$\{1,3,4\}$ = path from 1 to 4

- *Circuit* or *cycle* = path starting and ending on the same vertex.
$= \{j_1, j_2, \dots, j_\ell \equiv j_1\}$.

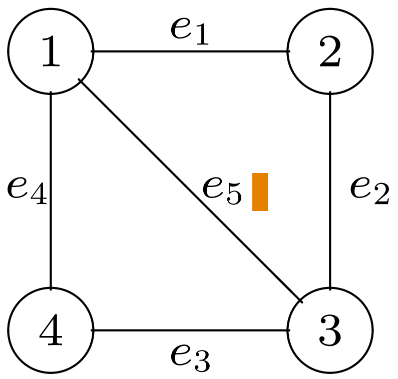
$\{1,4,3,1\}$ | $\{1,3,4,1\}$

Various data structures are used to represent a graph (adjacency matrix, cost matrix, forward star, ...). Let's see a couple that will be used in the following.

Adiacency matrix

- The **Adiacency matrix** of an undirected graph has
  - one row per vertex;
  - one column per vertex;
  - $M_{ij} = \begin{cases} 1 & \text{if edge } [i,j] \text{ exists} \\ 0 & \text{otherwise} \end{cases}$

- **Example:**



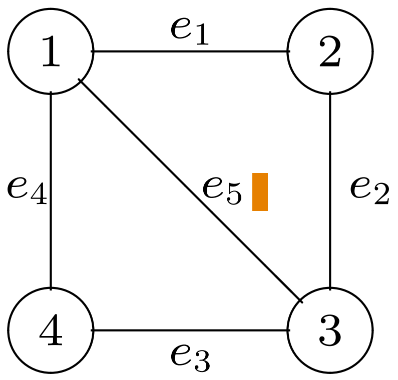
$$M = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

- This matrix is symmetric.

Vertex-edge incidence matrix

- The **Vertex-edge incidence matrix** of an undirected graph has
  - one row per vertex;
  - one column per edge;
  - $M_{ij} = \begin{cases} 1 & \text{if edge } j \text{ is incident to vertex } i \\ 0 & \text{otherwise} \end{cases}$

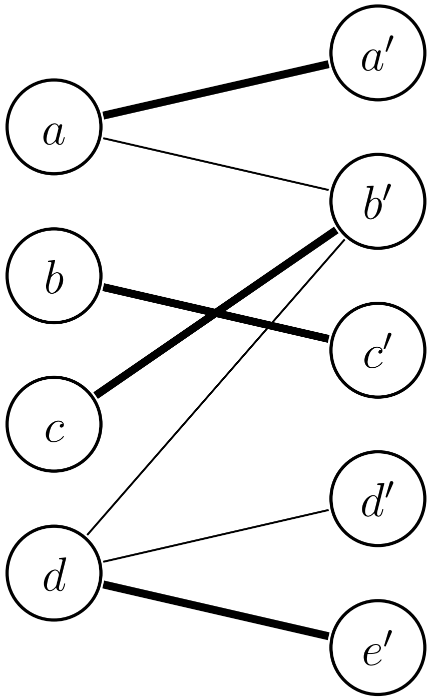
- **Example:**



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

- For each edge (column): two 1s (the two connected vertices);
- For each vertex (row): a 1 for each incident edge.

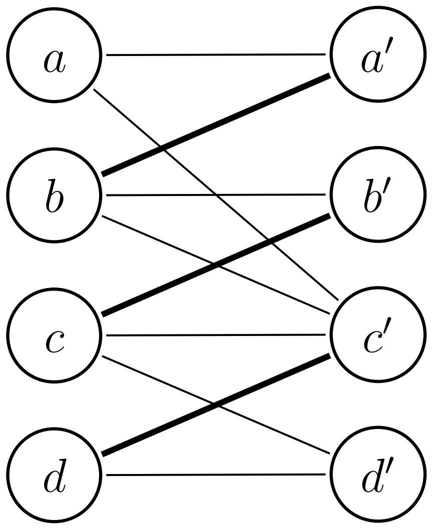
- A bipartite graph $G = (U, V; E)$ is defined by
  - two sets of vertices U and V;
  - a set E of edges, each connecting a vertex of U and a vertex of V.
- A matching $M \subseteq E$ is a subset of edges with no vertex in common.



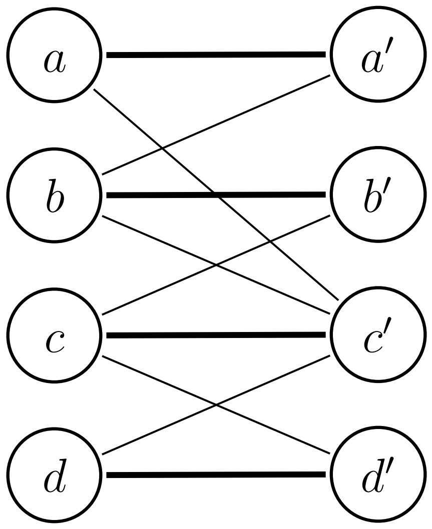
- If $|U| = |V|$, a matching of $|U|$ edges is called a Perfect matching.

Matchings:

- bipartite graph $G = (U, V; E)$;
- $M \subseteq E$ is a *matching*, if every vertex is incident with at most one edge;
- M is *perfect* if *every* vertex is incident with exactly one edge ($|U| = |V|$)



Matching



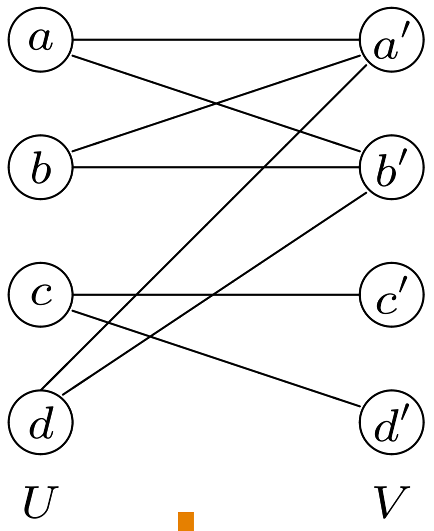
Perfect matching

The Maximum Matching problem

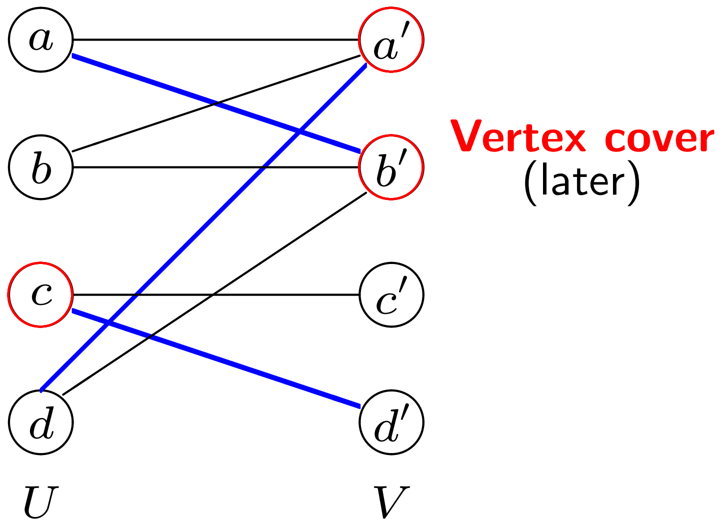
We will now concentrate on the following problem:

- Given a bipartite graph $G = (U, V; E)$ of $2n$ vertices
  - with vertex sets U, V satisfying $|U| = |V| = n$,
  - and edge set E, with $|E| = m$ and edges $[i, j]$ $(i \in U,\ j \in V)$,
- find a **matching**, i.e. a set M of **disjoint edges** (edges with no vertex in common)

  having maximum cardinality $|M|$.
- If $|M| = n$ the matching is called **perfect**.



$\longrightarrow$



Bipartite graph

Maximum matching
(non perfect)

Relationship with the Assignment Problem

Problem 1: given n jobs and n candidates, knowing, for each candidate i, if s/he is qualified for job j, assign one job to the maximum number of qualified candidates: **Models:**

1. Define an $n \times n$ matrix Q, having one row per candidate and one column per job, with

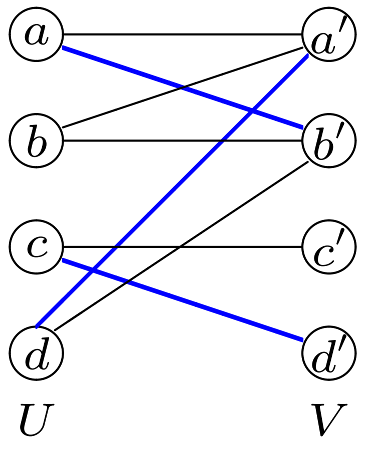
$$Q_{ij} = \begin{cases} 1 & \text{if candidate } i \text{ is qualified for job } j \\ 0 & \text{otherwise} \end{cases}$$

and solve a maximization **assignment problem**. **OR**

2. Define a bipartite graph $G = (U, V; E)$ having
   - one vertex $u \in U$ per candidate and one vertex $v \in V$ per job;
   - an edge $[i, j] \in E$ iff candidate i is qualified for job j

   and solve a **maximum matching problem**.

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



Note: Q is the **biadjacency matrix of bipartite graph** G: it has $|U|$ rows, $|V|$ columns, and $Q_{ij} = 1$ if edge $[i, j]$ with $i \in U$ and $j \in V$ exists, or $Q_{ij} = 0$ otherwise.

Observe the difference with the adjacency matrix of G, which has 8 rows and 8 columns.

Relationship with the Assignment Problem (cont'd)

Problem 2: given n jobs and n candidates, knowing, for each candidate i and job j, the profit, w_{ij}, obtained if candidate i performs job j;

assign one job to each candidate so that the overall profit is maximized. **Models:**

1. This is a maximization **assignment problem**. **OR**
2. Define a weighted bipartite graph $G = (U, V; E)$ having
   - one vertex $u \in U$ per candidate and one vertex $v \in V$ per job;
   - an edge $[i, j] \in E$ with weight w_{ij} = profit obtained if candidate i performs job j, and weight $w_{ij} = -M$ (M a very large value) if candidate i cannot perform job j.

   and find a matching of maximum profit (**weighted bipartite matching problem**).

$$W = \begin{pmatrix} \mathbf{6} & \mathbf{9} & -M & -M \\ \mathbf{7} & \mathbf{7} & -M & -M \\ -M & -M & \mathbf{6} & \mathbf{8} \\ \mathbf{8} & \mathbf{7} & -M & -M \end{pmatrix}$$

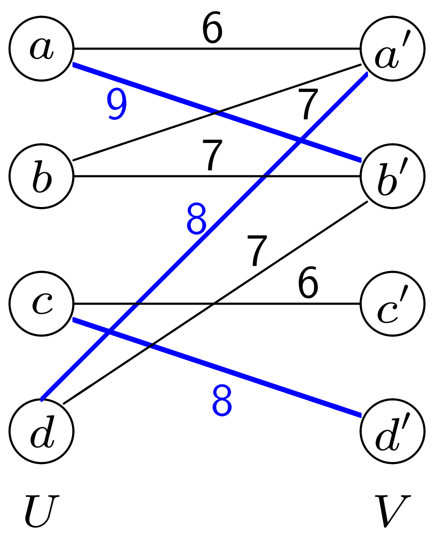


Figure 2: **Interpretation of maximum matching:** Philip Hall (1904 - 1982)

The marriage theorem (Hall, 1935)

- Bipartite graph $G = (U, V; E)$ ($|U| = |V| = n$, $|E| = m$, edges $[i, j] \in E$).

  Vertices of U = young ladies, vertices of V = young men;

- an edge $[i, j]$ indicates that Miss i is a friend of Mr j:

- **A perfect matching corresponds to a marriage of all young ladies and men where a couple can only be married if the partners are friends** (**Hermann Weyl**, 1949)

- **Marriage theorem** (existence of a perfect matching)
  Given a bipartite graph $G = (U, V; E)$ with $|U| = |V|$, let $F(i)$ = set of friends of Miss i;

  Given a subset U' of U, let $F(U') = \cup_{i \in U'} F(i)$.
  There exists a perfect matching (marriage) in G, if and only if for all subsets U' of U

$$|U'| \quad \leq \quad |F(U')| \qquad\qquad \textbf{(Hall condition).}$$

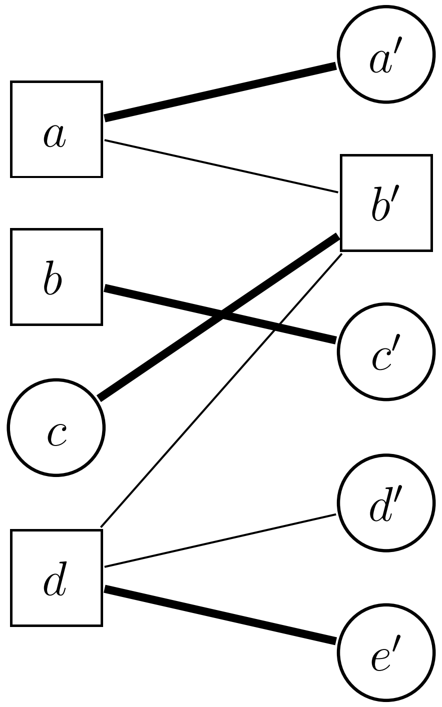
- Formulated in **1935**, christened in **1949**, but anticipated in **1916** by Denes König.

- We need a last **definition**: Given an undirected graph, a **Vertex cover** is a subset C of vertices such that every edge is incident with at least one vertex of C.

- **Problem:** Given an undirected graph, find a **vertex cover** C having **minimum cardinality** $|C|$. (See an example a couple of slides before.)

König's matching theorem (1916)

The maximum cardinality of a matching is equal to the minimum cardinality of a vertex cover.



Vertex cover $=$ subset C of vertices such that every edge is incident with at least one vertex of C (squares)

First full duality result!

Proof $=$ alternating path algorithm!

Who was Denes König?

Denes König was born in Budapest in 1884 to a family of Jewish origin, but was baptized as a Christian. He was a boy prodigy, publishing his first scientific paper (on mathematical recreations) in 1899, while still in a Budapest Gymnasium.

In the Forties he made efforts to help persecuted Jewish mathematicians.

In 1944 German forces occupied Hungary and installed a radically anti-semitic puppet government.

In October 1944, fearing to be ordered to move to the ghetto, König committed suicide.

König gave a **constructive proof** of his duality result:

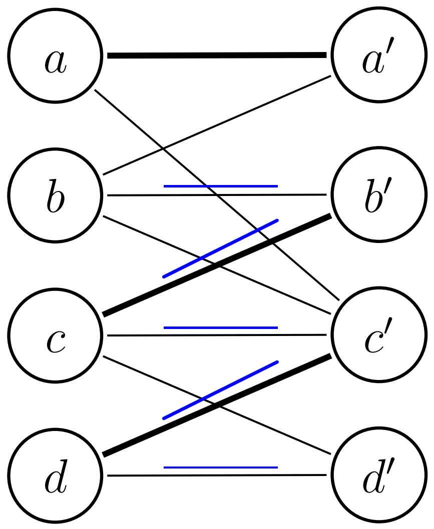
1. given a matching M, there is an **Algorithm** to produce a new matching with cardinality increased by 1;
2. if the algorithm fails then there exists a vertex cover having the same cardinality as M.

Algorithm: given a matching M,

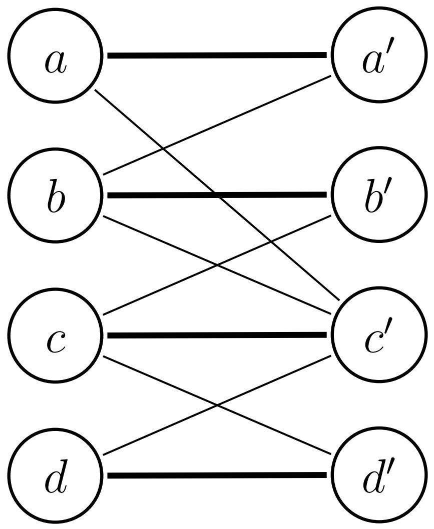
1. find an **augmenting path**, i.e., a path that
   - is formed by an odd number of edges, and is such that
   - all edges in odd position do not belong to M and
   - all edges in even position belong to M;
2. interchange matching and non-matching edges along the path
   $\Longrightarrow$ new matching with one more edge.

Symmetric difference of two sets A and B: $A \ominus B = (A \setminus B) \cup (B \setminus A)$.
Observation: augmenting a matching M by a path P is equivalent to execute $M \ominus P$.

Marriage: Let any young lady $i \in U' \subseteq U$ marry one of her friends in a matching M:
if M is not already a maximum cardinality matching
then there always exists a matching M' of larger cardinality where
all ladies $i \in U'$ remain married, but now possibly to other friends.



Matching



Augmented matching

Augmenting path (b, b', c, c', d, d'):

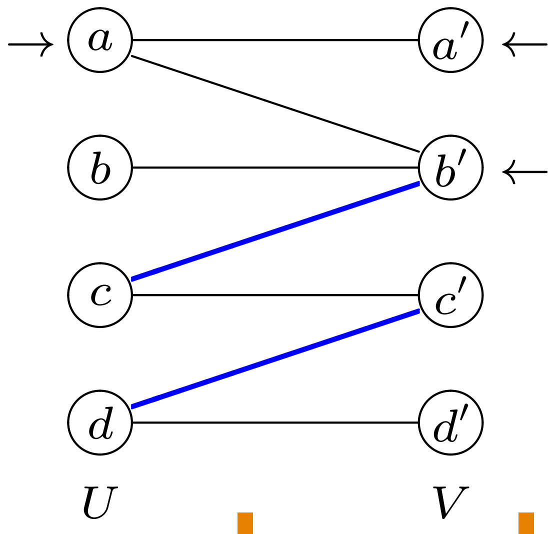
odd number of edges, odd edges $\notin$ M, even edges $\in$ M.

Basis of the Hungarian algorithm
Equivalent to Hall's Theorem.
Special case of Ford-Fulkerson's algorithm.

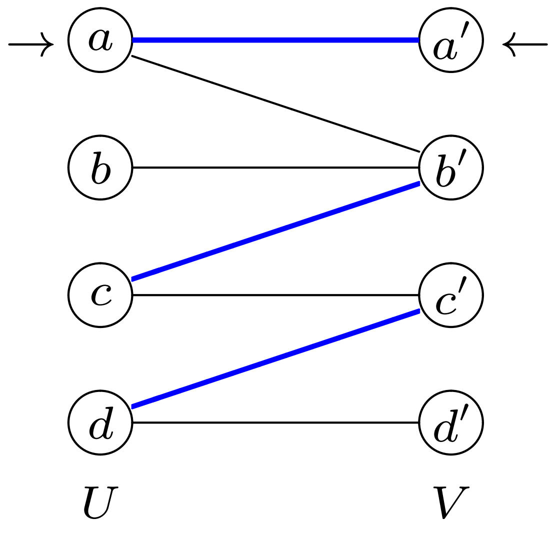
Maximum cardinality matching algorithm

- We start with an arbitrary matching M and iteratively augment it through augmenting paths;
- $L = \{$**unmatched** vertices $i \in U\}$ (**L**eft vertices);
- $R = \{$**labeled** vertices $j \in V\}$ (**R**ight vertices; initially $R := \emptyset$);
- the **label** of a vertex is its predecessor vertex in the current alternating path.
- We start from a left vertex $i \in L$;
- we label by i all the unlabeled right vertices j such that $[i, j] \in E$, and add them to R;
- if a labeled right vertex j is unmatched, we have found an augmenting path $P \Rightarrow$

  $\Rightarrow$ new matching $M \ominus P$ with one more edge; ...



Current matching

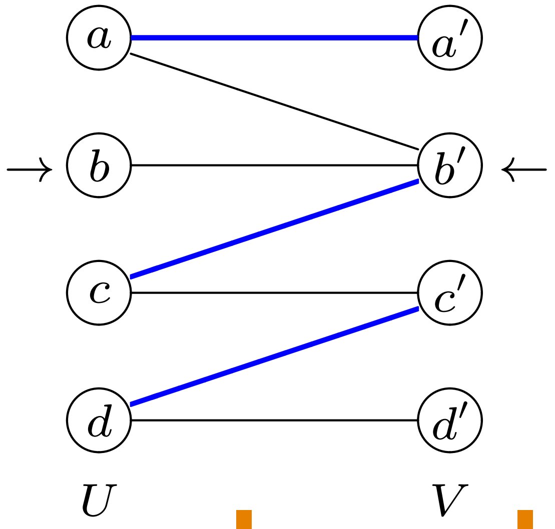


Augmented matching

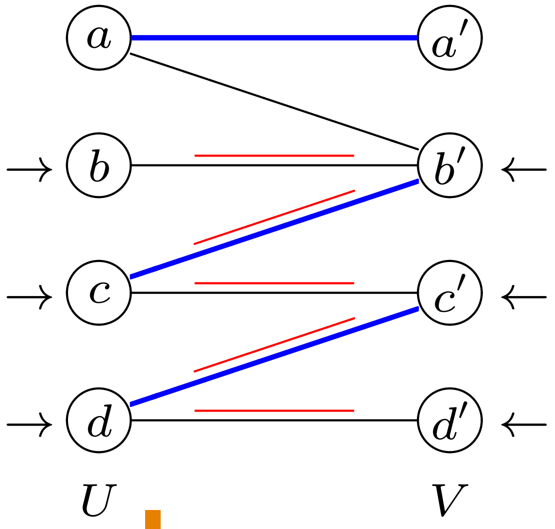
Maximum cardinality matching algorithm (cont'd)

- ... otherwise we remove a (matched) vertex j from R: j is matched by an edge $[\bar{\imath}, j]$, hence
  - vertex $\bar{\imath}$ is labeled by j and added to L,

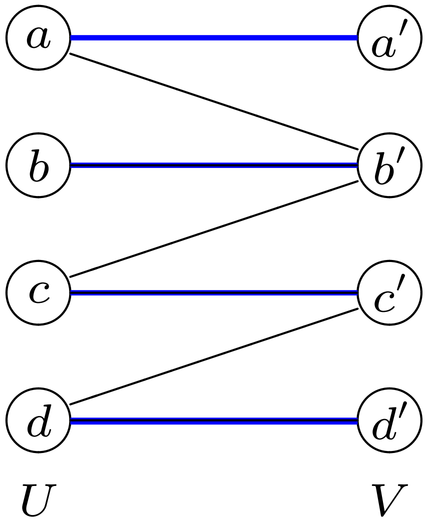
  and we try to continue the augmenting path from the newly labeled vertex $\bar{\imath}$;



Current matching



Augmenting path



Augmented matching

- in this way either we find an augmenting path or we conclude that no such path exists;
- in the latter case it can be proved that the matching has already maximum cardinality.

Algorithm Maximum_Matching:

```
$M$ := any matching in $G = (U, V; E)$ (possibly $M = \emptyset$);
$L$ := { all unmatched vertices of $U$}; $R$(labeled vertices of $V$):= $\emptyset$;
while $L \cup R \neq \emptyset$ do
    choose a vertex $x$ from $L \cup R$ giving priority to unmatched vertices of $R$;
    if $x \in L$ then Scan_leftvertex($x$) else Scan_rightvertex($x$)
endwhile
```

Procedure Scan_leftvertex(x)

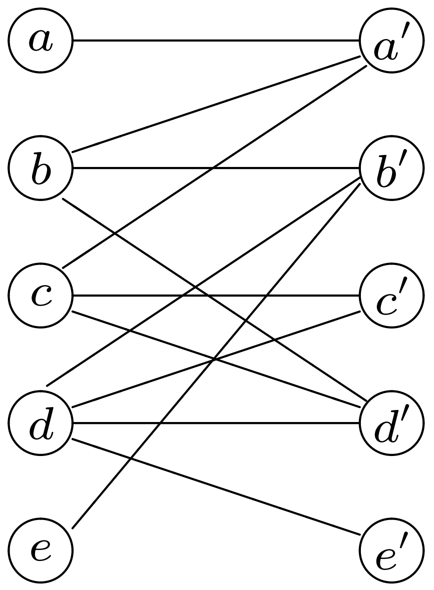
```
$L := L \setminus \{x\}$;
while there exists an edge $[x, j]$ with $j$ unlabeled do
    label $j$ as $l(j) := x$; $R := R \cup \{j\}$;
endwhile
```

Procedure Scan_rightvertex(x)

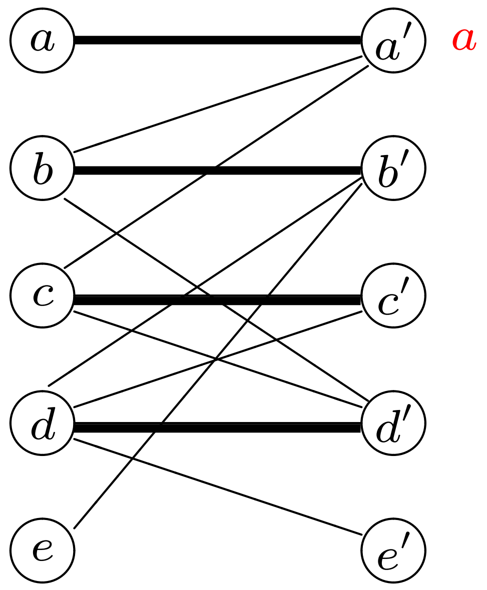
```
$R := R \setminus \{x\}$;
if there is a matching edge $[i, x]$ then label $i$ as $r(i) := x$ and set $L := L \cup \{i\}$
else [comment: augmentation of the matching]
    find the alternating path $P$ by backtracking the labels from $x$: $P := (\dots, r(l(x)), l(x), x)$;
    $M := M \ominus P$;
    let $L$ contain all unmatched vertices of $U$;
    $R := \emptyset$; cancel all labels
endif
```

Note: **Minimum vertex cover** = (unlabeled vertices of U) $\cup$ (labeled vertices of V).

Example



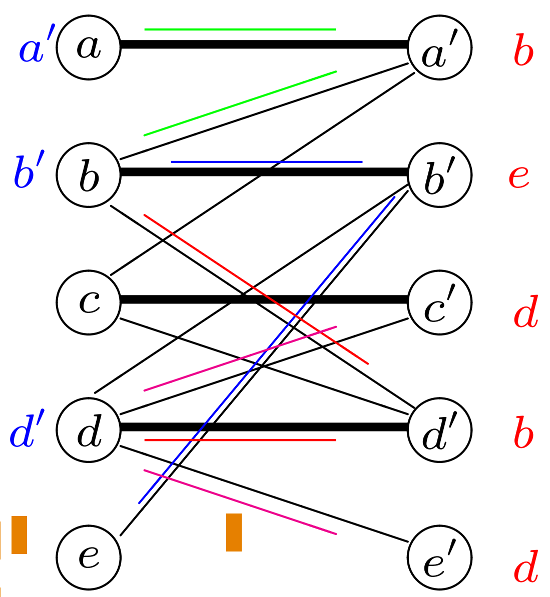
$\longrightarrow$



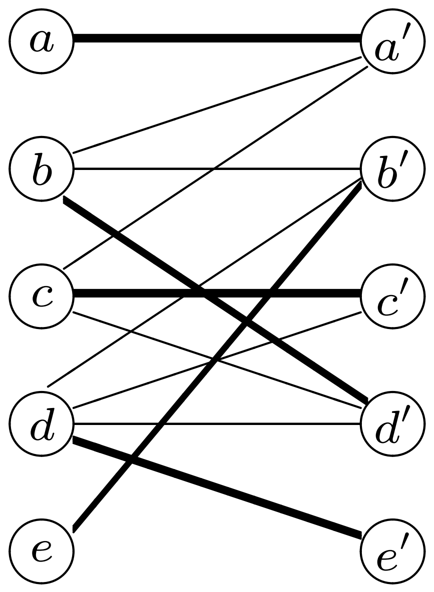
$L = \{a, b, c, d, e\}$, $R = \emptyset$;

1. $x = a$; Scan_leftvertex(a): $L = \{b, c, d, e\}$, $R = \{a'\}$;
2. $x = a'$; Scan_rightvertex(a'): $R = \emptyset$; **Augmentation** $P = (a, a')$;
3. Same for $(x = b,\ x = b')$;
4. , 5. Same for $(x = c,\ x = c')$ and $(x = d,\ x = d')$;

Example (cont'd)



$\longrightarrow$



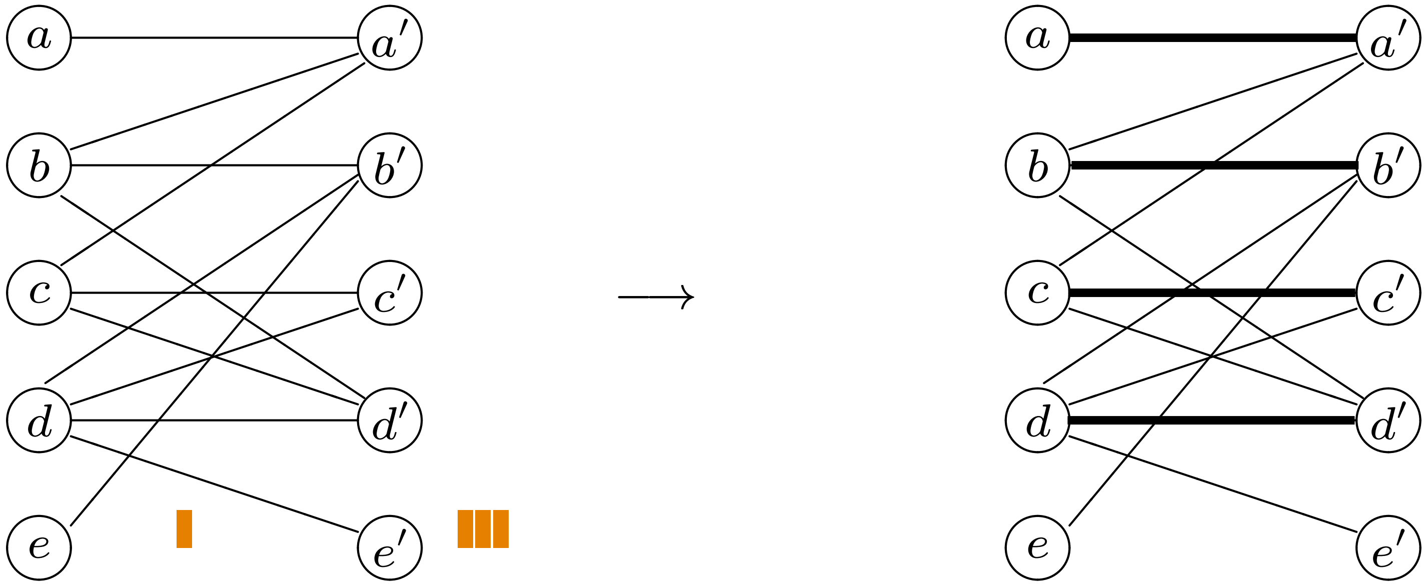
$L = \{e\}$, $R = \emptyset$;

6. $x = e$; Scan_leftvertex(e): $L = \emptyset$, $R = \{b'\}$;
   $x = b'$; Scan_rightvertex(b'): $R = \emptyset$, $L = \{b\}$;
   $x = b$; Scan_leftvertex(b): $L = \emptyset$, $R = \{a', d'\}$;
   $x = a'$; Scan_rightvertex(a'): $R = \{d'\}$, $L = \{a\}$;
   $x = d'$; Scan_rightvertex(d'): $R = \emptyset$, $L = \{a, d\}$;
   $x = a$; Scan_leftvertex(a): $L = \{d\}$, //;
   $x = d$; Scan_leftvertex(d): $L = \emptyset$; $R = \{c', e'\}$
   $x = e'$; Scan_rightvertex(e'): $R = \{c'\}$; **Augmentation** $P = (e, b', b, d', d, e')$.

Finding an initial matching

- One can start the algorithm with the empty matching, but

  in practice, it is quite simple to find a matching of large cardinality.
- **1. Straightforward Initialization Algorithm:**
  - scan the vertices i of U one by one:
  - match an edge $[i, j]$ if $j \in V$ is not already matched.
- Example:



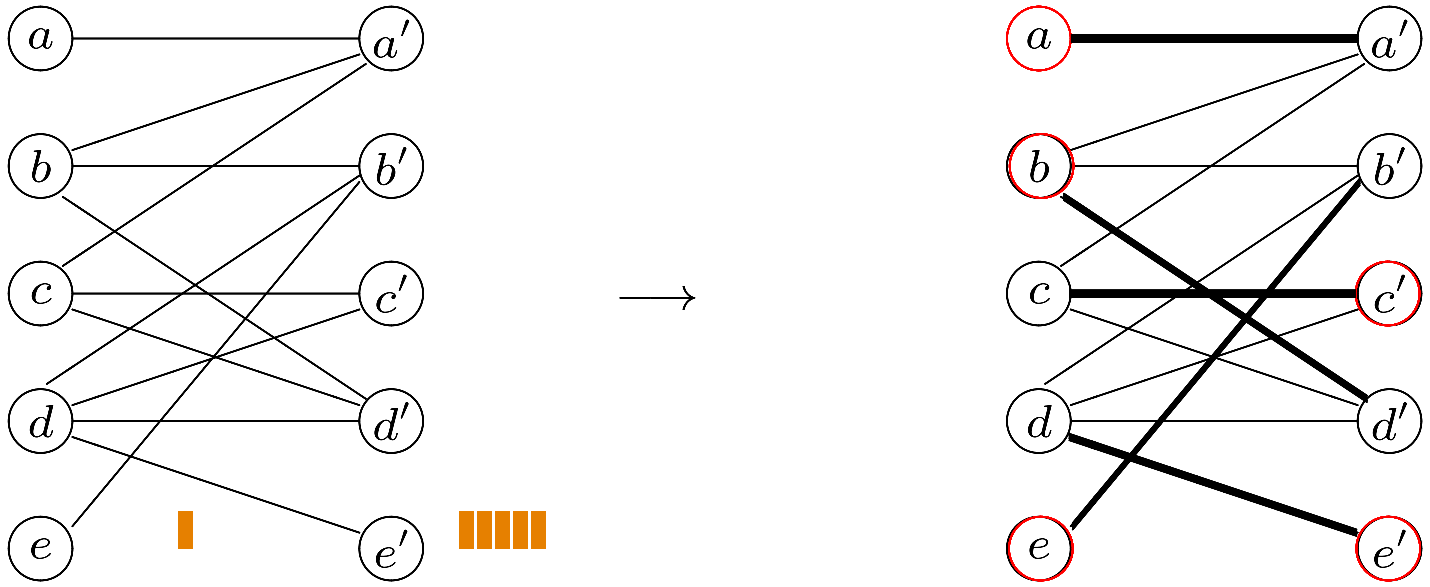
Finding an initial matching (cont'd)

- **2. Greedy Initialization Algorithm:**
  - scan the vertices of $U \cup V$ in increasing order of their degrees:
  - **if** the next unmatched vertex is in U (say, $i \in U$)

    then add to the matching an edge $[i, j]$ (if any) such that
    j is unmatched and has minimum degree.

    else (next unmatched vertex is $j \in V$) add to the matching an edge $[i, j]$ such that
    i is unmatched (if any) and has minimum degree.

- Example:



- Possible improvement: update the degrees at each iteration (but time increases):

  when $[i, j]$ is added to the matching remove all edges $[i, k]$ $(k \neq j)$ and $[\ell, j]$ $(\ell \neq i)$.

Complexity of maximum cardinality matching

Remind the basic concepts of **Computational complexity**:

- **Instance** of a problem P = specific input for a numeric case of the problem;
  (Problem P = (infinite) set of all its instances.)
- **Complexity of an algorithm** = measure of the **time** it takes, **in the worst case**, to solve an instance of P;
- **(Computational) complexity of a problem** P = complexity of the best algorithm for P.
- **Time:** number of elementary steps, or number of milliseconds on a specific computer, or ...
  Time as a **function of the instance size**.
- **Size of an instance =** number of bits needed to encode the input (**formal definition**)
  frequently (**and in our case**) equivalent to the number of values in the input.
- **Example (Straightforward Initialization Algorithm):**
  - scan the vertices i of U one by one:
  - match an edge $[i, j]$ if $j \in V$ is not already matched;
- **exactly** n (# vertices of U) iterations;
- at each iteration, **at most** n (# vertices of V) are examined
- the time taken by the algorithm is proportional **in the worst case**, to n^2, i.e.,
  - $\exists$ constant α such that (running time) $\leq \alpha n^2$;
  - The **time complexity** of the algorithm is $O(n^2)$.

Complexity of maximum cardinality matching (cont'd)

- **Algorithm Maximum_Matching:**
  - at every augmentation, one additional vertex of U is matched
    $\Longrightarrow$ at most n (# vertices) augmentations;
  - every vertex is labeled at most once per augmentation
    $\Longrightarrow$ each of its incident edges is scanned at most once;

    $\Longrightarrow$ finding an augmenting path requires a time at most proportional to m (# edges);

    in computational complexity terminology, finding an augmenting path requires $O(m)$ time;
  - hence the algorithm finds a maximum cardinality matching in $O(nm)$ time.

- **More efficient algorithms:**
  - **Hopcroft and Karp** (1973) augmented a matching not by a single augmenting path,

    but by a maximal system of vertex-disjoint augmenting paths of the same minimum length;

    – time complexity $O(m\sqrt{n})$, i.e., $O(n^{2.5})$ for "dense" graphs (graphs with $m \cong n^2$).
  - **Alt, Blum, Mehlhorn and Paul** (1991) used a fast adjacency matrix scanning technique;

    – time complexity $O(n^{1.5}\sqrt{m/\log n})$, where $n = |U| + |V|$,

    better for the case of dense graphs.

Another marriage problem

- Assume that every man and every lady ranks the members of the opposite sex in order of preference:
- **Stable Marriage (matching) Problem:**
  - assign each man to a lady so that not both of a pair of married people prefer to be married to someone with a higher rank than his/her actual partner;
  - in other words, if **(A, B)** and **(X, Y)** are married couples,
  - and **A** prefers **Y** to **B**,
  - then **Y** must prefer **X** to **A**
  - (otherwise **A** and **Y** could break off their marriages).
- It can be proved that a stable marriage exists for any choice of rankings.
- The problem can be solved in polynomial time.

Exercise 1

Starting with the given matching ($M = \{[c, c'], [d, d']\}$), find the maximum cardinality matching through algorithm Maximum_Matching. Use one graph per augmentation. Use the lines for the sequence of calls to procedures, and for values x, L, R and P.