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.

5. The Hungarian algorithm

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

Figure 3: Harold W. Kuhn (b. 1925), left ☺.

Hungarian algorithm: Theoretical bases

The Hungarian algorithm is recognized as a predecessor of the **primal-dual method** for linear programming, designed one year later by Dantzig, Ford and Fulkerson.

1. **Initialize** with any (u_i) and (v_j) satisfying $u_i + v_j \leq c_{ij}$ $(i, j = 1, \ldots, n)$;
2. find a maximum matching M (König) in the subgraph $\mathbf{G^0 = (U,V;E^0)}$ of $G = (U, V; E)$ that only contains the edges of E that satisfy $u_i + v_j = c_{ij}$ (i.e., such that $\bar{c}_{ij} = 0$);
3. **if** M is perfect **then** it has maximum weight $w(M) = \sum_{k=1}^{n}(u_k + v_k)$, hence **stop**;
4. **else** G^0 must contain (Hall) a subset $U' \subseteq U$ such that $|U'| > |F(U')|$: update the current covering system through (Egerváry)

$$\begin{cases} u_i & := & u_i + 1 \text{ for } i \in U'; \\ v_j & := & v_j - 1 \text{ for } j \in F(U'), \end{cases} \tag{12}$$

thus keeping $u_i + v_j = c_{ij}$, but increasing the value of $\sum_{k=1}^{n}(u_k + v_k)$ by $|U'| - |F(U')| > 0$ and **go to 2** (possibly new edges satisfy $u_i + v_j = c_{ij}$).

Pseudo-polynomial time complexity, but the two 1s in (12) can be replaced by

$$\min\{u_i + v_j - c_{ij} \ : \ i \in U', j \in F(U')\}$$

$\Longrightarrow$ **Polynomial time complexity**

Hungarian algorithm: Main structure

1. The algorithm starts with:

- **a feasible dual solution** u, v satisfying $u_i + v_j \leq c_{ij}$ $(i, j = 1, 2, \ldots, n)$;
- **a partial primal solution** (matching), in which less than n rows are assigned satisfying,
$$x_{ij}\,\overline{c}_{ij} = x_{ij}(c_{ij} - u_i - v_j) = 0 \quad (i, j = 1, 2, \ldots, n).$$

2. Each iteration

- tries to increase the cardinality of the current assignment by searching for an augmenting path in the subgraph **$G^0 = (U,V;E^0)$** of $G = (U, V; E)$ that only contains the **edges of E having zero reduced cost** $\overline{c}_{ij} = c_{ij} - u_i - v_j$;
- **if** the attempt is successful, we obtain a **new primal solution** with one more row assigned;
- **else** we update the **current dual solution** so that there are new edges with zero reduced cost.

3. Terminology:

- **partial assignment** = set of *assigned edges* and corresponding set of *assigned vertices*;
- **alternating path** = elementary path whose edges are alternately not assigned and assigned;
- **alternating tree rooted in k** = tree in which all paths emanating from k are alternating;
- **augmenting path** = alternating path whose initial and terminal edges (and, hence, vertices) are not assigned.

Finding an alternating tree rooted at an unassigned vertex $k \in U$

- A vertex is **labeled** if it belongs to a path emanating from k, or **unlabeled**;
- a **labeled** vertex of **V** is **scanned** if it has been used for extending the tree, or **unscanned**;
- a **labeled** vertex of **U** is also **scanned** (i.e., labeling and scanning coincide for **U**);
- **LV** $=$ labeled vertices of **V**; **SU, SV** $=$ scanned vertices of **U** and **V**.

The method is similar to Algorithm Maximum_Matching. **Two phases**:

- **Phase 1:** the unique candidate vertex $i \in U$ is labeled and scanned

  ($i = k$ at the 1st iteration);

  scanning $=$ extend the tree by assigning i as predecessor to all unlabeled vertices $j \in V$ for which $[i, j] \in E^0$, and label these vertices.
- **Phase 2:** a labeled unscanned vertex $j \in V$ is selected;

  if j is not in the current matching, we have an augmenting path from k to j;

  otherwise j is scanned by adding the unique edge $[row(j), j] \in E^0$ to the tree;

  vertex $row(j) \in U$ becomes then the candidate for the next Phase 1.
- **If**, in Phase 2, V contains no labeled unscanned vertex then the current tree cannot grow any longer.

Finding an alternating tree rooted at an unassigned vertex $k \in U$ (cont'd)

- A vertex is **labeled** if it belongs to a path emanating from k, or **unlabeled**;
- a **labeled** vertex of **V** is **scanned** if it has been used for extending the tree, or **unscanned**;
- a **labeled** vertex of **U** is also **scanned** (i.e., labeling and scanning coincide for **U**);
- **LV** = labeled vertices of **V**; **SU, SV** = scanned vertices of **U** and **V**.

Procedure Alternate(k)

$SU := LV := SV := \emptyset$; *fail* := `false`, $sink := 0$, $i := k$;
while (*fail* = `false` **and** $sink = 0$) **do**
  $SU := SU \cup \{i\}$ [**comment:** the candidate vertex $i \in U$ is labeled and scanned];
  comment: i is assigned as predecessor to all unlabeled vertices $j \in V$ for which $[i, j] \in E^0$;
  for each $j \in V \setminus LV$ st $c_{ij} - u_i - v_j = 0$ **do** $pred_j := i$, $LV := LV \cup \{j\}$;
  if $LV \setminus SV = \emptyset$ **then** *fail* := `true`
  else
    let j be any vertex in $LV \setminus SV$;
    $SV := SV \cup \{j\}$ [**comment:** a labeled unscanned vertex $j \in V$ is selected and scanned];
    if $row(j) = 0$ **then** $sink := j$ [**comment:** j is unassigned: augmenting path from k to j]
    else $i := row(j)$ [**comment:** unique edge $[row(j), j] \in E^0$]
  endif
endwhile
return $sink$

Complexity

- Each iteration of the main loop requires $O(n)$ time.
- at each iteration, a different vertex $j \in LV \setminus SV$ is selected and added to SV,

  $\rightarrow$ so a different vertex $i = row(j)$ is considered;

  $\rightarrow$ the time complexity of Alternate(k) is $O(n^2)$ (as an iteration of Maximum_Matching).

Example (resumed)

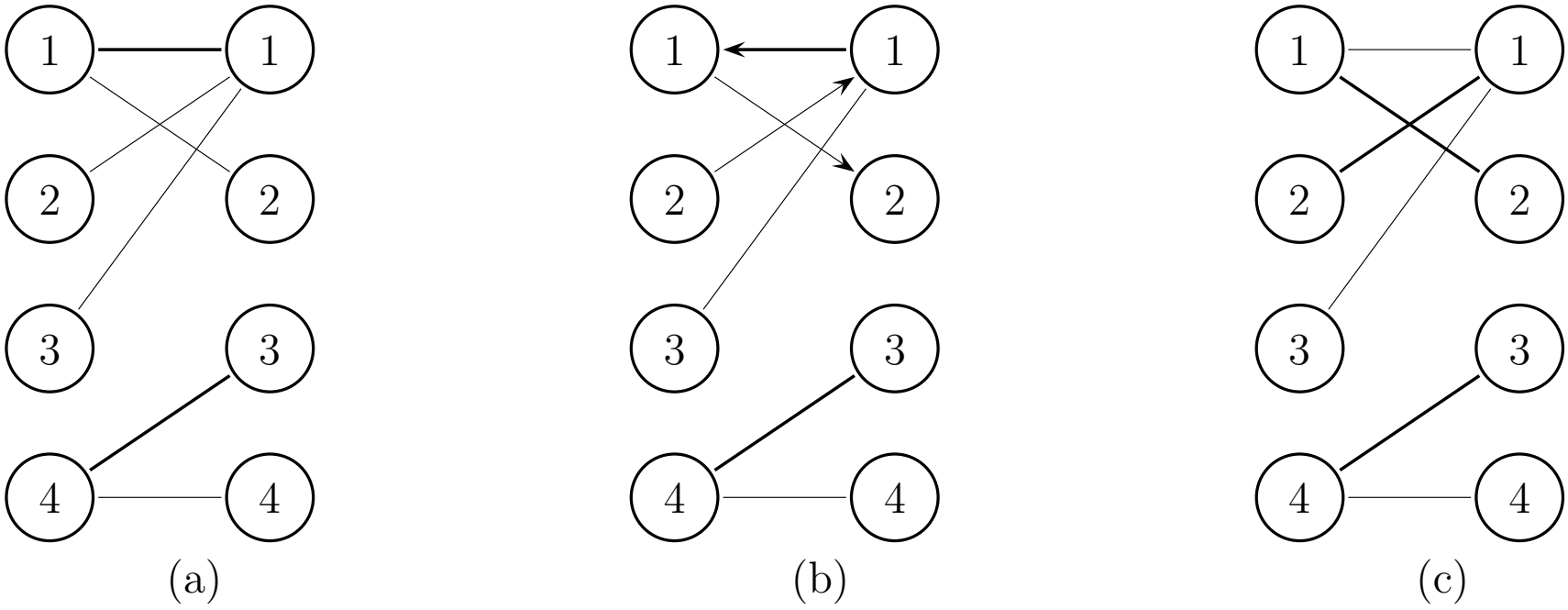
- $$\begin{array}{c} \\ 7 \\ 2 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 2 & 0 & 0 \end{array} \\ \begin{pmatrix} 7 & 9 & 8 & 9 \\ 2 & 8 & 5 & 7 \\ 1 & 6 & 6 & 9 \\ 3 & 6 & 2 & 2 \end{pmatrix} \\ C \end{array} \quad \begin{array}{c} \\ \begin{pmatrix} \underline{0} & 0 & 1 & 2 \\ 0 & 4 & 3 & 5 \\ 0 & 3 & 5 & 8 \\ 1 & 2 & \underline{0} & 0 \end{pmatrix} \\ \overline{C} \end{array}$$ **$\underline{0}$ = initial solution**
- $row = (1,\ 0,\ 4,\ 0)$ ($\Leftrightarrow \varphi = (1,\ 0,\ 0,\ 3)$)
- Assume $k = 2$: $SU = LV = SV = \emptyset$, *fail* = `false`, $sink = 0$;

  $i = 2$: $SU = \{2\}$, $pred_1 = 2$, $LV = \{1\}$;

  $j = 1$: $SV = \{1\}$;

  $i = 1$: $SU = \{2, 1\}$, $pred_2 = 1$, $LV = \{1, 2\}$;

  $j = 2$: $SV = \{1, 2\}$, $sink = 2 \rightarrow$ **return**.



(a) graph G^0; (b) alternating tree; (c) new graph G^0 (by interchanging the edges)

Hungarian algorithm

- (u, v) = current feasible dual solution; row, φ = current assignment;
- $\overline{U}$ $(\subseteq U)$ = set of **assigned vertices** of U.

Algorithm Hungarian

initialize u, v, row, φ and $\overline{U}$ [**comment:** either to zero, or through preprocessing];
while $|\overline{U}| < n$ **do** [**comment:** select an unassigned vertex of U and assign it]
  let k be any vertex in $U \setminus \overline{U}$;
  while $k \notin \overline{U}$ **do**
    $sink := \text{Alternate}(k)$;
    if $sink > 0$ **then** [**comment:** augmenting path found: assign k]
      $\overline{U} := \overline{U} \cup \{k\}$, $j := sink$;
      repeat
        $i := pred_j$, $row(j) := i$, $h := \varphi(i)$, $\varphi(i) := j$, $j := h$
      until $i = k$
    else [**comment:** no augmenting path found: update the dual solution]
      $\delta := \min\{c_{ij} - u_i - v_j (= \overline{c}_{ij})$ st $i \in SU, j \in V \setminus LV\}$;
      for each $i \in SU$ **do** $u_i := u_i + \delta$; **for each** $j \in LV$ **do** $v_j := v_j - \delta$
      comment: it can be proved that Alternate(k) will now produce an enlarged tree.
    endif
  endwhile
endwhile

Complexity

- Initialization (procedure **Basic_preprocessing**): $O(n^2)$ time;
- the **outer while loop** is executed $O(n)$ times. At each iteration
  the **inner while loop** executes **Alternate(k)** and **dual updating** $O(n)$ times;
  Alternate(k) takes $O(n^2)$ time; computing δ takes $O(n^2)$ time;
- overall time complexity $O(n^4)$.

Example (resumed)

- **Basic_preprocessing**: $row = (1,\ 0,\ 4,\ 0)$, $\varphi = (1,\ 0,\ 0,\ 3)$;
- 1st iteration: **Alternate(2)**: $sink = 2$ and $pred = (2, 1, -, -)$ $\Rightarrow$
  $\overline{U} = \{1, 4, 2\}$, $row = (2, 1, 4, 0)$, $\varphi = (2, 1, 0, 3)$;
- 2nd iteration: **Alternate(3)**: $SU = LV = SV = \emptyset$, *fail* = `false`, $sink = 0$;

  $i = 3$: $SU = \{3\}$, $pred_1 = 3$, $LV = \{1\}$;
  $j = 1$: $SV = \{1\}$;

  $i = 2$: $SU = \{3, 2\}$, $fail =$ `true` $\rightarrow$ **return**.

- $\overline{C} = \begin{pmatrix} 0 & \underline{\mathbf{0}} & 1 & 2 \\ \underline{\mathbf{0}} & 4 & 3 & 5 \\ 0 & 3 & 5 & 8 \\ 1 & 2 & \underline{\mathbf{0}} & 0 \end{pmatrix}$, $u = (7, 2, 1, 2)$, $v = (0, 2, 0, 0)$;

  $SU = \{3, 2\}$, $V \setminus LV = \{2, 3, 4\}$; the two ladies in SU have only one friend, i.e., 1.

We modify the dual variables so that new men of $V \setminus LV$ become their friends:

- $\delta = 3$, $u = (7, 5, 4, 2)$, $v = (-3, 2, 0, 0)$;
- $\overline{C} = \begin{pmatrix} 3 & \underline{\mathbf{0}} & 1 & 2 \\ \underline{\mathbf{0}} & 1 & 0 & 2 \\ 0 & 0 & 2 & 5 \\ 4 & 2 & \underline{\mathbf{0}} & 0 \end{pmatrix}$.

- 3rd iteration: second execution of **Alternate(3)** left as an exercise (see figure);
- new primal solution: $\overline{U} = \{1, 4, 2, 3\}$, $row = (3, 1, 2, 4)$, $\varphi = (2, 3, 1, 4)$;
- optimal solution:

$$x = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

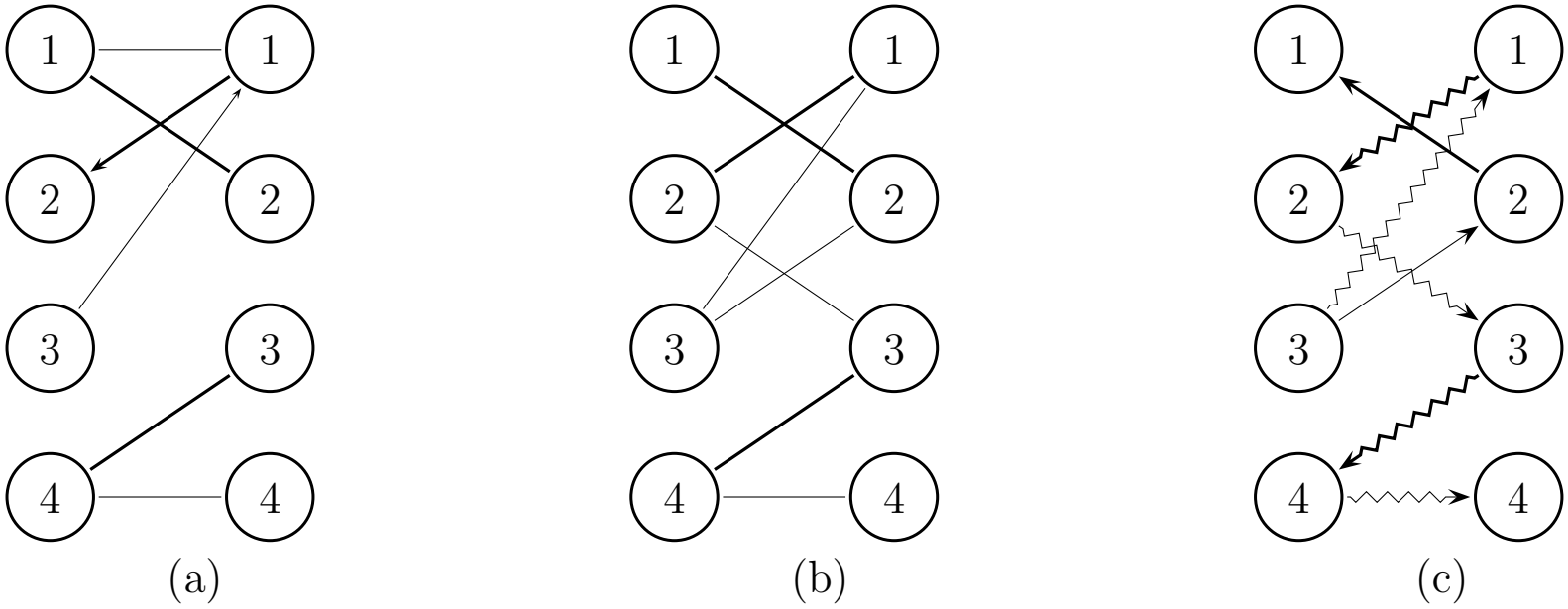


Figure 1: (a) 1st execution of Alternate(3): no augmenting path found; (b) New graph G^0 after dual updating; (c) 2nd execution of Alternate(3): augmenting path (zig-zag lines).

Exercise 4

Consider the assignment problem defined by cost matrix C:

$$\begin{array}{c} \\ -- \\ -- \\ -- \\ -- \end{array} \begin{array}{c} \begin{array}{cccc} -- & -- & -- & -- \end{array} \\ \begin{pmatrix} 5 & 9 & 8 & 5 \\ 1 & 6 & 6 & 6 \\ 2 & 4 & 5 & 4 \\ 3 & 5 & 3 & 2 \end{pmatrix} \end{array} \qquad C$$

$$\begin{pmatrix} -- & -- & -- & -- \\ -- & -- & -- & -- \\ -- & -- & -- & -- \\ -- & -- & -- & -- \end{pmatrix} \qquad \overline{C}$$

U	V
①	①
②	②
③	③
④	④

1. Apply procedure Basic_preprocessing:
   - give the values of (u_i) and (v_j), respectively besides column 1 and above row 1 of C;
   - define the reduced costs of matrix $\overline{C}$; give the values of φ and row:

     $(\varphi) = (\quad , \quad , \quad , \quad)$; $(row) = (\quad , \quad , \quad , \quad)$;
   - highlight, in matrix $\overline{C}$, the values corresponding to the current solution.
2. Draw in the graph the edges of the bipartite subgraph G^0 containing only the edges with zero reduced cost, and highlight the current solution.
3. Apply the Hungarian algorithm, and highlight the edges of the augmenting path.
4. Give the final values of φ and row:

   $(\varphi) = (\quad , \quad , \quad , \quad)$; $(row) = (\quad , \quad , \quad , \quad)$.

Conclusion?

Who invented the Hungarian algorithm?

- König (**1916**) ?
- Egerváry (**1931**) ?
- Kuhn (**1955**) ?
- **None of them ?**
- A recent historical discovery: A posthumous paper written, prior to **1851 (!!!)** by **one of the greatest mathematicians of all time:**

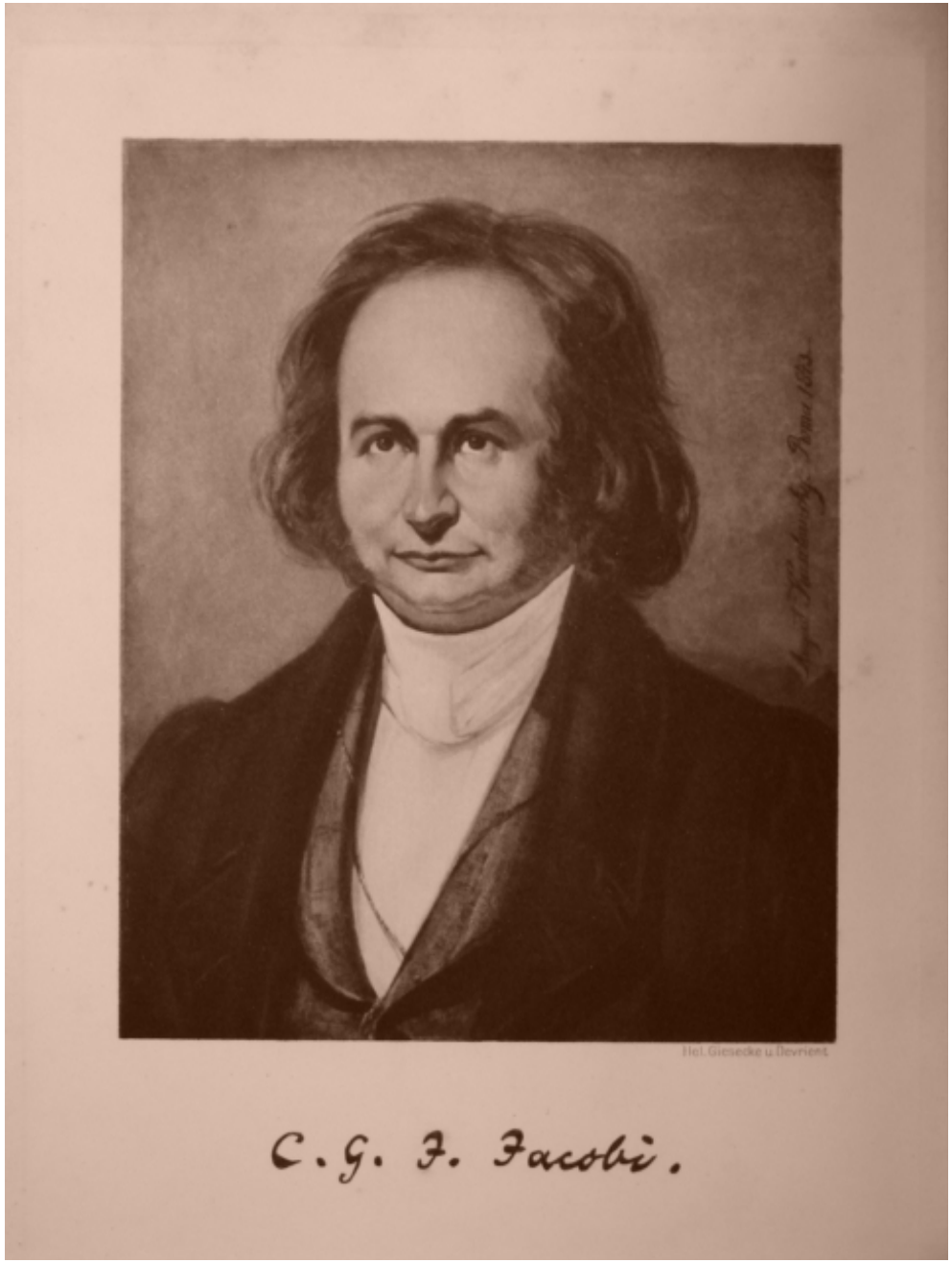


Figure 4: Karl Gustav Jacob Jacobi (1804 - 1851)

Who was Jacobi?

Carl Gustav Jacob Jacobi was born in Potsdam in 1804 to a Jewish family. He was tutored by an uncle and entered the Gymnasium at 12, directly in the highest class in spite of his youth. However, the law did not allow him to enter University until he was 16.

Before he was 20, he passed his preliminary examination for Oberlehrer, which gave him permission to teach not only mathematics but also Greek and Latin to high school students. In spite of being a Jew, he was offered a position at a prestigious Gymnasium for the following summer. He had already submitted his Ph. D. thesis to the University of Berlin.

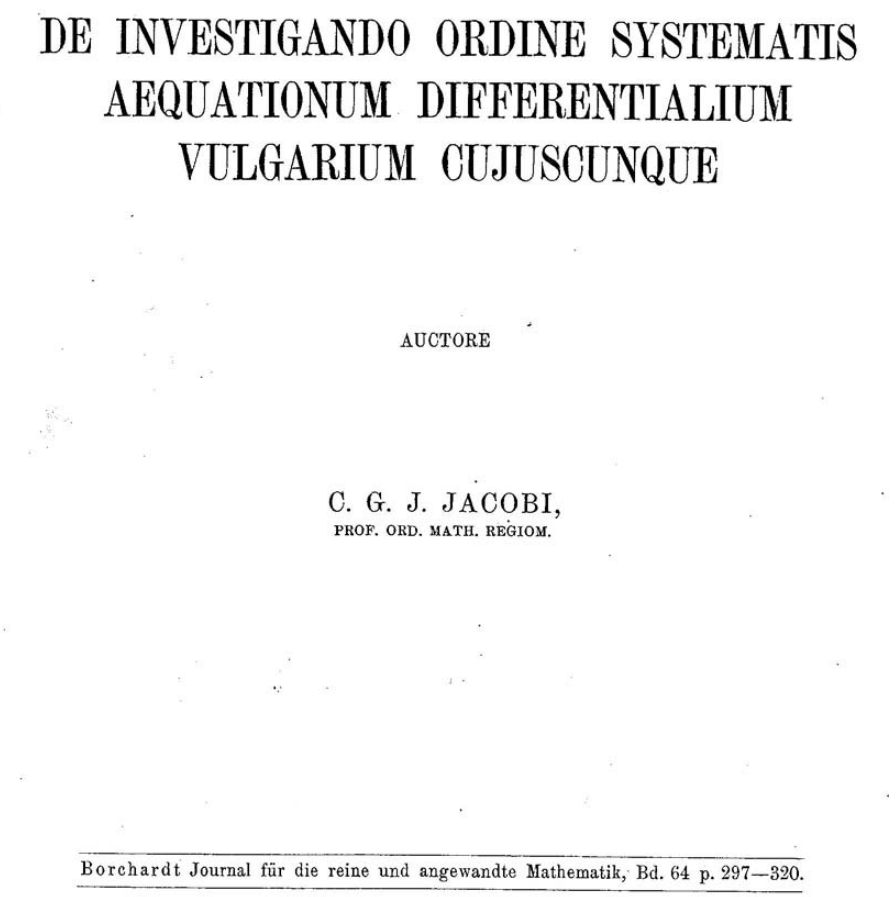
At this point, Jacobi became a Christian. Thus, he was able to begin a university career at the University of Berlin at the age of twenty. Although his lectures were a success, there was no prospect of a good position in Berlin, so in 1926 he moved to the University of Königsberg.

He is universally famous for the concepts of Jacobian matrix and for his theory of elliptic functions, but he was proficient in many fields of mathematics and published prolifically.

In 1843, Jacobi suffered a breakdown from overwork, and made a trip to Italy in the hope of regaining his health. Returning to Prussia, he lectured occasionally at the University of Berlin, supported by an allowance provided by the King of Prussia for over twenty years.

In 1847 he entered politics, and signed a petition in 1848 asking for the King to turn over his power to the Parliament. The King threatened to cut off his allowance but, when Jacobi received an offer from Austria, the King relented and Jacobi stayed on in Berlin.

In 1851 he contracted influenza, developed small pox and died within a week. He was 46 years old.



DE INVESTIGANDO ORDINE SYSTEMATIS
AEQUATIONUM DIFFERENTIALIUM
VULGARIUM CUJUSCUNQUE

AUCTORE

C. G. J. JACOBI,
PROF. ORD. MATH. REGIOM.

Borchardt Journal für die reine und angewandte Mathematik, Bd. 64 p. 297—320.

Figure 5: About the research of the order of a system of arbitrary ordinary differential equations (from the posthumous manuscripts)

2.

De solutione problematis inaequalitatum, quo investigatio ordinis systematis aequationum differentialium quarumcunque innititur. Proposito schemate, definitur Canon. Dato Canone quocunque, invenitur simplicissimus.

Antecedentibus investigatio ordinis systematis aequationum differentialium vulgarium revocata est ad sequens problema inaequalitatum etiam per se tractatu dignum:

Problema.

Disponantur nn quantitates $h_k^{(i)}$ *quaecunque in schema Quadrati, ita ut habeantur n series horizontales et n series verticales, quarum quaeque est n terminorum. Ex illis quantitatibus eligantur n transversales, i. e. in seriebus horizontalibus simul atque verticalibus diversis positae, quod fieri potest* $1.2 \ldots n$ *modis; ex omnibus illis modis quaerendus est is, qui summam n numerorum electorum suppeditet maximam.*

Dispositis quantitatibus $h_k^{(i)}$ in figuram quadraticam

$$\begin{array}{cccc} h_1' & h_2' & \ldots & h_n' \\ h_1'' & h_2'' & \ldots & h_n'' \\ \cdot & \cdot & \cdot & \cdot \\ h_1^{(n)} & h_2^{(n)} & \ldots & h_n^{(n)}, \end{array}$$

earum systema appellabo *schema propositum;* omne schema, inde ortum addendo singulis ejusdem seriei horizontalis terminis eandem quantitatem, appellabo *schema derivatum.* Sit

Figure 6: Note: Jacobi did not even have proper terminology for a matrix!!! The term "Matrix" was invented in same years by James Joseph Sylvester (next slide).

This homaloidal law has not been stated in the above commentary in its form of greatest generality. For this purpose we must commence, not with a square, but with an oblong arrangement of terms consisting, suppose, of m lines and n columns. This will not in itself represent a determinant, but is, as it were, a Matrix out of which we may form various systems of determinants by fixing upon a number p, and selecting at will p lines and p columns, the squares corresponding to which may be termed determinants of the pth order. We have, then, the following proposition. The number of uncoevanescent determinants constituting a system of the pth order derived from a given matrix, n terms broad and m terms deep, may equal, but can never exceed the number

$$(n-p+1)(m-p+1).$$

Canon derivatus I.

	I	II	III	IV	V	VI	VII	l
I	11*	11	8	19	18	10	5	4
II	10	15*	14	13	18	21	17	7
III	8	13	17*	18	17	25	12	2
IV	4	11	14	25*	20	21	27	0
V	9	6	12	14	27*	22	34	4
VI	6	13	8	14	11	25*	22	5
VII	11	12	8	22	24	21	40*	0

Canon derivatus II.

	I	II	III	IV	V	VI	VII	l
I	11*	11	8	19	18	10	5	4
II	8	13*	12	11	16	19	15	5
III	6	11	15*	16	15	23	10	0
IV	4	11	14	25*	20	21	27	0
V	7	4	10	12	25*	20	32	2
VI	4	11	6	12	9	23*	20	3
VII	11	12	8	22	24	21	40*	0

Canon simplicissimus.

	I	II	III	IV	V	VI	VII	l
I	11*	11	8	19	18	10	5	4
II	7	12*	11	10	15	18	14	4
III	6	11	15*	16	15	23	10	0
IV	4	11	14	25*	20	21	27	0
V	6	3	9	11	24*	19	31	1
VI	4	11	6	12	9	23*	20	3
VII	11	12	8	22	24	21	40*	0

E schemate proposito, addendo terminis serierum diversarum respective

Figure 7: The Jacobi method exactly replicates the patterns obtained by the Hungarian Method!

- Jacobi invented the Hungarian method (primal-dual) without knowing the duality of linear programming!

- **Stigler's law of eponymy (Stigler, 1980)**:

  No scientific discovery is named after its original discoverer.

- Scientific observations and results are often associated with people who have high visibility and social status, and are named long after their discovery:

  - the Pythagorean theorem was discovered by Babylonian mathematicians;
  - the Gaussian distribution (Gauss, 1794) was introduced by Abraham de Moivre in 1733;
  - the Halley's comet was observed by astronomers since 240 BC;
  - ... **AND**
  - Stigler attributes the discovery of Stigler's Law to Robert K. Merton. ☺

6. Non-Hungarian algorithms

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

$\mathbf{O(n^3)}$ Hungarian algorithms

- Other $O(n^4)$ algorithms: Silver (1960), Iri (1960), Desler and Hakimi (1969).

- **1971** Tomizawa and, independently, Edmonds and Karp (non-Hungarian algorithm): shortest path computations on the reduced costs produce an $\mathbf{O(n^3)}$ time algorithm.

- Best Hungarian algorithm: Lawler **(1976)**:
  - when the procedure Alternate(k) is unsuccessful, and the dual update is performed, the current alternating tree remains valid;
  - $\mathbf{O(n^2)}$ procedure that iterates tree extensions and dual updates until a new vertex is assigned;
  - **time complexity $\mathbf{O(n^3)}$.**

- Successful implementations of $O(n^3)$ **Hungarian algorithms + shortest path techniques**: Burkard and Derigs (1980), Jonker and Volgenant (1987), Carpaneto, Martello and Toth (1988)

- However **the first $\mathbf{O(n^3)}$ algorithm** was given in 1969 by Dinic and Kronrod in an obscure paper on *Doklady Academy Naukk SSSR* (**Dual algorithm**):
  the algorithm is based on algebraic considerations, and is **totally independent** of linear programming duality theory.

Dokl. Akad. Nauk SSSR
Tom 189 (1969), No. 1

Soviet Math. Dokl.
Vol.10 (1969), No. 6

AN ALGORITHM FOR THE SOLUTION OF THE ASSIGNEMT PROBLEM

UDC 518.5

E. A. DINIC AND M. A. KRONROD

Let (a_{ij}) be a square matrix of order n. A solution of the assignment problem for this matrix is a set of n elements of the matrix, one in each row and each column, such that the sum of these elements is minimal with respect to all such sets. Algorithms are known which solve the assignment problem after Cn^4 operations,* for example the Bradford method.

In the present article an algorithm is constructed which solves the problem more rapidly: one modification after $Cn^3 \log n$ operations, the other after Cn^3 operations.

Definition. Let some vector $\Delta = (\Delta_1, \Delta_2, \cdots, \Delta_n)$ be given. An element a_{ij} is called Δ-*minimal*, if $a_{ji} - \Delta_j < a_{ik} - \Delta_k$ for all k.

Lemma. *For any Δ let there be given a set of n Δ-minimal elements $a_{1j(1)}, a_{2j(2)}, \cdots, a_{nj(n)}$, one from each row and column. Then this set is a solution of the assignment problem.*

Proof.

$$\sum_{i=1}^{n} a_{ij(i)} = \sum_{k=1}^{n} \Delta_k + \sum_{i=1}^{n} (a_{ij(i)} - \Delta_{j(i)}).$$

The first sum in the right member is constant for all sets, and the second is minimal. □

For any vector Δ let us select in each row one of the Δ-minimal elements (we shall call these selected elements basics). The number of free columns (columns without basics) is called the deficiency of the set of basics. From the lemma it follows that the set of basics with deficiency zero is a solution of the problem.

An algorithm is derived below permitting iteration: for a given vector Δ and a set of basics with nonzero deficiency, find a new vector and a set of basics possessing a lesser deficiency.

Taking the zero vector as Δ and some set of basics for it, we can solve the problem by successive iteration.

Iteration. For any vector Δ let us have a set of basics $a_{1j(1)}, a_{2j(2)}, \cdots, a_{nj(n)}$ with deficiency $m \neq 0$. We single out some free column. Let its index be s_1. We increase Δ_{s_1} to the maximum δ such that all basics remain Δ-minimal elements (a method of finding δ is indicated below). We obtain that for some i $a_{ij(i)} - \Delta_{j(i)} = a_{is_1} - \Delta_{s_1}$, i.e. the element a_{is_1} remains Δ-minimal. We call it an alternative basic and single out a column with index $s_2 = j(i)$, containing a basic of this row (we now have two marked columns). We increase Δ_{s_1} and Δ_{s_1} to the maximal δ such that all basics remain Δ-minimal, we find a new alternative basic in one of the columns and single out a new column, and so on until we single out a column with two or more basics.

Now we shall construct a new set of basics. We observe that in each row there is not more than one alternative basic.

* Contemplated is a number of operations realizable on a computer under a program realizing the algorithm. C is a multiplier not depending on n.

Dinic and Kronrod (1969)

- Approach **totally independent** of linear programming duality theory.
- **Theorem** Given n values Δ_j $(j = 1, 2, \ldots, n)$, let an element c_{ij} be called Δ-*minimal* if

$$c_{ij} - \Delta_j \leq c_{ik} - \Delta_k \text{ for all } k:$$

$$\begin{array}{cc} \Delta_j & \begin{array}{cccc} 0 & 4 & 3 & 3 \end{array} \\ & \begin{pmatrix} 7 & 9 & 8 & 9 \\ 2 & 8 & 5 & 7 \\ 1 & 6 & 6 & 9 \\ 3 & 6 & 2 & 2 \end{pmatrix} \end{array} \qquad \begin{array}{cc} \Delta_j & \begin{array}{cccc} 0 & 4 & 3 & 3 \end{array} \\ & \begin{pmatrix} 7 & 5 & 5 & 6 \\ 2 & 4 & 2 & 4 \\ 1 & 2 & 3 & 6 \\ 3 & 2 & -1 & -1 \end{pmatrix} \end{array}$$

c_{ij} $\qquad\qquad$ $c_{ij} - \Delta_j$

Then a set of n Δ-minimal elements $c_{i\varphi(i)}$ $(i = 1, 2, \ldots, n)$ such that $\varphi(i) \neq \varphi(k)$ if $i \neq k$ is an optimal solution to AP.

- **Proof** Solution value: $\displaystyle\sum_{i=1}^{n} c_{i\varphi(i)} = \sum_{j=1}^{n} \Delta_j + \sum_{i=1}^{n} (c_{i\varphi(i)} - \Delta_{\varphi(i)})$.

  The first sum in the right hand side is a constant. The second sum is minimal by definition. $\triangle$

- **Alternative proof** $v_j = \Delta_j$ $(j = 1, 2, \ldots, n)$ and $u_i = c_{i\varphi(i)} - \Delta_{\varphi(i)}$ $(i = 1, 2, \ldots, n)$ are a feasible dual solution satisfying complementary slackness. $\triangle$

Dinic–Kronrod $O(n^3)$ Algorithm

1. Start with a (possibly zero) vector Δ, and select a Δ-minimal element in each row;
2. **if** all the selected elements are in different columns, **then** the solution is optimal
3. **else ($\Leftrightarrow \exists$ unassigned columns and multiply assigned columns)**
   - select an unassigned column s and increase Δ_s by a quantity δ such that:
     - all currently selected elements remain Δ-minimal;
     - there exists a row i for which $c_{i\varphi(i)} - \Delta_{\varphi(i)} = c_{is} - (\Delta_s + \delta)$ ($\Leftrightarrow$ we have an alternative element selection for row i);

$$\begin{array}{c} \\ i \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} \varphi(i) & & s & \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 \end{pmatrix} \end{array} \longrightarrow \begin{array}{c} \\ i \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} & & \varphi(i) & \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 \end{pmatrix} \end{array}$$

   - **if** column $\varphi(i)$ has more than one row assigned **then** set $\varphi(i) := s$ and **go to 2**
   - **else** increase Δ_s and $\Delta_{\varphi(i)}$ s.t. there is a new row with alternative assignment and iterate.

The Hungarian algorithm (**primal-dual**) operates on a partial feasible assignment:
iteration $\Rightarrow$ one additional assignment.

The Dinic-Kronrod algorithm (**dual**) operates on a complete but unfeasible assignment:
iteration $\Rightarrow$ one more column is assigned, a multiply assigned column has one assignment less.

Other dual (non-simplex) algorithms

- **Dual Algorithm** = maintains an optimal ("more than" optimal) but unfeasible solution, and strives to reach feasibility.
- For the AP, strongly related to the Dinic-Kronrod algorithm.
- Hung and Rom (1980):
  - series of relaxed problems where one set of constraints ($\sum_{i=1}^{n} x_{ij} = 1$) is disregarded;
  - solved by updating the current solution through shortest paths, until it becomes feasible;
  - time complexity $O(n^3)$.
- Bertsekas (1981):
  - *Auction algorithm*: pseudo-polynomial time $O(n^3 + n^2\mathcal{C})$ ($\mathcal{C} = \max\{c_{ij}\}$), but
  - very efficient in practice.
- Polynomial-time auction algorithms:
  - Bertsekas and Eckstein (1988);
  - Orlin and Ahuja (1992): auction + scaling technique, $O(\sqrt{n}\ m \log(n\mathcal{C}))$.
- Other $O(\sqrt{n}\ m \log(n\mathcal{C}))$ scaling algorithms:
  - Gabow and Tarjan (1989);
  - Goldberg and Kennedy (1995), very efficient in practice.

Primal non-simplex algorithms

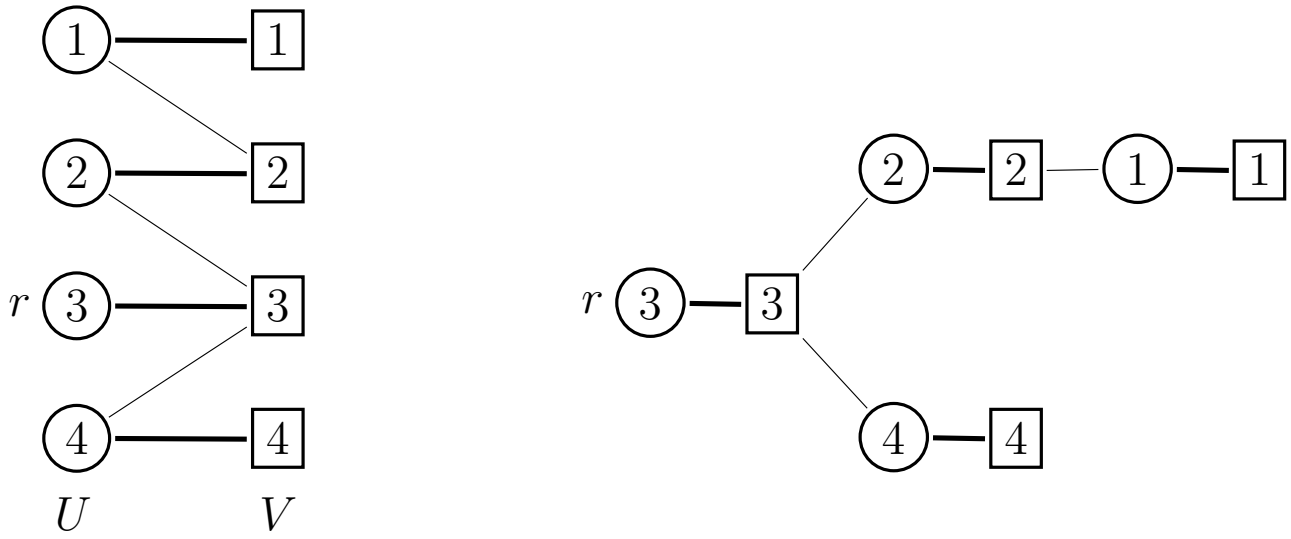
- **Primal Algorithm** = maintains a feasible but not optimal solution, and strives to reach optimality.
- For the AP, older than the dual algorithms, and much less efficient in practice.
- Balinski and Gomory (1964):
  - primal feasible solution and an unfeasible dual solution satisfying complementary slackness;
  - improved through alternating trees;
  - time complexity $O(n^4)$.
- Klein(1967), Cunningham and Marsh (1978):
  - *negative cycle canceling techniques*;
  - time complexity $O(n^3)$.
- Srinivasan and Thompson (1977), Akgül (1992)
  - operator theory of parametric programming;
  - can be implemented in time complexity $O(n^3)$.
- **Primal-Dual algorithm** = dual algorithm that, at each iteration, improves the feasibility of the current solution through the solution of a *restricted primal problem* (independent of the costs).

Primal simplex algorithms

- Derived from methods for the Hitchcock-Koopmans (1941) Transportation Problem;
- starting point: Dantzig (1963) specialization of the simplex algorithm to network problems;
- limited practical efficiency, but interesting theoretical aspects.

- Main result: there is a one-to-one correspondence between
  - primal (integer) basic solutions of the linear problem associated with AP and
  - spanning trees on the associated bipartite graph.

- Main algorithm (Cunningham (1976) and Barr, Glover and Klingman (1977), independently):
  - special subsets of bases/trees: *strongly feasible trees*, aka *alternating path bases*;
  - non polynomial time complexity $\Leftarrow$ high degeneracy;
  - Roohy-Laleh (1980): $O(n^5)$;
  - Akgül (1993): $O(n^3)$.



Feasible solution and strongly feasible tree (two views)

The root has degree one, and every other vertex of U has degree two.

Dual simplex algorithms

- Most famous algorithm *Signature method* by Balinski (1985);
- related to the strongly feasible trees;
- limited practical efficiency, but again theoretical relevance.

- *Signature* of a spanning tree T of $G =$ vector $d(T)$ of the degrees of the vertex $i \in U$ in T;

- Main Property: Given a dual feasible tree T in G,
  if $d_i(T) = 1$ for one vertex of U and $d_i(T) = 2$ for all other vertices of U,
  then the solution $x_{ij} = 1$ for all odd edges $[i, j] \in T$ is **optimal**.
- Time complexity $O(n^4)$;
- Cunningham (1983), Goldfarb (1985): time complexity $O(n^3)$.
- The algorithm is equivalent to the Hung and Rom dual non-simplex algorithm;
- Other signature algorithms: Akgül (1988), Paparrizos (1988).
- Using signatures, Balinski showed that the **Hirsch(1957) conjecture**
  "Given any two bases B_1 and B_2 of a linear program,
  there exists a sequence of $rank(B_1)$ adjacent feasible bases leading from B_1 to B_2."
  holds for the linear assignment problem.
- The Hirsch conjecture has been **disproved** in 2010 (F. Santos).

Evolution of algorithms for AP ($\mathcal{C} = \max_{i,j}\{c_{ij}\}$)

Year	Reference	Time complexity	Category
1946	Easterfield	exponential	Combinatorial
1955	Kuhn	$O(n^4)$	Primal-Dual
1957	Munkres	$O(n^4)$	Primal-Dual
1964	Balinski and Gomory	$O(n^4)$	Primal
1969	Dinic and Kronrod	$O(n^3)$	Dual
1971	Tomizawa	$O(n^3)$	Shortest path
1972	Edmonds and Karp	$O(n^3)$	Shortest path
1976	Lawler	$O(n^3)$	Primal-Dual
1976	Cunningham	exponential	Primal Simplex
1977	Barr, Glover and Klingman	exponential	Primal Simplex
1978	Cunningham and Marsh	$O(n^3)$	Primal
1980	Hung and Rom	$O(n^3)$	Dual
1981	Bertsekas	$O(n^3 + n^2\mathcal{C})$	Auction
1985	Balinski, Goldfarb	$O(n^3)$	Dual simplex
1985	Gabow	$O(n^{3/4} m \log \mathcal{C})$	Cost scaling
1988	Bertsekas and Eckstein	$O(nm \log(n\mathcal{C}))$	Auction + ε-rel.
1989	Gabow and Tarjan	$O(\sqrt{n}\ m \log(n\mathcal{C}))$	Cost scaling
1993	Orlin and Ahuja	$O(\sqrt{n}\ m \log(n\mathcal{C}))$	Auction + ε-rel.
1993	Akgül	$O(n^3)$	Primal Simplex
1995	Goldberg and Kennedy	$O(\sqrt{n}\ m \log(n\mathcal{C}))$	Pseudoflow
2001	Kao, Lam, Sung and Ting	$O(\sqrt{n}\ \mathcal{W} \log(\frac{n^2}{\mathcal{W}/\mathcal{C}}) / \log n)$	Decomposition

Average solution times. CPU seconds, Pentium 4

n	APC	CTCS	LAPJV	LAPm	NAUC	AFLP	AFR	CSA
				$c_{ij} \in [1, 10^3]$				
1000	0.42	0.05	0.05	0.01	0.18	0.31	0.44	0.15
2000	3.70	0.25	0.26	0.05	1.38	4.58	1.75	0.63
3000	21.33	7.62	0.74	0.13	5.84	52.32	7.10	1.95
4000	57.20	14.63	1.75	0.22	13.10	252.19	11.71	5.00
5000	89.78	18.61	2.55	0.30	18.75	tl	37.82	10.42
				$c_{ij} \in [1, 10^4]$				
1000	0.49	0.51	0.05	0.02	0.07	0.34	–	0.14
2000	3.01	3.16	0.26	0.06	0.49	1.53	–	0.61
3000	9.05	9.64	0.75	0.13	1.54	5.19	–	1.41
4000	18.97	20.50	1.85	0.23	3.50	7.19	–	2.58
5000	35.50	38.74	3.78	0.35	7.39	14.27	–	4.28
				$c_{ij} \in [1, 10^5]$				
1000	0.47	0.05	0.05	0.02	0.10	0.29	–	0.14
2000	2.94	0.23	0.28	0.06	0.70	1.01	–	0.60
3000	8.31	8.83	0.76	0.13	2.28	3.03	–	1.39
4000	17.57	18.72	2.05	0.22	5.19	3.79	–	2.60
5000	32.41	34.75	3.86	0.33	10.22	8.15	–	4.29
				$c_{ij} \in [1, 10^6]$				
1000	0.49	0.51	0.05	0.02	0.07	0.34	–	0.14
2000	3.01	3.16	0.26	0.06	0.49	1.53	–	0.61
3000	9.05	9.64	0.75	0.13	1.54	5.19	–	1.41
4000	18.97	20.50	1.85	0.23	3.50	7.19	–	2.58
5000	35.50	38.74	3.78	0.35	7.39	14.27	–	4.28

Another theorem by Egerváry (1931)

- We are given a **non-negative integer** $n \times n$ **matrix** T;
- consider the $n!$ distinct **permutation matrices** $P^k = (p_{ij}^k)$:
- a system of permutation matrices which contains the kth matrix, P^k, with **multiplicity** λ_k ($\lambda_k \geq 0 \ \forall \ k$) is called a **diagonal covering system** for T if

$$\sum_{k=1}^{n!} \lambda_k p_{ij}^k \geq t_{ij} \quad (i, j = 1, 2, \ldots, n). \tag{13}$$

- **Problem:** find a diagonal covering system of minimum value $\sum_{k=1}^{n!} \lambda_k$.
- **Question:** Do you recognize this problem?
- **Answer: Yes**, this is the **Time Division Multiple Access / Preemptive Open Shop Scheduling Problem** problem.
- **Theorem** (Egerváry, 1931): $\min \sum_{k=1}^{n!} \lambda_k = \max \left(\max_i \sum_{j=1}^{n} t_{ij}, \max_j \sum_{i=1}^{n} t_{ij} \right) (= t^*)$
- in other words, the minimum number of permutation matrices needed to cover a matrix T is equal to the maximum sum t^* of the elements in a line (row or column) of T.

- **Proof** (Egerváry, 1931): two steps:
  1. Define a **majorant** of T, i.e., a matrix T^* such that

$$t^*_{ij} \geq t_{ij} \quad (i,j = 1,2,\ldots,n) \text{ and}$$

$$\sum_{i=1}^{n} t^*_{ij} = \sum_{j=1}^{n} t^*_{ij} = t^* \quad (i,j = 1,2,\ldots,n).$$

  2. Iteratively subtract from T^* permutation matrices P such that, for at least one (ij), $t^*_{ij} > 0$ if $p_{ij} = 1$, until T^* becomes a zero matrix.
- **Question:** Do you recognize this algorithm?
- **Answer: Yes**, It is a simplified version of the Inukai/Gonzalez-Sahni algorithm of the Seventies.
- **Difference:** instead of subtracting the current permutation matrix P

  we subtract P multiplied by the smallest current t^*_{ij} value such that $p_{ij} = 1$.
- **In addition**, it implicitly contains the celebrated **Birkhoff-von Neumann theorem** on doubly stochastic matrices, that was published in Spanish by Garrett Birkhoff in 1946 and for which John von Neumann gave in 1953 an elegant elementary proof:

  Si una matriz $n \times n$ A satisface

$$\sum_{i=1}^{n} a_{ij} = \sum_{j=1}^{n} a_{ij} \quad (i,j = 1,2,\ldots,n)$$

  entonces es una media aritmética de permutaciones.