The Assignment Problem

E.A Dinic, M.A Kronrod

Moscow State University

January 30, 2012
1 Introduction
   - Motivation
   - Problem Definition

2 Algorithm
   - Basic Idea
   - Deficiency reduction
   - Finding Maximum delta
Outline

1. Introduction
   - Motivation
   - Problem Definition

2. Algorithm
   - Basic Idea
   - Deficiency reduction
   - Finding Maximum delta
Find the best way to assign each constructor with a job, paying the minimal cost.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>123$</td>
<td>210$</td>
<td>112$</td>
<td>180$</td>
<td>150$</td>
</tr>
<tr>
<td>2</td>
<td>573$</td>
<td>499$</td>
<td>680$</td>
<td>540$</td>
<td>510$</td>
</tr>
<tr>
<td>3</td>
<td>360$</td>
<td>240$</td>
<td>370$</td>
<td>362$</td>
<td>250$</td>
</tr>
<tr>
<td>4</td>
<td>780$</td>
<td>999$</td>
<td>600$</td>
<td>820$</td>
<td>900$</td>
</tr>
<tr>
<td>5</td>
<td>450$</td>
<td>500$</td>
<td>360$</td>
<td>440$</td>
<td>480$</td>
</tr>
</tbody>
</table>
Find the best way to assign each constructor with a job, paying the minimal cost.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>123$</td>
<td>210$</td>
<td>112$</td>
<td>180$</td>
<td>150$</td>
</tr>
<tr>
<td>2</td>
<td>573$</td>
<td>499$</td>
<td>680$</td>
<td>540$</td>
<td>510$</td>
</tr>
<tr>
<td>3</td>
<td>360$</td>
<td>240$</td>
<td>370$</td>
<td>362$</td>
<td>250$</td>
</tr>
<tr>
<td>4</td>
<td>780$</td>
<td>999$</td>
<td>600$</td>
<td>820$</td>
<td>900$</td>
</tr>
<tr>
<td>5</td>
<td>450$</td>
<td>500$</td>
<td>360$</td>
<td>440$</td>
<td>480$</td>
</tr>
</tbody>
</table>

Valid solution 2082$
Find the best way to assign each constructor with a job, paying the minimal cost.

Valid solution 2081$
Find the best way to assign each constructor with a job, paying the minimal cost.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>123$</td>
<td>210$</td>
<td>112$</td>
<td>180$</td>
<td>150$</td>
</tr>
<tr>
<td>2</td>
<td>573$</td>
<td>499$</td>
<td>680$</td>
<td>540$</td>
<td>510$</td>
</tr>
<tr>
<td>3</td>
<td>360$</td>
<td>240$</td>
<td>370$</td>
<td>362$</td>
<td>250$</td>
</tr>
<tr>
<td>4</td>
<td>780$</td>
<td>999$</td>
<td>600$</td>
<td>820$</td>
<td>900$</td>
</tr>
<tr>
<td>5</td>
<td>479$</td>
<td>500$</td>
<td>360$</td>
<td>440$</td>
<td>480$</td>
</tr>
</tbody>
</table>

Optimal solution 1912$
Outline

1. Introduction
   - Motivation
   - Problem Definition

2. Algorithm
   - Basic Idea
   - Deficiency reduction
   - Finding Maximum delta

E.A Dinic and M.A Kronrod
Problem Definition

Input:
Square matrix, A, of order \( n \)

Output:
A set of an \( n \) elements (cells), exactly one in each row and each column, such that the sum of these elements is minimal with respect to all such sets.
So what is a solution?

A permutation $\beta$ over the set $\{1, \ldots, n\}$ such that for any permutation $\lambda$:

$$\sum_{i=1}^{n} a_{i, \beta(i)} \leq \sum_{i=1}^{n} a_{i, \lambda(i)}.$$ 

In which cases is it easy to find the solution?

Example

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>6</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
Outline

1 Introduction
   • Motivation
   • Problem Definition

2 Algorithm
   • Basic Idea
   • Deficiency reduction
   • Finding Maximum delta
Definition

Let some vector $\Delta = (\Delta_1, \ldots, \Delta_n)$ be given. An element, $a_{ij}$, of the matrix $A$ is called $\Delta$-minimal if

$$\forall 1 \leq k \leq n \ a_{ij} - \Delta_j \leq a_{ik} - \Delta_k$$

Example:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>8</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>15</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>3</td>
<td>7</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
Definition

Let some vector $\Delta = (\Delta_1, \ldots, \Delta_n)$ be given. An element, $a_{ij}$, of the matrix $A$ is called $\Delta$-minimal if

$$\forall 1 \leq k \leq n \quad a_{ij} - \Delta_j \leq a_{ik} - \Delta_k$$

Example:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>8</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>15</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>3</td>
<td>7</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

$\rightarrow$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>8</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>15</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>3</td>
<td>7</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
Let some vector $\Delta = (\Delta_1, \ldots, \Delta_n)$ be given. An element, $a_{ij}$, of the matrix $A$ is called $\Delta$-minimal if

$$\forall 1 \leq k \leq n \ a_{ij} - \Delta_j \leq a_{ik} - \Delta_k$$

Lemma

For any $\Delta$ let there be given a set of $n$ $\Delta$-minimal elements: $a_{1j_1}, a_{2j_2}, \ldots, a_{nj_n}$, one from each row and each column. Then this set is an optimal solution for the Assignment Problem.
Let some vector $\Delta = (\Delta_1, \ldots, \Delta_n)$ be given. An element, $a_{ij}$, of the matrix $A$ is called $\Delta$-minimal if

$$\forall 1 \leq k \leq n \ a_{ij} - \Delta_j \leq a_{ik} - \Delta_k$$

**Lemma**

For any $\Delta$ let there be given a set of $n$ $\Delta$-minimal elements: $a_{1j_1}, a_{2j_2}, \ldots, a_{nj_n}$, one from each row and each column. Then this set is an optimal solution for the Assignment Problem.

**Proof**

1. For some vector $\Delta = (\Delta_1, \ldots, \Delta_n)$.
   A set of $n$ $\Delta$-minimal elements has the minimal sum among all sets of $n$ elements one from each column.

2. A set of $n$ $\Delta$-minimal elements one from each row and each column is a minimal and valid solution.
For some vector $\Delta = (\Delta_1, \ldots, \Delta_n)$.
A set of $n \Delta$-minimal elements has the minimal sum among all sets of $n$ elements one from each column.

Proof:
Let there be a set of $n$ elements $a_{1j_1}, a_{2j_2}, \ldots, a_{nj_n}$ we can write the sum of the set as:

$$\sum_{i=1}^{n} a_{ij_i} = \sum_{k=1}^{n} \Delta_k + \sum_{i=1}^{n} (a_{ij_i} - \Delta_{j_i})$$

Let there be a set of $n \Delta$-minimal elements $a^*_{1c_1}, a^*_{2c_2}, \ldots, a^*_{nc_n}$

$$\sum_{i=1}^{n} a^*_{ic_i} = \sum_{k=1}^{n} \Delta_k + \sum_{i=1}^{n} (a^*_{ic_i} - \Delta_{c_i})$$

$$\sum_{k=1}^{n} \Delta_k + \sum_{n}^{i=1} (a^*_{ic_i} - \Delta_{c_i}) \leq \sum_{k=1}^{n} \Delta_k + \sum_{n}^{i=1} (a_{ij_i} - \Delta_{j_i})$$

$$\sum_{i=1}^{n} a^*_{ic_i} \leq \sum_{i=1}^{n} a_{ij_i}$$
More definitions

- Given a vector $\Delta$, an element $a_{ij}$ is a **basic** if it is a $\Delta$-**minimal** element of the row $i$.
- A **set of basics** is a set of $n$ basics, one from each row.
- **Deficiency** of a set of basics is the number of free columns, i.e. columns without a basic.

$$
\begin{array}{ccc}
 & 1 & 2 & 3 \\
1 & 2 & 5 & 4 \\
2 & 9 & 8 & 10 \\
3 & 12 & 15 & 7 \\
\hline
\Delta & 1 & 1 & 5 \\
\end{array}
$$
More definitions

Given a vector $\Delta$, an element $a_{ij}$ is a basic if it is a $\Delta$-minimal element of the row $i$.

A set of basics is a set of $n$ basics, one from each row.

Deficiency of a set of basics is the number of free columns, i.e. columns without a basic.

$$
\begin{array}{cccc}
 & 1 & 2 & 3 \\
1 & 2 & 5 & 4 \\
2 & 9 & 8 & 10 \\
3 & 12 & 15 & 7 \\
\Delta & 1 & 1 & 5 \\
\end{array}
$$

deficiency=2.
Redefinition of problem

Input:
- Square matrix, A, of order $n$

Output:
- vector, $\Delta$, of size $n$
- a set of an $n$ basics, with deficiency 0.
Integer linear programming problem

Given the \( n \times n \) matrix \( C \) we will define an \( n \times n \) matrix \( X \) of integer variables. The following constraints define the equivalent linear programming problem.

**linear constraints:**

1. All the variables of \( X \) are 0 or 1:
   \[ \forall i, j \; x_{i,j} \in \{0, 1\}. \]
2. In each row and column the sum of variables is 1:
   \[ \forall i \; \sum_{j=0}^{n} x_{i,j} = \sum_{j=0}^{n} x_{j,i} = 1. \]

**Goal function:**

minimize \[ \sum_{i=0}^{n} \sum_{i=0}^{n} x_{i,j} c_{i,j}. \]
In the primal-dual method we generate a dual linear programming problem such that for every variable in the original problem we have a constraint in the dual problem, and for every constraint in the original we have a variable in the dual.
Primal-dual method

We iterate on the pairs: primal and dual solutions. At any time we have a NON-FEASIBLE primal solution $S$ to the primal problem, while the dual solution PROVES that $S$ is OPTIMAL among the ”similarly non-feasible” primal solutions. In the end of the process we have a feasible, and thus optimal solution to the original problem.
Intuition continues

We would want a function $f$ such that for a matrix $A$ with a solution $\beta$, $f(A)$ is a matrix for which $\beta$ is a row minimal solution.

Example: $f$-function

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

$A$

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

$f(A)$

Notice: $f(A)$ is obtained by subtracting 1 from all the elements of the first column of $A$. 
The function $f$

**Input:**

$\Delta = (\Delta_1, \ldots, \Delta_n)$, $A$ an $n \times n$ matrix

**Output:** $f_\Delta(A) = B = (b_{i,j})$

for every indice $(i, j) \in \{1, \ldots, n\}^2$ $b_{i,j} = a_{i,j} - \Delta_j$.

<table>
<thead>
<tr>
<th>$A$</th>
<th>$f_\Delta(A)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>4 6 2 2</td>
</tr>
<tr>
<td>8</td>
<td>3 1 3 1</td>
</tr>
<tr>
<td>6</td>
<td>5 3 4 3</td>
</tr>
<tr>
<td>5</td>
<td>2 5 2 5</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>3 2 2 0</td>
</tr>
</tbody>
</table>

E.A Dinic and M.A Kronrod
Redfinition of problem

Input:

- Square matrix, A, of order \( n \)

Output:

- vector, \( \Delta \), of size \( n \)
- a set of an \( n \) basics, with deficiancy 0.
Deficiency reduction

We will solve this in an iterative manner, such that in each iteration we will reduce the deficiency by 1.

**Input:**
- Square matrix, $A$, of order $n$
- Vector, $\Delta$, of size $n$
- A set of $n$ basics, with deficiency $m$.

**Output:**
- Vector, $\Delta'$, of size $n$
- A set of $n$ basics, with deficiency $m-1$.

In the first iteration we start with $\Delta = (0, ..., 0)$, finding the basics and the deficiency takes $O(n^2)$. 

E.A Dinic and M.A Kronrod
Outline

1 Introduction
   - Motivation
   - Problem Definition

2 Algorithm
   - Basic Idea
   - Deficiency reduction
   - Finding Maximum delta
Phase 1 - Finding alternative Basics

We begin with vector $\Delta$ and a set of basics $a_{1,j(1)}, \ldots, a_{n,j(n)}$

\[
\begin{array}{cccc}
7 & 8 & 4 & 2 \\
6 & 3 & 5 & 1 \\
8 & 5 & 6 & 3 \\
5 & 7 & 4 & 5 \\
0 & 0 & 0 & 0 \\
\end{array}
\]
Phase 1 - Finding alternative Basics

Let $s_1$ be the index of a free column.

<table>
<thead>
<tr>
<th>S1</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Phase 1 - Finding alternative Basics

We will increase $\Delta_{s_1}$ with maximal $\delta_1$ such that all basics remain $\Delta$-minimal elements (let's assume we have a function which finds such a $\delta$).

$$\delta=1$$
Phase 1 - Finding alternative Basics

We obtain that for some row index $i_1$ $a_{i_1,s_1} - \Delta s_1 = a_{i_1,j(i_1)} - \Delta j(i_1)$.

$a_{i_1,s_1}$ is called an alternative basic.
Phase 1 - Finding alternative Basics

We define $s_2 = j(i_1)$.

\[
\begin{array}{ccc}
S1 & S2 \\
7 & 8 & 4 & 2 \\
6 & 3 & 5 & 1 \\
8 & 5 & 6 & 3 \\
\end{array}
\]

\[
\begin{array}{c}
i_1 \\
5 & 7 & 4 & 5 \\
\end{array}
\]
Phase 1 - Finding alternative Basics

We now increase $\Delta_s_1$, $\Delta_s_2$ with maximal $\delta_2$ such that all basics remain $\Delta$-minimal.

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
<th>$\delta$=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
**Phase 1 - Finding alternative Basics**

Again for same row index $i_2 \neq i_1$  

\[
a_{i_2,s_k} - \Delta s_k = a_{i_2,j(i_2)} - \Delta j(i_2)
\]

were $k \in \{1, 2\}$. $a_{i_2,s_k}$ is an alternative basic.

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th></th>
<th>S2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$i_2$</td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>5</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>$i_1$</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

$4-2=2-0$
Phase 1 - Finding alternative Basics

We define $s_3 = j(i_2)$. We will continue this process until we find an alternative basic in a column with 2 or more basics.
Phase 1 - Pseudo Code

Input:
- \((a_x, y)\) nxn matrix
- \(\Delta\) n long vector
- \(j(i)\) function such that for each row \(a_{i,j(i)}\) is a basic

\[ S = \{\text{chooseEmptyColumn}(j)\} \]
\[ R = \{\} \]
do: 
\[ \delta = \text{findMaxPreserving}\Delta\text{Minimalty}(R, S, (a_x, y), \Delta, j(i)) \]
for \(s \in S\) do: \(\Delta_s = \Delta_s + \delta\)
let \(i \in \{1, ..., n\} \setminus R\) such that \(\exists s \in S\) \(a_{i,j(i)} - \Delta_j(i) = a_{i,s} - \Delta_s\.
\[ R = R \cup \{i\} \]
\[ S = S \cup \{j(i)\} \]
while every column in \(S\) has 1 or 0 basics.
Phase 1 - Complexity Analysis

In each step of phase 1:
- $\delta$ is found - $O(n^2)$
- $\Delta$ is updated - $O(n)$
- A new alternative basic is found (during the search of $\delta$) - $O(1)$

In each round the size of $S$ increases by 1, and $S$ is bounded by $n$

\[ \Downarrow \]

There are at most $n - 1$ steps in phase 1.

Total complexity: $O(n) \times [O(n^2) + O(n) + O(1)] = O(n^3)$
Phase 2 - Change of basics

Now as we mark a column \((s_3)\) with 2 or more basics. This is the end of phase 1. We start changing our basics.

\[
\begin{array}{ccc}
\text{S1} & \text{S2} & \text{S3} \\
\hline
\text{i2} & 7 & 8 & 4 & 2 \\
\text{i1} & 5 & 7 & 4 & 5 \\
\end{array}
\]
Phase 2 - Change of basics

We reduce the number of basics for our last marked column by one.
Phase 2 - Change of basics

In total we reduce the deficiency by 1.
Phase 2 - Complexity Analysis

The complexity of this step is $O(n)$ as the number of basics.
Example continues

We start phase 1 again and choose a column $s_1$ with no basics. $S = \{s_1\}$. $\Delta$ remains as it was built at the previous iteration $\Rightarrow$ all basics remain $\Delta$-minimal.
Example continues

We find a maximal $\delta$ to add to $\Delta_s$ were $s \in S$, such that it preserves $\Delta$-minimality.

$$\delta=2$$
Example continues

For some row index $i_1$, $a_{i_1,s_1} - \Delta s_1 = a_{i_1,j(i_1)} - \Delta j(i_1)$. $a_{i_1,s_1}$ is an alternative basic.

\[
\begin{array}{cccc}
7 & 8 & 4 & 2 \\
6 & 3 & 5 & 1 \\
8 & 5 & 6 & 3 \\
5 & 7 & 4 & 5 \\
3 & 2 & 2 & 0 \\
\end{array}
\]

$3-2=1-0$
We end phase 1 as we found a column $j(i_1) = s_2 \in S$ with more than one basic.

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th></th>
<th>S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>i₁</td>
<td>7 8 4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6 3 5 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 5 6 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 7 4 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 2 2 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example continues

Changing our basics leads us to a set of basics with deficiency $m = 0$. Therefore it is an optimal solution. $B = (a_{i,j} - \Delta_j)$. 
Outline

1 Introduction
   • Motivation
   • Problem Definition

2 Algorithm
   • Basic Idea
   • Deficiency reduction
   • Finding Maximum delta
Naive computation

\[ \delta = \min_{i \in R, s \in S} [(a_{i,s} - \Delta_s) - (a_{i,j(i)} - \Delta_{j(i)})] \]

Where:

- \( R \) - set of row indices which do not contain alternative basics
- \( S \) - set of potential alternative basics column indices
- \((a_{x,y})\) - nxn matrix
- \( \Delta \) - n long vector
- \( j(i) \) - function such that for each row \( a_{i,j(i)} \) is the basic in row \( i \)

Computing \( \delta \) in a straightforward manner takes \( O(n^2) \)
The maximum deficiency is $n - 1$.

In each iteration we perform phase 1 + phase 2: $O(n^3) + O(n)$

Total complexity: $O(n) \times [O(n^3) + O(n)] = O(n^4)$
First improvement

\[ \delta = \min_{i \in R, s \in S} [(a_{is} - \Delta_s) - (a_{ij} - \Delta_j)] \]

- For each \( k \), let \( b_k = (b_{1k}, ..., b_{nk}) \) be a column of the values:
  \[ b_{ik} = [(a_{ik} - \Delta_k) - (a_{ij} - \Delta_j)] \]
- Let \( B \) be the \( nxn \) matrix: \( (b_1^*, ..., b_n^*) \),
  where \( b_k^* = \text{Sort}(b_k) \)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>( \Delta )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
**First improvement**

\[ \delta = \min_{i \in R, s \in S} [(a_{is} - \Delta_s) - (a_{ij} - \Delta_j)] \]

- For each k, let \( b_k = (b_{1k}, ..., b_{nk}) \) be a column of the values:
  \[ b_{ik} = [(a_{ik} - \Delta_k) - (a_{ij} - \Delta_j)] \]
- Let \( B \) be the \( nxn \) matrix: \((b^*_1, ..., b^*_n)\), where \( b^*_k = \text{Sort}(b_k) \)

**Example**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>( \Delta )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- \( b_1 \):
  - 0
  - 1
  - 5

- \( b_2 \):
  - 3
  - 0
  - 8

- \( b_3 \):
  - 2
  - 2
  - 0

\( E.A \) Dinic and \( M.A \) Kronrod
First improvement

\[ \delta = \min_{i \in R, s \in S} [(a_{is} - \Delta_s) - (a_{ij} - \Delta_{ji})] \]

- For each k, let \( b_k = (b_{1k}, ..., b_{nk}) \) be a column of the values:
  \[ b_{ik} = [(a_{ik} - \Delta_k) - (a_{ij} - \Delta_{ji})] \]
- Let \( B \) be the \( nxn \) matrix: \( (b^*_1, ..., b^*_n) \),
  where \( b^*_k = \text{Sort}(b_k) \)

**Example**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>(\Delta)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(b_1)</th>
<th>(b_2)</th>
<th>(b_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b_1)</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>(b_2)</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>(b_3)</td>
<td>5</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(b_1^*)</th>
<th>(b_2^*)</th>
<th>(b_3^*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b_1^*)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(b_2^*)</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>(b_3^*)</td>
<td>5</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>
First improvement

\[ \delta = \min_{i \in R, s \in S} [(a_{is} - \Delta_s) - (a_{ij} - \Delta_j)] \]

- For each \( k \), let \( b_k = (b_{1k}, \ldots, b_{nk}) \) be a column of the values:
  \[ b_{ik} = [(a_{ik} - \Delta_k) - (a_{ij} - \Delta_j)] \]
- Let \( B \) be the \( nxn \) matrix: \( (b_1^*, \ldots, b_n^*) \),
  where \( b_k^* = \text{Sort}(b_k) \)

What is the complexity of the construction of \( B \)?
First improvement

\[ \delta = \min_{i \in R, s \in S} [(a_{is} - \Delta_s) - (a_{ij} - \Delta_j)] \]

- For each \( k \), let \( b_k = (b_{1k}, \ldots, b_{nk}) \) be a column of the values:
  \[ b_{ik} = [(a_{ik} - \Delta_k) - (a_{ij} - \Delta_j)] \]

- Let \( B \) be the \( nxn \) matrix: \( (b_1^*, \ldots, b_n^*) \), where \( b_k^* = \text{Sort}(b_k) \)

What is the complexity of the construction of \( B \)?

\[
\begin{align*}
& n \times O(n \log(n)) \\
\downarrow & \\
& O(n^2 \log(n))
\end{align*}
\]
First improvement

- As preprocessing phase of an iteration build matrix $B$ $O(n^2 \log n)$.

In each succeeding step of phase 1:
- clear the matrix from items of rows which are not in $R$.
  $n \times O(1) = O(n)$
- find $\min_{k \in S} b_k$ $O(n)$
First improvement

- As preprocessing phase of an iteration build matrix $B$ 
  $O(n^2 \log n)$.

In each succeeding step of phase 1:

- clear the matrix from items of rows which are not in $R$. 
  $n \times O(1) = O(n)$

- find $\min_{k \in S} b_k$  $O(n)$

What is the total complexity?
First improvement

- As preprocessing phase of an iteration build matrix B $O(n^2 \log n)$.

In each succeeding step of phase 1:
- clear the matrix from items of rows which are not in R.
  \[ n \times O(1) = O(n) \]
- find $\min_{k \in S} b_k$ $O(n)$

What is the total complexity?

\[ n \times \left[ O(n^2 \log n) + n \times (O(n) + O(n) + O(n)) + n \right] \]

\[ \text{phase 0} \quad \text{phase 1} \quad \text{phase 2} \]

one iteration
First improvement

- As preprocessing phase of an iteration build matrix B
  \( O(n^2 \log n) \).

In each succeeding step of phase 1:
- clear the matrix from items of rows which are not in R.
  \( n \times O(1) = O(n) \)
- find \( min_{k \in S} b_k \) \( O(n) \)

What is the total complexity?
\( O(n^3 \log n) \)
Second Improvement

\[ \delta = \min_{i \in R, s \in S} [(a_{is} - \Delta_s) - (a_{ij} - \Delta_j)] \]
Second Improvement

\[ \delta = \min_{i \in R, s \in S} \left[ (a_{is} - \Delta_s) - (a_{ij} - \Delta_j) \right] \]
Second Improvement

\[
\delta = \min_{i \in R, s \in S} [(a_{is} - \Delta_s) - (a_{ij} - \Delta_j)] \\
\delta = \min_{i \in R} \left[ \min_{s \in S} [(a_{is} - \Delta_s) - (a_{ij} - \Delta_j)] \right]
\]
Second Improvement

\[
\delta = \min_{i \in R, s \in S} [(a_{is} - \Delta_s) - (a_{ij} - \Delta_{ji})]
\]

At the beginning of an iteration compute the column vector \( q_i \).

In each succeeding step of phase 1:

update the vector \( q_i \):

\[
\forall i \quad q_i \leftarrow \min \left[ a_{is} - \Delta_s; q_i - \delta \right]
\]

const. for a row
Second Improvement

\[ \delta = \min_{i \in R, s \in S} [(a_{is} - \Delta_s) - (a_{ij_i} - \Delta_{ji})] \]

\[ \downarrow \]

\[ \delta = \min_{i \in R} \left[ \min_{s \in S} [(a_{is} - \Delta_s)] - (a_{ij_i} - \Delta_{ji}) \right] \]

const. for a row

E.A Dinic and M.A Kronrod
Second Improvement

\[ \delta = \min_{i \in R, s \in S} [(a_{is} - \Delta_s) - (a_{ij} - \Delta_{ji})] \]

\[ \downarrow \]

\[ \delta = \min_{i \in R} [ \min_{s \in S} [(a_{is} - \Delta_s) - (a_{ij} - \Delta_{ji})] ] \]

\[ \text{const. for a row} \]

\[ \downarrow \]

\[ \delta = \min_{i \in R} [ \min_{s \in S} (a_{is} - \Delta_s) - (a_{ij} - \Delta_{ji}) ] \]
Second Improvement

\[ \delta = \min_{i \in R, s \in S} [(a_{is} - \Delta_s) - (a_{ij} - \Delta_{ji})] \]

\[ \delta = \min_{i \in R} \left[ \min_{s \in S} [(a_{is} - \Delta_s)] - (a_{ij} - \Delta_{ji}) \right] \]

\[ \delta = \min_{i \in R} \left[ \min_{s \in S} (a_{is} - \Delta_s) - (a_{ij} - \Delta_{ji}) \right] \]

At the beginning of an iteration compute the column vector \( q_i \).

In each succeeding step of phase 1:

update the vector \( q_i \):

\[ \forall i \ q_i \leftarrow \min \left[ a_{is} - \Delta_s; q_i - \delta \right] \]
Second Improvement

\[ \delta = \min_{i \in R, s \in S} [(a_{is} - \Delta_s) - (a_{ij} - \Delta_{ji})] \]

\[ \delta = \min_{i \in R} \left[ \min_{s \in S} [(a_{is} - \Delta_s)] - (a_{ij} - \Delta_{ji}) \right] \]

\[ \downarrow \]

\[ \delta = \min_{i \in R} \left[ \min_{s \in S} (a_{is} - \Delta_s) - (a_{ij} - \Delta_{ji}) \right] \]

- At the beginning of an iteration compute the column vector \( q_i \)

In each succeeding step of phase 1:

- update the vector \( q \):
  \[ \forall i \quad q_i \leftarrow \min [a_{ism} - \Delta_{sm}; q_i - \delta] \]
Second improvement

- At the beginning of an iteration compute the column vector $q_i$
  $O(n)$

In each succeeding step of phase 1:

- update the vector $q$:
  $\forall i, q_i \leftarrow \min[a_{is_m} - \Delta_{sm}; q_i - \delta]$
  $O(n)$

What is the total complexity?
Second improvement

- At the beginning of an iteration compute the column vector $q_i$:
  $O(n)$

In each succeeding step of phase 1:

- update the vector $q$:
  $\forall i \quad q_i \leftarrow \min[a_{ism} - \Delta_{sm}; q_i - \delta]$
  $O(n)$

What is the total complexity?

$$n \times [O(n) + n \times O(n) + n]$$

- phase 0
- phase 1
- phase 2

one iteration

E.A Dinic and M.A Kronrod
Second improvement

- At the beginning of an iteration compute the column vector $q_i$
  $O(n)$

In each succeeding step of phase 1:

- update the vector $q$:
  $\forall i \ q_i \leftarrow \min[a_{is_m} - \Delta_{s_m}; q_i - \delta]$
  $O(n)$

What is the total complexity?

$O(n^3)$