The Assignment Problem

E.A Dinic, M.A Kronrod

Moscow State University

Soviet Math.Dokl. 1969

January 30, 2012

1. Introduction
   - Motivation
   - Problem Definition

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

Outline

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

	A	B	C	D	E
1	123$	210$	112$	180$	150$
2	573$	499$	680$	540$	510$
3	360$	240$	370$	362$	250$
4	780$	999$	600$	820$	900$
5	450$	500$	360$	440$	480$

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

Valid solution

2082$

	A	B	C	D	E
1	123$	210$	112$	180$	150$
2	573$	499$	680$	540$	510$
3	360$	240$	370$	362$	250$
4	780$	999$	600$	820$	900$
5	450$	500$	360$	440$	480$

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

Valid solution
2081$

	A	B	C	D	E
1	123$	210$	112$	180$	150$
2	573$	499$	680$	540$	510$
3	360$	240$	370$	362$	250$
4	780$	999$	600$	820$	900$
5	450$	500$	360$	440$	480$

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

	A	B	C	D	E
1	123$	210$	112$	180$	150$
2	573$	499$	680$	540$	510$
3	360$	240$	370$	362$	250$
4	780$	999$	600$	820$	900$
5	479$	500$	360$	440$	480$

Optimal solution
1912$

Outline

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 permutaion β over the set $\{1, ..., n\}$
such that for any permutation λ:

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

In which cases is it easy to find the solution?

Example

4	6	2	2
3	1	3	1
5	3	4	3
2	5	2	5

Outline

Definition

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

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

Example:

	1	2	3	4
1	2	5	4	1
2	9	8	10	2
3	12	15	7	4
4	7	8	9	3
Δ	3	7	3	1

Definition

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

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

Example:

	1	2	3	4
1	2	5	4	1
2	9	8	10	2
3	12	15	7	4
4	7	8	9	3
Δ	3	7	3	1

→

	1	2	3	4
1	2	**5**	4	1
2	9	8	10	**2**
3	12	15	7	**4**
4	7	**8**	9	3

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

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

Lemma

For any Δ let there be given a set of n Δ-*minimal* elements: $a_{1j_1}, a_{2j_2}, ..., a_{nj_n}$, 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 *Δ-minimal* if

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

Lemma

For any Δ let there be given a set of n Δ-*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 Δ-*minimal* elements has the minimal sum among all sets of n elements one from each column.
2. A set of n Δ-*minimal* elements one from each row and each column is a minimal and valid solution.

For some vector $\Delta = (\Delta_1, ..., \Delta_n)$.
A set of n Δ-*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 $a_{1j_1}, a_{2j_2}, ..., a_{nj_n}$
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 Δ-*minimal* elements $a^*{}_{1c_1}, a^*{}_{2c_2}, ..., a^*{}_{nc_n}$

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

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

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

$$\Downarrow$$

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

More definitions

- Given a vector Δ, an element a_{ij} is a *basic* if it is a Δ-*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.

	1	2	3
1	2	5	4
2	9	8	10
3	12	15	7
Δ	1	1	5

.

More definitions

- Given a vector Δ, an element a_{ij} is a *basic* if it is a Δ-*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.

	1	2	3
1	2	5	4
2	9	8	10
3	12	15	7
Δ	1	1	5

deficiency=2.

Redfenition of problem

Input:

- Square matrix, A, of order n

Output:

- vector, Δ, of size n
- a set of an n basics, with deficiancy 0.

Integer linear programming problem

Given the $n * n$ matrix C we will define an $n * n$ 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 $\sum_{i=0}^{n} \sum_{i=0}^{n} x_{i,j} c_{i,j}$.

Primal-dual method

In the primal-dual method we generate a dual linear programing 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 soultion β, $f(A)$ is a matrix for which β is a row minimal soultion.

Example: f-function

A

2	1
4	2

f(A)

1	1
3	2

Notice: $f(A)$ is obtaind by substracting 1 from all the elemnts of the first column of A.

The function f

Input:

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

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

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

A

7	8	4	2
6	3	5	1
8	5	6	3
5	7	4	5

$f_\Delta(A)$

4	6	2	2
3	1	3	1
5	3	4	3
2	5	2	5

Δ

3	2	2	0

Redfenition of problem

Input:

- Square matrix, A, of order n

Output:

- vector, Δ, of size n
- a set of an n basics, with deficiancy 0.

Deficiency reduction

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

Input:

- Square matrix, A, of order n
- vector, Δ, of size n
- a set of n basics, with deficiancy m.

Output:

- vector, Δ', of size n
- a set of n basics, with deficiancy m-1.

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

Outline

1. Introduction
   - Motivation
   - Problem Definition
2. Algorithm
   - Basic Idea
   - Deficiency reduction
   - Finding Maximum delta

Phase 1 - Finding alternative Basics

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

7	8	4	2
6	3	5	1
8	5	6	3
5	7	4	5

0	0	0	0

Phase 1 - Finding alternative Basics

Let s_1 be the index of a free column.

S1

7	8	4	2
6	3	5	1
8	5	6	3
5	7	4	5

0	0	0	0

Phase 1 - Finding alternative Basics

We will increase Δ_{s_1} with maximal δ_1 such that all basics remain Δ-minimal elements (lets assume we have a function which finds such a δ).

S1

7	8	4	2
6	3	5	1
8	5	6	3
5	7	4	5

1	0	0	0

δ=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.

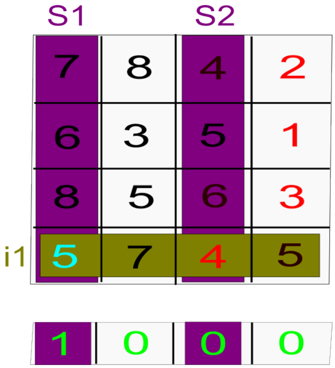
	S1			
	7	8	4	2
	6	3	5	1
	8	5	6	3
i1	5	7	4	5

5-1 = 4-0

1	0	0	0

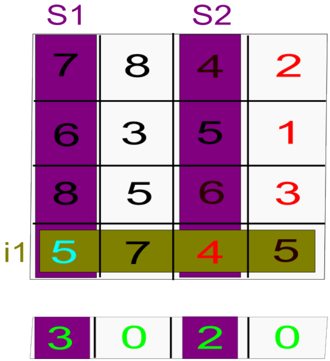
Phase 1 - Finding alternative Basics

We define $s_2 = j(i_1)$.



Phase 1 - Finding alternative Basics

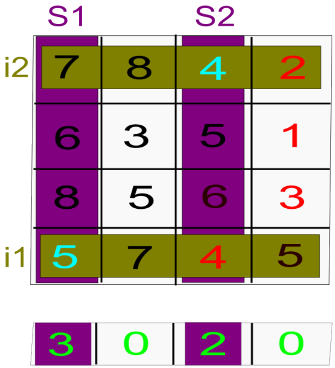
We now increase Δ_{s_1}, Δ_{s_2} with maximal δ_2 such that all basics remain Δ-minimal.



δ=2

Phase 1 - Finding alternative Basics

Again for same row index $i_2 \neq i_1$ $\quad a_{i_2,s_k} - \Delta_{s_k} = a_{i_2,j(i_2)} - \Delta_{j(i_2)}$ were $k \in \{1,2\}$. a_{i_2,s_k} is an alternative basic.



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.

	S1		S2	S3
i2	7	8	4	2
	6	3	5	1
	8	5	6	3
i1	5	7	4	5

3	0	2	0

Phase 1 - Pseudo Code

Input:

- $(a_{x,y})$ nxn matrix
- Δ n long vector
- $j(i)$ function such that for each row $a_{i,j(i)}$ is a basic

$S = \{chooseEmptyColumn(j)\}$
$R = \{\}$
do:
  $\delta = findMaxPreserving\Delta Minimalty(R, S, (a_{x,y}), \Delta, j(i))$
  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:

- δ is found - $O(n^2)$
- Δ is updated - $O(n)$
- A new alternative basic is found (during the search of δ)- $O(1)$

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

$\Downarrow$

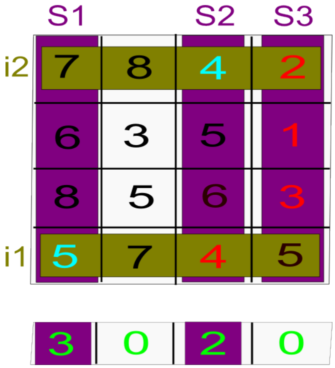
There are at most $n-1$ steps in phase 1.

Total complexity: $O(n)x[O(n^2)+O(n)+O(1)] =$

$O(n^3)$

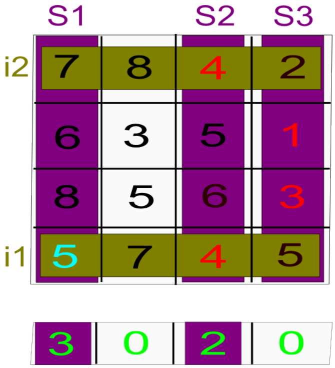
Phase 2 - Change of basics

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.



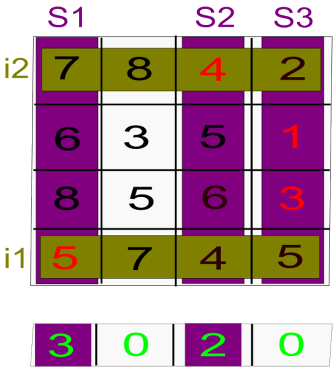
Phase 2 - Change of basics

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. $S = \{s_1\}$. Δ remains as it was built at the previous iteration $\Rightarrow$ all basics remain Δ-minimal.

	S1		
7	8	4	2
6	3	5	1
8	5	6	3
5	7	4	5

3	0	2	0

Example continues

We find a maximal δ to add to Δ_s were $s \in S$, such that it preserves Δ-minimality.

	S1		
7	8	4	2
6	3	5	1
8	5	6	3
5	7	4	5

3	2	2	0

δ=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.

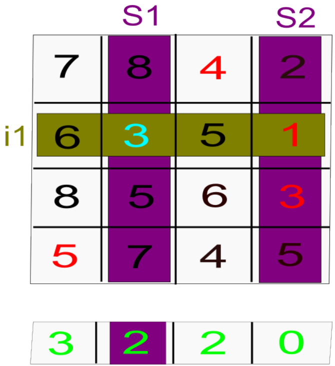
		S1		
	7	8	4	2
i1	6	3	5	1
	8	5	6	3
	5	7	4	5

3	2	2	0

3-2=1-0

Example continues

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



Example continues

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

		S1		S2
	7	8	4	2
i1	6	3	5	1
	8	5	6	3
	5	7	4	5

3	2	2	0

B

4	6	2	2
3	1	3	1
5	3	4	3
2	5	2	5

Outline

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
- Δ -n long vector
- $j(i)$ -function such that for each row $a_{i,j(i)}$ is the basic in row i

Computing δ in a strightforward manner takes $O(n^2)$

Total Complexity Analysis

- The maximum deficiency is $n - 1$.
- In each itration we preform phase 1 + phase 2: $O(n^3) + O(n)$

Total complexity: $O(n)x[O(n^3) + O(n)] = O(n^4)$

First improvement

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

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

Example

	1	2	3
1	2	5	4
2	9	8	10
3	12	15	7
Δ	0	0	0

First improvement

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

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

Example

	1	2	3
1	2	5	4
2	9	8	10
3	12	15	7
Δ	0	0	0

b_1	b_2	b_3
0	3	2
1	0	2
5	8	0

First improvement

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

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

Example

	1	2	3
1	2	5	4
2	9	8	10
3	12	15	7
Δ	0	0	0

b_1	b_2	b_3
0	3	2
1	0	2
5	8	0

B

b_1^*	b_2^*	b_3^*
0	0	0
1	3	2
5	8	2

First improvement

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

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

What is the compexity of the construction of B?

First improvement

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

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

What is the compexity of the construction of B?

$$n \times O(n \log(n))$$
$$\downarrow$$
$$O(n^2 \log(n))$$

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

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?

$$n \times [\underbrace{\underbrace{O(n^2 \log n)}_{phase0} + \underbrace{n \times (O(n) + O(n) + O(n))}_{phase1} + \underbrace{n}_{phase2}}_{\text{one iteration}}]$$

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

What is the total complexity?

$$O(n^3 \log n)$$

Second Improvement

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

Second Improvement

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

Second Improvement

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

$$\downarrow$$

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

Second Improvement

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

$$\downarrow$$

$$\delta = min_{i \in R}[\ min_{s \in S}[(a_{is} - \Delta_s)] - \underbrace{(a_{ij_i} - \Delta_{j_i})}_{\text{const. for a row}}\ ]$$

Second Improvement

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

$$\downarrow$$

$$\delta = min_{i \in R}[\ min_{s \in S}[(a_{is} - \Delta_s)] - \underbrace{(a_{ij_i} - \Delta_{j_i})}_{\text{const. for a row}}\ ]$$

$$\downarrow$$

Second Improvement

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

$$\downarrow$$

$$\delta = min_{i \in R}[\ min_{s \in S}[(a_{is} - \Delta_s)] - \underbrace{(a_{ij_i} - \Delta_{j_i})}_{\text{const. for a row}}\ ]$$

$$\downarrow$$

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

Second Improvement

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

$$\downarrow$$

$$\delta = min_{i \in R}[\ min_{s \in S}[(a_{is} - \Delta_s)] - \underbrace{(a_{ij_i} - \Delta_{j_i})}_{\text{const. for a row}}\ ]$$

$$\downarrow$$

$$\delta = min_{i \in R}[\ \underbrace{min_{s \in S}(a_{is} - \Delta_s)}_{q_i}\ - (a_{ij_i} - \Delta_{j_i})]$$

Second Improvement

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

$$\downarrow$$

$$\delta = min_{i \in R}[\ min_{s \in S}[(a_{is} - \Delta_s)] - \underbrace{(a_{ij_i} - \Delta_{j_i})}_{\text{const. for a row}}\ ]$$

$$\downarrow$$

$$\delta = min_{i \in R}[\ \underbrace{min_{s \in S}(a_{is} - \Delta_s)}_{q_i}\ - (a_{ij_i} - \Delta_{j_i})]$$

- At the begining of an iteration compute the column vector q_i

In each succeeding step of phase 1:

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

Second improvement

- At the begining 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?

Second improvement

- At the begining 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?

$$\underbrace{n \times [\underbrace{O(n)}_{phase0} + \underbrace{n \times O(n)}_{phase1} + \underbrace{n}_{phase2}]}_{\text{one iteration}}$$

Second improvement

- At the begining 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?

$$O(n^3)$$