A SOLVING METHOD FOR ASSIGNMENT PROBLEMS USING IMBEDDING PRINCIPLE

YASUKI SEKIGUCHI

1. INTRODUCTION

Assignment problems are some of the most familiar and fundamental problems in the mathematical programming field. One of the more famous and effective solving methods is the "HUNGARIAN METHOD"[3),4)], the method based on the König-Egervary theorem. The Hungarian method handles the whole cost matrix ($n \times n$ cost matrix) through its solution process, but the König-Egervary theorem is not restricted only to square matrices.

The fact that the theorem is valid for rectangular matrices also gives a hint that there may be another solution where an original cost matrix is divided into a set of rectangular submatrices, i. e. $n \times 1, n \times 2, \cdots, n \times (n-1)$, $n \times n$.

By going along with this rather simple and evident idea, a new solving method can be considered. The original problem with a $n \times n$ cost matrix is imbedded in a set of rectangular assignment problems with $n \times 1, n \times 2, \cdots$, $n \times (n-1)$ and $n \times n$ cost matrices. Therefore the larger the size of the original problem, the greater the computation-saving in applying a new method as compared to the conventional Hungarian method. The computing effort needed for the new method is about 50% of the conventional one if the problem size is large enough. Moreover, the proposed method has several merits from the applicational point of view.

The Hungarian method for (m, n) assignment problems is derived from the solving method of the Hitchcock transportation problems (This is a generalization of the original Hungarian method)[1)]. The new solving method using the imbedding principle is then explained in section 3. In section 4 and 5 there are discussions on the computations.

2. PROBLEM STATEMENT AND THE CONVENTIONAL SOLUTION PROCESS

An assignment problem is usually stated as follows; "There are n-men and n-machines. If i-th man is assigned to j-th machine, it causes a loss of $c_{i,j}$. Every man can be assigned to only one machine and every machine has to be assigned to one person. What assignment minimizes the total loss?

Here, we expand this situation to m-men-n-machines case ($m \geqq n$). Mathematically, this problem is defined as $P(m, n)$.

$$P(m,n) \qquad \text{minimize} \qquad z=\sum_{i,j} c_{i,j}x_{i,j}$$

$$\text{s. t. } \sum_{j} x_{i,j} \leqq 1 \tag{1}$$

$$\sum_{i} x_{i,j} \geqq 1 \tag{2}$$

$$c_{i,j} \geqq 0, \ x_{i,j} \geqq 0$$

$$i=1, 2, \cdots, m, \ j=1, 2, \cdots, n$$

Inequality (2) in $P(m, n)$ can be altered by equality because of non-negativity of $c_{i,j}$'s. $P(m, n)$ is a special case of the "Hitchcock transportation problem" (HTP, hereafter) as it is easily understood. A solution process of HTP is also effective on $P(m, n)$. One of the most effective solutions is basicly a primal-dual method though it is not the simplex method.

Let Π_i, Π_j be dual variables corresponding to constraints (1) and (2), respectively. The dual problem of $P(m, n)$ is;

$$\bar{P}(m,n) \qquad \text{maximize} \qquad \xi=-\sum_{i}\Pi_i+\sum_{j}\Pi_j$$

$$\text{s. t. } -\Pi_i+\Pi_j \leqq c_{i,j} \tag{3}$$

$$\Pi_i, \ \Pi_j \geqq 0$$

Assume (Π_i, Π_j) be a dual feasible solution for $\bar{P}(m, n)$, then the restricted primal problem for $P(m, n)$ in the primal-dual simplex method is equivalent to (RP 1); (RP 1) is a special type of maximal flow problems and can be efficiently solved by a labeling process.

$$\text{(RP 1)} \qquad \text{maximize} \qquad f=\sum_{i,j} x_{i,j}$$

$$\text{s. t. } \sum_{j} x_{i,j} \leqq 1 \tag{4}$$

$$\sum_{i} x_{i,j} \leqq 1 \tag{5}$$

$$x_{i,j} \begin{cases} \geqq 0 \text{ if } -\Pi_i+\Pi_j=c_{i,j} \\ =0 \text{ otherwise} \end{cases} \tag{6}$$

The solution procedure emphasized above is the Hungarian method for the HTP's[1]. For the general HTP's, this procedure starts with a dual feasible solution (Π_i, Π_j), solves a maximal flow problem defined by the dual solution and, if the given maximal flow is not a primal feasible solution for the HTP, the current dual solution is improved through minimal line covering of admissible cells (a cell $c_{i,j}$ is called admissible if $-\Pi_i+\Pi_j=c_{i,j}$). This process is repeated until a maximal flow corresponding to a dual feasible solution becomes a primal feasible solution. The generalized assignment problem defined above is a special HTP and the algorithm can be simplified using the specialities such as;

(a) all the constant terms in the constraints are equal to 1.

(b) all the integral $x_{i,j}$'s in a optimal solution are equal to 1 or 0.

In the simplified algorithm stated below, Step 1 is an initial setting of dual variables and etc. Steps two through seven give a simplified labeling procedure for maximal flow problems (i. e. RP 1) and Step 8 improves the current dual solution when the labeling procedure failed to find a solution satisfying the constraints of $P(m, n)$, and a new maximal flow problem emerges. When the termination happens, set $x_{i,j}=1$ if $c_{i,j}$ is enclosed with a circle, otherwise set $x_{i,j}=0$, then $X=|x_{i,j}|$ is the optimal solution and the current value of z is the value of the objective function z for the optimal X.

[Algorithm for HTP]*[1]

Step 1: Set $\Pi_i=0$, $\Pi_j=\min\limits_i c_{i,j}$, $z=\sum\limits_j \Pi_j$

Step 2: Search for admissible cells $c_{i,j}$ in order of $c_{1,1}, c_{1,2} \cdots, c_{1,n}, c_{2,1}, \cdots, c_{2,n}, \cdots, c_{m,n}$ and let admissible $c_{i,j}$'s be enclosed with circles if there is no enclosed cell in column j.

Step 3: If there is an enclosed cell in every column, terminate the algorithm. Otherwise, (erase all the old labels and) go to Step 4.

Step 4: Assign label $(-)$ to every row i which has no enclosed cell.

Step 5: Select a (newly) labeled row, say row i, scan it for all unlabeled columns j such that cell $c_{i,j}$ is admissible, and label these columns (i). Repeat this process until (1) the newly labeled rows have been all scaned or (2) a column j, which has no enclosed cell, is labeled. In case (2), go to Step 7. In case (1), if there are some newly labeled columns, go to Step 6, otherwise go to Step 8.

Step 6: Select a newly labeled column, say column j, scan it for an enclosed cell $c_{i,j}$ where the row i has not been labeled yet, and label these rows (j). Repeat this process until the newly labeled columns have all been scaned. If there are some newly labeled rows, then go to Step 5, otherwise go to Step 8.

Step 7: Assume row i is the current scaned row and column j is the one which caused the jump to Step 7 ($r(j)$ is the label of column j and $c(i)$ is the one of row i). (1) Let $c_{r(j),j}$ be enclosed with a circle and let i be equal to $r(j)$. (2) Erase the circle enclosing $c_{i,c(i)}$ and let j be equal to $c(i)$. Repeat the process (1) and (2) until $c(i)=(-)$ happens, then go to Step 3.

*[1] (The max-flow min-cut theorem is used in the steps from 4 to 7, i. e. in the labeling process. On the other hand, the Hungarian method is said to be based on the König-Egervary's theorem. In order to see this in the above algorithm, notice that $|\bar{I}|+|J|$ =(the maximum number of independent admissible cells under Π_i, Π_j))

Step 8: Let I and J be the index sets of labeled rows and columns, respectively. Define new dual variables by;

$$\Pi'_i = \begin{cases} \Pi_i & \text{if } i \in I \\ \Pi_i + \delta & \text{if } i \in \bar{I} = \{1, 2, \cdots, \mathrm{m}\} - I \end{cases} \tag{7}$$

$$\Pi'_j = \begin{cases} \Pi_j & \text{if } j \in J \\ \Pi_j + \delta & \text{if } j \in \bar{J} = \{1, 2, \cdots, n\} - J \end{cases} \tag{8}$$

$$z = -\sum_i \Pi'_i + \sum_j \Pi'_j = z + \delta \cdot (-|\bar{I}| + |\bar{J}|) \tag{9}$$

where

$$\delta = \min_{i \in I, j \in \bar{J}} \{c_{i,j} + \Pi_i - \Pi_j\} \tag{10}$$

Go back to Step 4 and repeat Step 4 through 6 with new admissible cells defined by eqs. (7) through (10).

Let a matrix $C = |c_{i,j}|$ and define C' by eq. (11),

$$C' = |c'_{i,j}| = |c_{i,j} + \Pi_i - \Pi_j| \tag{11}$$

then

$$c'_{i,j} \begin{cases} = 0 & \text{if cell } c_{i,j} \text{ is an admissible cell} \\ > 0 & \text{otherwise} \end{cases} \tag{12}$$

If C' is used in the steps 2 through 8, the statements of each step is valid only if the word "admissible cell" is changed to "0-cell" (see (12)) and only if the eq. (10) is changed to (10)$'$.

$$\delta = \min_{i \in I, j \in \bar{J}} c'_{i,j} \tag{10$'$}$$

Let C'' be the matrix C' corresponding to Π'_i, Π'_j in Step 8. C'' can be generated by directly applying eq. (11) for C, Π'_i, Π'_j. But if C' corresponding to Π_i, Π_j is given, the two-step procedure below is more efficient.

(i) add δ to all cells on row i if $i \in \bar{I}$

(ii) subtract δ from all cells on column j if $j \in \bar{J}$

Thus further simplification of the algorithm is possible and the resulting algorithm is shown in Figure 1.

The algorithm shown in Fig. 1 is slightly different from the usual one for (n, n) assignment problems. These differences are caused by the fact that $m \geqq n$ and $m = n$ is not always true (this means that all columns have at least one 0-element but there may be some rows having no 0-element when the algorithm is terminated). Firstly, only column minimums are concerned in block 1. Secondly, the labeling process is started with searching columns for 0-elements and subsequent procedures are revised along this line.

3. PROPOSED ALGORITHM

The proposed algorithm uses almost the same method as one stated in Section 2. Only one difference is that the proposed one solves a sequence of subproblems (they are also assignment problems) derived from the original problem, that is, the original problem is embedded in a sequence of easier problems.

I'd like to discuss its validity first. Assume there is a (m, n) assignment problem $P(m, n)$ and its optimal solution $X^*=|x_{i,j}|$. Also assume the cost matrix is $C=|c_{i,j}|$ and C_k is a submatrix made of the first k columns of C. Let $P(m, k)$ be a subproblem of $P(m, n)$ concerning to a cost matrix C_k. Define X_k in the same way as C_k. X_k^* has to be a feasible solution of $P(m, k)$ for any k $(1 \leq k \leq m)$, though it may not be an optimal solution. This fact suggests an algorithm where an optimal solution for $P(m, k-1)$ can be used as an initial solution for $P(m, k)$ (see block 1 in Fig. 1).

To see how the solution process of $P(m, k)$ is initiated, define as follows.

δ_i: The sum of all δ which have been added to row i in the solution process of $P(m, 1)$, $P(m, 2)$, $\cdots$, $P(m, k-1)$.

C'_{k-1}: The resulting cost matrix when the solution process of $P(m, k-1)$ is terminated.

C'_k: A $m \times k$ matrix where the first $k-1$ columns are those of C'_{k-1} and the k-th column made of $c'_{i,k}$.

$$c'_{i,k} = c_{i,k} + \delta_i$$

(Π_i^*, Π_j^*): Π_i^* is the optimal Π_i for $\bar{P}(m, k-1)$ $(i=1, 2, \cdots, m)$, Π_j^* is the optimal Π_j for $\bar{P}(m, k-1)$ if $j=1, 2, \cdots, k-1$ and $\Pi_j^* = \min\limits_i c'_{i,k}$ if $j=k$.

Then, (Π_i^*, Π_j^*) is a feasible dual solution for $P(m, k)$ because

(1) $c_{i,j} + \Pi_i^* - \Pi_j^* = c'_{i,j} \geqq 0$ for $i=1, 2, \cdots$, m, $j=1, 2, \cdots, k-1$ by the optimality of (Π_i^*, Π_j^*).

(2) $c_{i,k} + \Pi_i^* - \Pi_k^* = c'_{i,k} - \min\limits_i c'_{i,k} \geqq 0$. To see this, notice that $\Pi_i^* = \delta_i$ for $i=1, 2, \cdots, m$ because Π_i is the sum of δ by eq. (7). And

$$z_k = -\sum_i \Pi_i^* + \sum_j \Pi_j^* = z_{k-1} + \min_i c'_{i,k} \tag{13}$$

$$C''_k = |c''_{i,j}|$$

where

$$c''_{i,j} = \begin{cases} c'_{i,j} \text{ for } i=1, 2, \cdots, m,\ j=1, 2, \cdots, k-1 \\ c'_{i,k} - \min\limits_i C'_{i,k} \text{ for } i=1, 2, \cdots, m,\ j=k \end{cases}$$

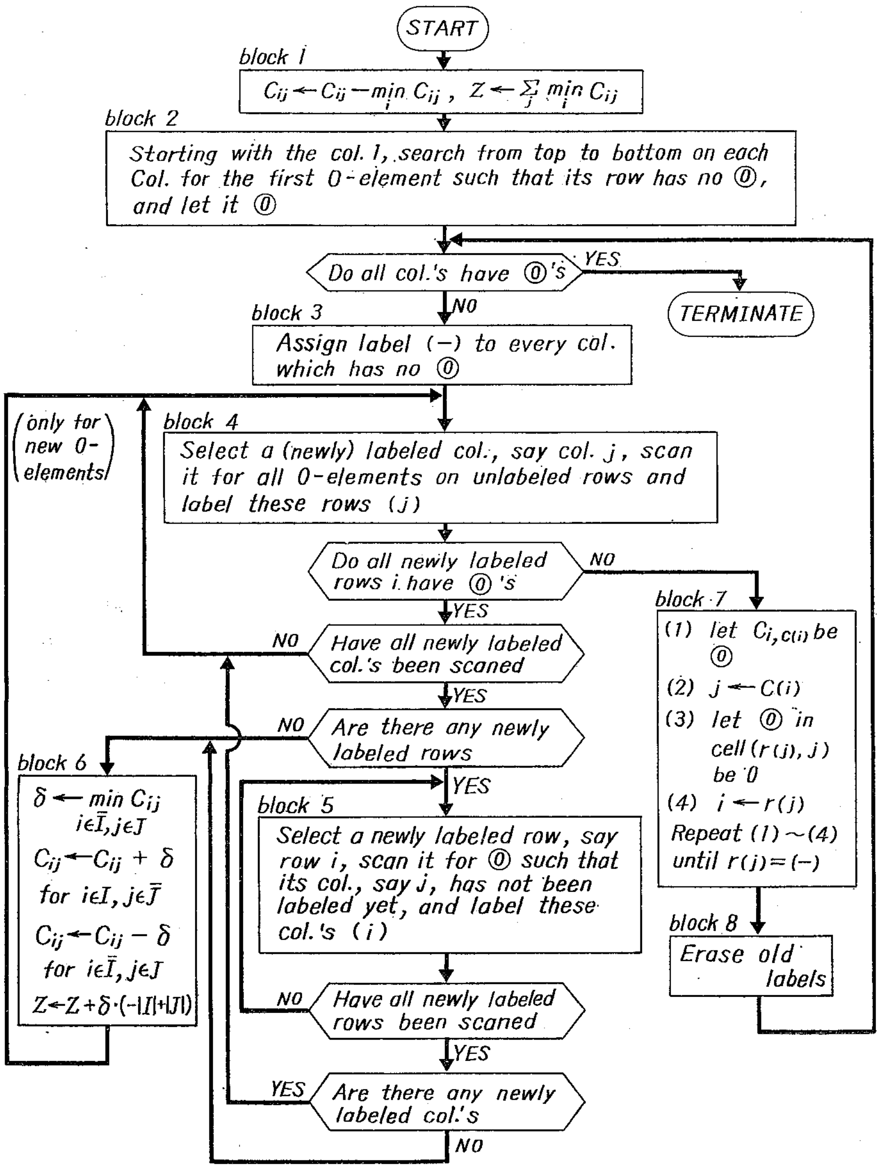
can be used as an initial z and C in block 1 of Fig. 1.

An outline of the proposed algorithm is the next two-step procedure repeated from $k=1$ to $k=n$.

Step I; Solve $P(m, k)$ by the algorithm in Fig. 1

Step II; Make z_{k+1} and C''_{k+1}, and if $k \neq n$, let $k=k+1$ and go to Step I.

Because the optimal solution of $P(m, k-1)$ is used as an initial solution in Step I, the k-th column is the only one which has not an enclosed-with-circle 0-element. Therefore, block 2 is not needed, block 3 is far simple,

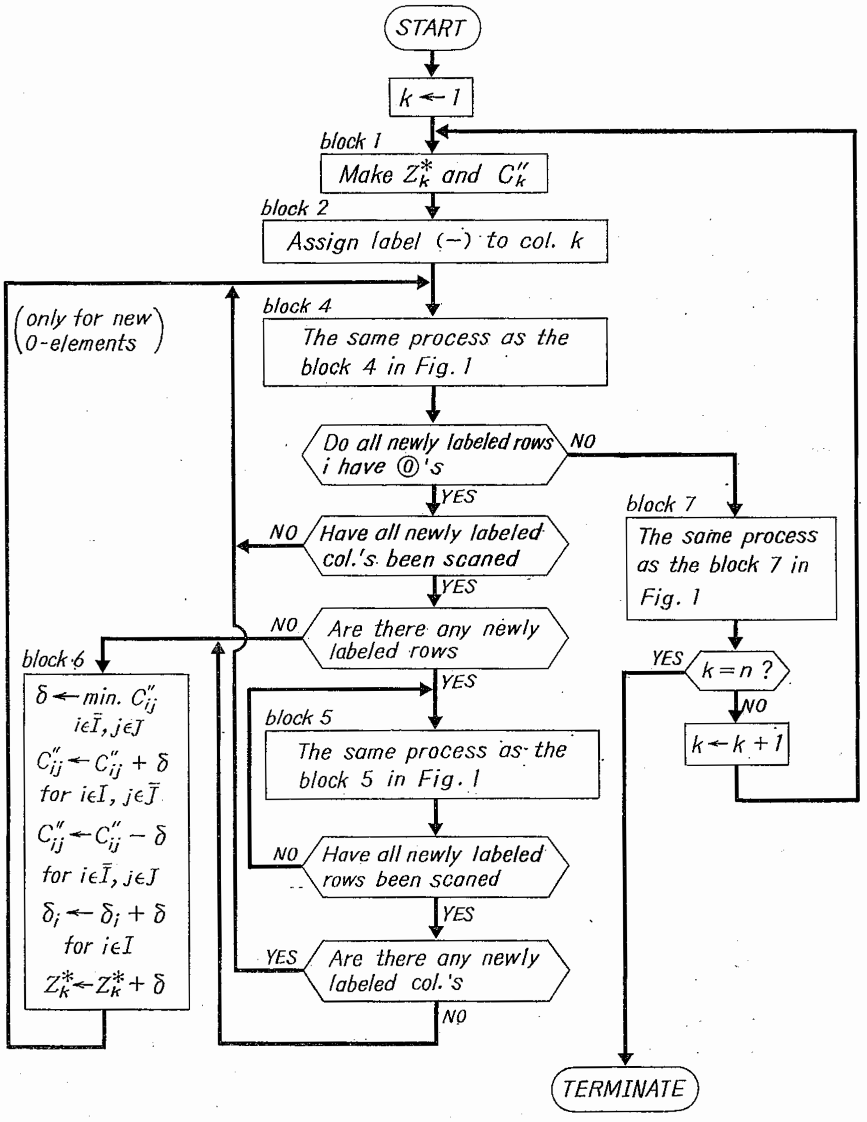


START
block 1
$C_{ij} \leftarrow C_{ij} - \min_i C_{ij}$, $Z \leftarrow \sum_j \min_i C_{ij}$
block 2
Starting with the col. 1, search from top to bottom on each Col. for the first 0-element such that its row has no ⓪, and let it ⓪
Do all col.'s have ⓪'s
YES
TERMINATE
NO
block 3
Assign label (−) to every col. which has no ⓪
only for new 0-elements
block 4
Select a (newly) labeled col., say col. j, scan it for all 0-elements on unlabeled rows and label these rows (j)
Do all newly labeled rows i have ⓪'s
NO
YES
Have all newly labeled col.'s been scaned
NO
YES
Are there any newly labeled rows
NO
YES
block 5
Select a newly labeled row, say row i, scan it for ⓪ such that its col., say j, has not been labeled yet, and label these col.'s (i)
Have all newly labeled rows been scaned
NO
YES
Are there any newly labeled col.'s
YES
NO
block 6
$\delta \leftarrow \min_{i \in \bar{I}, j \in J} C_{ij}$
$C_{ij} \leftarrow C_{ij} + \delta$ for $i \in I, j \in \bar{J}$
$C_{ij} \leftarrow C_{ij} - \delta$ for $i \in \bar{I}, j \in J$
$Z \leftarrow Z + \delta \cdot (-|I| + |J|)$
block 7
(1) let $C_{i,c(i)}$ be ⓪
(2) $j \leftarrow C(i)$
(3) let ⓪ in cell $(r(j), j)$ be 0
(4) $i \leftarrow r(j)$
Repeat (1)~(4) until $r(j) = (-)$
block 8
Erase old labels

Fig. 1. The conventional algorithm for the (m, n) assignment problem

$-|I|+|J|=1$ for every k in block 6 (notice that $|I|+|\bar{J}|=k-1$ for every k), and when a jump to block 7 first happens is the time of termination of Step I, in the algorithm of Fig. 1. A flow chart of the algorithm is shown in Fig. 2.

Fig. 3 shows an example comparing the conventional and proposed algorithms. The problem is taken from Little et al[2] (but the last column


START
$k \leftarrow 1$
block 1
Make Z_k^* and C_k''
block 2
Assign label (−) to col. k
(only for new 0-elements)
block 4
The same process as the block 4 in Fig. 1
Do all newly labeled rows i have ⓪'s
NO
YES
Have all newly labeled col.'s been scaned
NO
YES
Are there any newly labeled rows
NO
YES
block 5
The same process as the block 5 in Fig. 1
Have all newly labeled rows been scaned
NO
YES
Are there any newly labeled col.'s
YES
NO
block 6
$\delta \leftarrow \min_{i\in\bar{I}, j\in J} C_{ij}''$
$C_{ij}'' \leftarrow C_{ij}'' + \delta$ for $i\in I, j\in\bar{J}$
$C_{ij}'' \leftarrow C_{ij}'' - \delta$ for $i\in\bar{I}, j\in J$
$\delta_i \leftarrow \delta_i + \delta$ for $i\in I$
$Z_k^* \leftarrow Z_k^* + \delta$
block 7
The same process as the block 7 in Fig. 1
$k=n$?
YES
NO
$k \leftarrow k+1$
TERMINATE

note ; $\bar{J} = \{1, 2, \cdots, k\} - J$

Fig. 2. The proposed algorithm for the (m, n) assignment problem

is omitted). Fig. 3 (a) shows the problem and Fig. 3 (b)-(g) show the solution process by the conventional algorithm in Fig. 1. Fig. 3 (h)-(q) show the process by the proposed one. The number of comparisons needed in block 1 and block 6 of Fig. 1 was sixty-three, and in figure 2, in block 1 and 6 seventy-three (see section 4 for detail). In this case, computational effort increased by the new algorithm is about 16%.

i \ j	1	2	3	4	5
1	+∞	27	43	16	30
2	7	+∞	16	1	30
3	20	13	+∞	35	5
4	21	16	25	+∞	18
5	12	46	27	48	+∞
6	23	5	5	9	5

(a) an example problem [Little, et al[2)]]

i \ j	1	2	3	4	5
1	+∞	27	43	16	30
2	7	+∞	16	1	30
3	20	13	+∞	35	5
4	21	16	25	+∞	18
5	12	46	27	48	+∞
6	23	5	5	9	5
$\min_i C_{i,j}$	7	5	5	1	5

γ=5×5=25
Z=23
(b) block 1

i \ j	1	2	3	4	5	c(i)
1	+∞	22	38	15	25	
~~2~~	~~⓪~~	~~+∞~~	~~11~~	~~0~~	~~25~~	~~(4)~~
3	13	8	+∞	34	⓪	
4	14	11	20	+∞	13	
5	5^{Δ}	41	22	47	+∞	
~~6~~	~~16~~	~~⓪~~	~~0~~	~~8~~	~~0~~	~~(3)~~
γ(j)	(2)	(6)	(-)	(-)		

γ=25+4×4=41
Z=23+5(-2+4)=33
(c) block 2~5, 6

i \ j	1	2	3	4	5	c(i)
1	+∞	17	33	10	25	
2	⓪	+∞	11	⓪	30	(4)
3	8	3	+∞	29	⓪	
4	9	6	15	+∞	13	
5	⓪	36	17	42	+∞	(1)
6	16	⓪	0	8	5	(3)
γ(j)	(2)	(6)	(-)	(-)		

γ=41
Z=33
(d) block 4, 7, 8

i \ j	1	2	3	4	5	c(i)
1	+∞	17	33	10	25	
2	0	+∞	11	⓪	30	
3	8	3^{Δ}	+∞	29	⓪	
4	9	6	15	+∞	13	
5	⓪	36	17	42	+∞	
~~6~~	~~16~~	~~⓪~~	~~0~~	~~8~~	~~5~~	~~(3)~~
γ(j)		(6)	(-)			

γ=41+5×2=51
Z=33+3(-1+2)=36
(e) block 2~5, 6

i \ j	1	2	3	4	5	c(i)
1	+∞	14	30	10	25	
2	0	+∞	8	⓪	30	
~~3~~	~~8~~	~~0~~	~~+∞~~	~~29~~	~~⓪~~	~~(2)~~
4	9	3^{Δ}	12	+∞	13	
5	⓪	33	14	42	+∞	
~~6~~	~~19~~	~~⓪~~	~~0~~	~~11~~	~~8~~	~~(3)~~
γ(j)		(6)	(-)		(3)	

γ=51+4×3=63
Z=36+3(-2+3)=39
(f) block 4, 5, 6

i \ j	1	2	3	4	5	c(i)
1	+∞	11	27	10	22	
2	0	+∞	5	⓪	27	
3	11	0	+∞	32	⓪	(2)
4	9	⓪	9	+∞	10	(2)
5	⓪	30	11	39	+∞	
6	22	⓪	⓪	14	8	(3)
γ(j)		(6)	(-)		(3)	

γ=63
Z=39
(g) block 4, 7, 8

Fig. 3. An example of the conventional and proposed algorithms
(a) an example problem
(b)~(g) solution process by the conventional algorithm

i\j	1
1	+∞
2	7
3	20
4	21
5	12
6	23

$\gamma_p=5$
$Z=7$
(h) block 1

i\j	1	
1	+∞	
2	⓪	(1)
3	13	
4	14	
5	5	
6	16	
	(−)	

$\gamma_p=5$
$Z=7$
(i) block 2,4,7

i\j	1	2
1	+∞	27
2	⓪	+∞
3	13	13
4	14	16
5	5	46
6	16	5

$\gamma_p=5+5=10$
$Z=7+5=12$
(j) block 1

i\j	1	2	3	C(i)
1	+∞	22	43	
2	⓪	+∞	16	
3	13	8	+∞	
4	14	11	25	
5	5	41	27	
6	16	⓪	5	(2)
γ(j)		(−)		

$\gamma_p=10+5=15$
$Z=12+5=17$
(k) block 2,4,7,1

i\j	1	2	3	C(i)	δ_i
1	+∞	22	38		
2	⓪	+∞	11		
3	13	8^{Δ}	+∞		
4	14	11	20		
5	5	41	22		
~~6~~	~~16~~	~~⓪~~	~~0~~	~~(3)~~	8
γ(j)		(6)	(−)		

$\gamma_p=15+2\times5+1=26$
$Z=17+8=25$
(l) block 2,4,5,6

i\j	1	2	3	4	C(i)	δ_i
1	+∞	14	30	16		
2	⓪	+∞	3	1		
3	13	⓪	+∞	35	(2)	
4	14	3	12	+∞		
5	5	33	14	48		
6	24	$\cancel{⓪}$	⓪	17	(3)	8
γ(j)		(6)	(−)			

$\gamma_p=26+6+5=37$
$Z=25+1=26$
(m) block 4, 7, 1

i\j	1	2	3	4	C(i)	δ_i
1	+∞	14	30	15		
~~2~~	~~⓪~~	~~+∞~~	~~3~~	~~0~~	~~(4)~~	5
3	13	⓪	+∞	34		
4	14	3	12	+∞		
5	5^{Δ}	33	14	47		
6	24	0	⓪	16		8
γ(j)	(2)			(−)		

$\gamma_p=37+2\times5+1=48$
$Z=26+5=31$
(n) block 2, 4, 5, 6

i\j	1	2	3	4	5	C(i)	δ_i
1	+∞	14	30	10	30		
2	$\cancel{⓪}$	+∞	8	⓪	35	(4)	5
3	8	⓪	+∞	29	5		
4	9	3	12	+∞	18		
5	⓪	33	14	42	+∞	(1)	
6	19	0	⓪	11	13		8
γ(j)	(2)				(−)		

$\gamma_p=48+6+5=59$
$Z=31+5=36$
(o) block 4, 7, 1

i\j	1	2	3	4	5	C(i)	δ_i
1	+∞	14	30	10	25		
2	0	+∞	8	⓪	30		5
~~3~~	~~8~~	~~⓪~~	~~+∞~~	~~29~~	~~0~~	~~(5)~~	3
4	9	3^{Δ}	12	+∞	13		
5	⓪	33	14	42	+∞		
~~6~~	~~19~~	~~0~~	~~⓪~~	~~11~~	~~8~~	~~(2)~~	11
γ(j)		(3)	(6)		(−)		

$\gamma_p=59+4\times3+2=73$
$Z=36+3=39$
(p) block 2, 4, 5, 6

i\j	1	2	3	4	5	C(i)
1	+∞	11	27	10	22	
2	0	+∞	5	⓪	27	
3	11	$\cancel{⓪}$	+∞	32	⓪	(5)
4	9	⓪	9	+∞	10	(2)
5	⓪	30	11	42	+∞	
6	22	0	⓪	14	8	(2)
γ(j)		(3)	(6)		(−)	

$\gamma_p=73$
$Z=39$
(q) block 4, 7

Fig. 3. Continued
(h)~(q) solution process by the proposed algorithm

4. THEORETICAL COMPARISON OF COMPUTATIONAL EFFORTS

The proposed algorithm solves a sequence of assignment problems while the conventional one does only one problem. It intuitively seems that the former needs more computational effort than the latter. This was true at least for the example in Fig. 3. Nevertheless, it will be shown here that less computations are strongly expected for the proposed algorithm than for the conventional one.

Estimating exactly the computational effort needed for an algorithm is not easy even when its computer program has already been completed. So does this case. Therefore, the total number of comparisons which are needed for finding out the minimum elements on each row and each column (i. e. in block 1 and 6 of Fig. 1 and Fig. 2) is used as a measure of computational efforts. The reason is that the computations needed during one cycle of labeling process (i. e. until a jump to block 6 or 7 happens at first) may be made less than that in block 6 by using bit manipulation and that additions and subtractions needed are proportional to comparisons, and finally that the total computational effort needed is heavily dependent on the number of repetitions of block 6.

How many times block 6 is repeated depends on the data given to a certain problem. But its maximum can be estimated as follows.

For the conventional algorithm, comparisons needed in block 1 are

$$n(m-1) \text{ times}\,. \tag{15}$$

Next, assume that k⓪'s are made at block 2. $|I|=1$ for the first labeling process in the worst case, then at block 6, $c_{i,j}$ which determines δ (i. e. a new 0-element) may be on the row of a certain ⓪, in this case $|I|$ increases just by 1 during the second labeling process. If this process succeeds until $|I|$ becomes equal to k, the $k+1$st ⓪ is made after and block 6 is repeated k times (see, for example, Fig. 5). Comparisons needed at block 6 during each repetition is shown in Fig. 4.

In the worst case, the total number of comparisons needed at block 6 is;

$$\sum_{i=1}^{k}(m-i)(n-k+i) = km(n-k)+\sum_{i=1}^{k-1} i(k-i)+\sum_{i=1}^{k} i(m-n) \tag{16}$$

The above cases hardly succeed from the $k+1$st ⓪ to the n-th ⓪. But

repetition	$\lvert I\rvert$	$\lvert J\rvert$	$\lvert I\rvert\cdot\lvert J\rvert$ at block 6
1st	1	$n-k+1$	$(m-1)\cdot(n-k+1)$
2nd	2	$n-k+2$	$(m-2)\cdot(n-k+2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
i-th	i	$n-k+i$	$(m-i)\cdot(n-k+i)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
k-th	k	n	$(m-k)\cdot n$

Fig. 4. Number of comparisons needed at block 6; the worst case of the conventional algorithm

$i \backslash j$	1	2	3	4	5
1	81	10^{Δ}	61	10^{Δ}	10^{Δ}
2	10^{Δ}	71	62	20	11
3	82	72	10^{Δ}	30	12
4	83	73	63	40	13
5	84	74	64	50	14
$\min_i C_{ij}$	10	10	10	10	10

$\gamma=5\times4=20,\ Z=50$

(a) an example problem

$i \backslash j$	1	2	3	4	5	
~~1~~	~~71~~	~~⓪~~	~~51~~	~~0~~	~~0~~	~~(4)~~
2	⓪	61	52	10	1^{Δ}	
3	72	62	⓪	20	2	
4	73	63	53	30	3	
5	74	64	54	40	4	
	(1)			(−)	(−)	

$\gamma=20+12=32$

(b) $Z=50+1(-1+3)=52$

$i \backslash j$	1	2	3	4	5	
~~1~~	~~72~~	~~⓪~~	~~52~~	~~0~~	~~0~~	~~(4)~~
~~2~~	~~⓪~~	~~60~~	~~52~~	~~9~~	~~0~~	~~(5)~~
3	72	61	⓪	19	1^{Δ}	
4	73	62	53	29	2	
5	74	63	54	39	3	
	(2)	(1)		(−)	(−)	

$\gamma=32+12=44$

(C) $Z=52+1(-2+4)=54$

$i \backslash j$	1	2	3	4	5	
~~1~~	~~72~~	~~⓪~~	~~53~~	~~0~~	~~0~~	~~(4)~~
~~2~~	~~⓪~~	~~60~~	~~53~~	~~9~~	~~0~~	~~(5)~~
~~3~~	~~71~~	~~60~~	~~⓪~~	~~18~~	~~0~~	~~(5)~~
4	72	61	53	28	1^{Δ}	
5	73	62	54	38	2	
	(2)	(1)	(3)	(−)	(−)	

$\gamma=44+10=54$

(d) $Z=54+1(-3+5)=56$

$i \backslash j$	1	2	3	4	5	
1	72	⓪	53	0	0	(4)
2	⓪	60	53	9	0	(5)
3	71	60	⓪	18	0	(5)
4	71	60	52	27	⓪	(5)
5	72	61	53	37	1	
	(2)	(1)	(3)	(−)	(−)	

$\gamma=54$

(e) $Z=56$

$i \backslash j$	1	2	3	4	5	
~~1~~	~~72~~	~~⓪~~	~~53~~	~~0~~	~~0~~	~~(4)~~
2	⓪	60	53	9^{Δ}	0	
3	71	60	⓪	18	0	
4	71	60	52	27	⓪	
5	72	61	53	37	1	
		(1)		(−)		

$\gamma=54+8=62$

(f) $Z=56+9(-1+2)=65$

$i \backslash j$	1	2	3	4	5	
~~1~~	~~81~~	~~⓪~~	~~62~~	~~0~~	~~9~~	~~(4)~~
~~2~~	~~⓪~~	~~51~~	~~53~~	~~0~~	~~0~~	~~(4)~~
3	71	51	⓪	9^{Δ}	0	
4	71	51	52	18	⓪	
5	72	52	53	28	1	
	(2)	(1)		(−)		

$\gamma=62+9=71$

(g) $Z=65+9(-2+3)=74$

$i \backslash j$	1	2	3	4	5	
~~1~~	~~81~~	~~⓪~~	~~71~~	~~0~~	~~18~~	~~(4)~~
~~2~~	~~⓪~~	~~51~~	~~62~~	~~0~~	~~9~~	~~(4)~~
~~3~~	~~62~~	~~42~~	~~⓪~~	~~0~~	~~0~~	~~(4)~~
4	62	42	52	9^{Δ}	⓪	
5	63	43	53	19	1	
	(2)	(1)	(3)	(−)		

$\gamma=71+8=79$

(h) $Z=74+9(-3+4)=83$

$i \backslash j$	1	2	3	4	5	
~~1~~	~~81~~	~~⓪~~	~~71~~	~~0~~	~~27~~	~~(4)~~
~~2~~	~~⓪~~	~~51~~	~~62~~	~~0~~	~~18~~	~~(4)~~
~~3~~	~~62~~	~~42~~	~~⓪~~	~~0~~	~~9~~	~~(4)~~
~~4~~	~~53~~	~~33~~	~~43~~	~~0~~	~~⓪~~	~~(4)~~
5	54	34	44	10	1^{Δ}	
	(2)	(1)	(3)	(−)	(4)	

$\gamma=79+5=84$

(i) $Z=83+1(-4+5)=84$

$i \backslash j$	1	2	3	4	5	
1	81	⓪	71	0	27	(4)
2	⓪	51	62	0	18	(4)
3	62	42	⓪	0	9	(4)
4	53	33	43	⓪	⓪	(4)
5	53	33	43	9	⓪	(5)
	(2)	(1)	(3)	(−)	(4)	

$\gamma=84$

(j) $Z=84$

Fig. 5. An example of the worst case; the conventional algorithm, $m=n=5$, $k=3$, $\gamma(3)=84$

assuming that it happens, the maximum number $\gamma(k)$ $(1 \leqq k \leqq n-1)$ of the comparisons are estimated (an example of the worst case is given in Fig. 5).

$$\begin{aligned}
\gamma(k) &= n(m-1)+\sum_{p=k}^{n-1}\left\{pm(n-p)+\sum_{i=1}^{p-1} i(p-i)+\sum_{i=1}^{p} i(m-n)\right\} \\
&= n(m-1)+\sum_{p=k}^{n-1}(pmn-p^2m) \\
&\quad +\sum_{p=k}^{n-1}\left\{\frac{p^2(p-1)}{2}-\frac{(p-1)p(2p-1)}{6}\right\}+\sum_{p=k}^{n-1}(m-n)\frac{p(p+1)}{2} \\
&= n(m-1)+\frac{n}{24}(n-1)(8mn-3n^2-4m+7n-2) \\
&\quad -\frac{k}{24}(k-1)\left\{k^2-k(4m+4n+1)+12mn+8m-4n-2\right\} \qquad (17)
\end{aligned}$$

Next, the maximum comparison number for the proposed one is estimated. Assume that the $P(m, k)$ has solved and we are going to solve $P(m, k+1)$. Firstly, δ_i has to be added to $c_{i,k+1}$, then the minimum on the column $k+1$ is discovered and is subtracted from each $c'_{i,k+1}$. This completes C''_{k+1}. Adding δ_i to the $k+1$st column is an additional operation caused by adopting the iterative procedure and we count this addition also, though comparison is the only operation noticed now. Therefore, the total operation number needed at block 1 in Fig. 2 is;

$$2m-1 \qquad (18)$$

Doing the same thing as for the conventional one, comparisons needed during each iteration of block 6 in Fig. 2 are obtained and shown in Fig. 6.

In Fig. 6, additions needed to make δ_i's are counted because of the same reasons as additions in block 1. The total number of operations needed at block 1 and 6 when the $k+1$st ⓞ is made are;

$$\begin{aligned}
&\sum_{i=1}^{k}(m-i)(i+1)+\sum_{i=1}^{k} i+2m-1 \\
&\quad =\frac{m(k^2+3k+4)}{2}-\frac{k(k+1)(2k+1)}{6}-1 \qquad (19)
\end{aligned}$$

repetition	$\lvert I\rvert$	$\lvert J\rvert$	$\lvert \bar{I}\rvert\cdot\lvert J\rvert$ at block 6	δ_i
1st	1	2	$(m-1)\cdot 2$	1
2nd	2	3	$(m-2)\cdot 3$	2
⋮	⋮	⋮	⋮	⋮
i-th	i	$i+1$	$(m-i)\cdot(i+1)$	i
⋮	⋮	⋮	⋮	⋮
k-th	k	$k+1$	$(m-k)\cdot(k+1)$	k

Fig. 6. Number of comparisons needed at block 6; the worst case of the proposed algorithm

$i \backslash j$	1	2	3	4	5
1	81	10	61	10	10
2	10	71	62	20	21
3	82	72	10	30	32
4	83	73	63	40	43
5	84	74	64	50	44

(a) an example problem

$i \backslash j$	1
1	81
2	10^{Δ}
3	82
4	83
5	84

$\gamma_p=4$
(b) $Z=10$

$i \backslash j$	1	2
1	71	10^{Δ}
2	⓪	71
3	72	72
4	73	73
5	74	74

$\gamma_p=8$
(c) $Z=20$

$i \backslash j$	1	2	3
1	71	⓪	61
2	⓪	61	62
3	72	62	10^{Δ}
4	73	63	63
5	74	64	64

$\gamma_p=12$
(d) $Z=30$

$i \backslash j$	1	2	3	4
1	71	⓪	51	10^{Δ}
2	⓪	61	52	20
3	72	62	⓪	30
4	73	63	53	40
5	74	64	54	50

$\gamma_p=16$
(e) $Z=40$

$i \backslash j$	1	2	3	4	δ_i
~~1~~	~~71~~	~~⓪~~	~~51~~	~~0~~	~~(4)~~ 10
2	⓪	61	52	10^{Δ}	
3	72	62	⓪	20	
4	73	63	53	30	
5	74	64	54	40	
		(1)		(−)	

$\gamma_p=16+8+1=25$
(f) $Z=50$

$i \backslash j$	1	2	3	4	δ_i
~~1~~	~~81~~	~~0~~	~~61~~	~~0~~	~~(4)~~ 20
~~2~~	~~0~~	~~51~~	~~52~~	~~0~~	~~(4)~~ 10
3	72	52	0	10^{Δ}	
4	73	53	53	20	
5	74	54	54	30	
	(2)	(1)		(−)	

$\gamma_p=25+9+2=36$
(g) $Z=60$

$i \backslash j$	1	2	3	4	δ_i
~~1~~	~~81~~	~~⓪~~	~~71~~	~~0~~	~~(4)~~ 30
~~2~~	~~⓪~~	~~51~~	~~62~~	~~0~~	~~(4)~~ 20
~~3~~	~~62~~	~~42~~	~~⓪~~	~~0~~	~~(4)~~ 10
4	63	43	53	10^{Δ}	
5	64	44	54	20	
	(2)	(1)	(3)	(−)	

$\gamma_p=36+8+3=47$
(h) $Z=70$

$i \backslash j$	1	2	3	4	δ_i
1	81	⓪	71	0	(4) 30
2	⓪	51	62	0	(4) 20
3	62	42	⓪	0	(4) 10
4	53	33	43	⓪	(4)
5	54	34	44	10	
	(2)	(1)	(3)	(−)	

$\gamma_p=47$
(i) $Z=70$

$i \backslash j$	1	2	3	4	5	δ_i
1	81	⓪	71	0	10^{Δ}	30
2	⓪	51	62	0	41	20
3	62	42	⓪	0	42	10
4	53	33	43	⓪	43	
5	54	34	44	10	44	

$\gamma_p=47+5+4=56$
(j) $Z=110$

$i \backslash j$	1	2	3	4	5	δ_i
~~1~~	~~81~~	~~⓪~~	~~71~~	~~0~~	~~0~~	~~(5)~~ 31
2	⓪	51	62	0	1^{Δ}	20
3	62	42	⓪	0	2	10
4	53	33	43	⓪	3	
5	54	34	44	10	4	
		(1)			(−)	

$\gamma_p=56+8+1=65$
(k) $Z=111$

$i \backslash j$	1	2	3	4	5	δ_i
~~1~~	~~82~~	~~⓪~~	~~72~~	~~1~~	~~0~~	~~(5)~~ 32
~~2~~	~~⓪~~	~~50~~	~~62~~	~~0~~	~~0~~	~~(5)~~ 21
3	62	41	⓪	0	1^{Δ}	10
4	53	32	43	⓪	2	
5	54	33	44	10	3	
	(2)	(1)			(−)	

$\gamma_p=65+9+2=76$
(l) $Z=112$

$i \backslash j$	1	2	3	4	5	δ_i
~~1~~	~~82~~	~~⓪~~	~~73~~	~~2~~	~~0~~	~~(5)~~ 33
~~2~~	~~⓪~~	~~50~~	~~63~~	~~1~~	~~0~~	~~(5)~~ 22
~~3~~	~~61~~	~~40~~	~~⓪~~	~~0~~	~~0~~	~~(5)~~ 11
4	52	31	43	⓪	1^{Δ}	
5	53	32	44	10	2	
	(2)	(1)	(3)		(−)	

$\gamma_p=76+8+3=87$
(m) $Z=113$

$i \backslash j$	1	2	3	4	5	δ_i
~~1~~	~~82~~	~~⓪~~	~~73~~	~~3~~	~~0~~	~~(5)~~ 34
~~2~~	~~⓪~~	~~50~~	~~63~~	~~2~~	~~0~~	~~(5)~~ 23
~~3~~	~~61~~	~~40~~	~~⓪~~	~~1~~	~~0~~	~~(5)~~ 12
~~4~~	~~51~~	~~30~~	~~42~~	~~⓪~~	~~0~~	~~(5)~~ 1
5	52	31	43	10	1^{Δ}	
	(2)	(1)	(3)	(4)	(−)	

$\gamma_p=87+5+4=96$
(n) $Z=114$

$i \backslash j$	1	2	3	4	5	δ_i
1	82	⓪	73	3	0	(5) 34
2	⓪	50	63	2	0	(5) 23
3	61	40	⓪	1	0	(5) 12
4	51	30	42	⓪	0	(5) 1
5	51	30	42	9	⓪	(5)
	(2)	(1)	(3)	(4)	(−)	

$\gamma_p=96$
(o) $Z=114$

Fig. 7. An example of the worst case;
the proposed algorithm, $m=n=5$, $k_p=3$, $\gamma_p(3)=96$

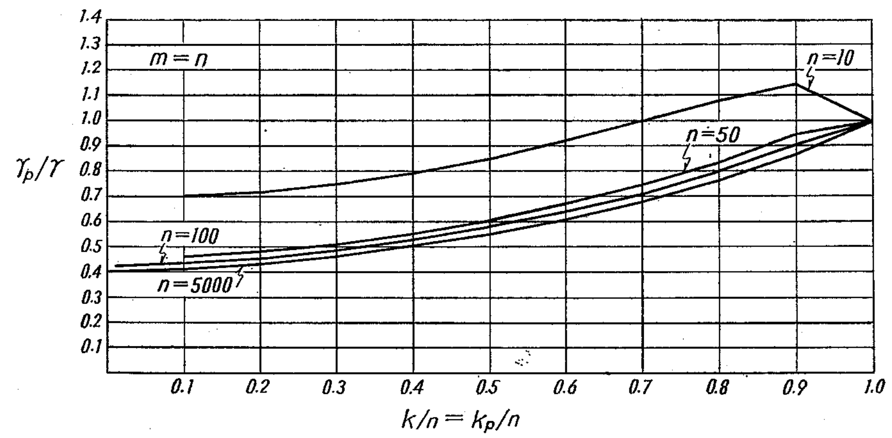


Fig. 8. Comparison of the computational efforts

The total maximum number γ_p needed during solution process of a (m, n) assignment problem is estimated at; (notice that $\delta_i=0$ for $i=1, 2, \cdots, m$ if $k=0$ or 1, and that the additions concerning to δ_i are not needed)

$$\begin{aligned}\gamma_p &= \underset{k=0}{\underset{\uparrow}{(m-1)}}+\underset{k=1}{\underset{\uparrow}{(m-1+2(m-1))}} \\ &+\sum_{k=2}^{n-1}\left\{\frac{m(k^2+3k+4)}{2}-\frac{k(k+1)(2k+1)}{6}-1\right\}=4(m-1) \\ &-\frac{1}{3}\sum_{k=2}^{n-1}k^3+\frac{m-1}{2}\sum_{k=2}^{n-1}k^2+\frac{9m-1}{6}\sum_{k=2}^{n-1}k+\sum_{k=2}^{n-1}2m-(n-2) \\ &=2m(n-1)-(n+1)+\frac{n(n-1)}{12}(2mn-n^2+8m-n) \qquad (20)\end{aligned}$$

Eq. (20) is derived from the assumption that block 6 is repeated by the maximum times for every k $(2\leqq k\leqq n-1)$, and this is the case where $k=1$ in eq. (17). In this case, the coefficients of the 4-th order terms in eq. (17) and (20) are 5/24 and 1/12, respectively, and the computational effort needed by the proposed algorithm is about 40% of that by the conventional one if m and n are large enough.

Assume a case where the first k_p ◎'s are made without passing block 6. In this case, additions relating to δ_i are not needed from $k=0$ to $k=k_p$ because Π_i's are all zero, and the total number $\gamma_p(k_p)$ $(1\leqq k_p\leqq n-1)$ of comparisons is;

$$\begin{aligned}\gamma_p(k_p) &= \sum_{k=0}^{k_p}(m-1)+\sum_{k=k_p+1}^{n-1}(2m-1) \\ &+\sum_{k=k_p}^{n-1}\left\{\frac{m(k^2+3k)}{2}-\frac{k(k+1)(2k+1)}{6}\right\}\end{aligned}$$

$$=m(2n-1)-k_pm-n+\frac{n(n-1)}{12}(2mn-n^2+8m-n)$$
$$-\frac{k_p(k_p-1)}{12}(2k_pm-k_p^2+8m-k_p) \qquad (21)$$

An example of the worst cases where $m=n=5$ and $k_p=3$ is given in Fig. 7.

If $k=k_p=n/2$, $m=n$ in eq. (17) and (21), the coefficients of the 4-th order terms are $47/(24\cdot16)$ and $26/(24\cdot16)$, respectively. This implies $\gamma_p(n/2)=(26/47)\gamma=0.55\gamma$ if $n\to\infty$. Fig. 8 shows γ_p/γ via $k/n=k_p/n$ when $n=10$, 50, 100, 5000 (FACŌM 230-75 Computer in Hokkaido University Computing Center was used).

5. DISCUSSION

Several discussions on the merits of the proposed algorithm are given here.

A little amount of the computational effort is saved by the proposed algorithm if problem sizes are large enough, as it is shown in the previous section. But, if problem sizes are fairly small, more computation than the conventional algorithm is sometimes needed (see for example, Fig. 5 and 7).

If problem sizes are fairly large, the whole cost matrix can not be stored in core memory and communications between auxiliary memory and central processing unit increase remarkably. This causes a remarkable increase of computation time. This increase is surely restrained by the fact that the problem size in making the k-th Ⓠ is only $m\times k$ (which is distinguishably smaller than $m\times n$ during the early stage of the solution process), if the proposed algorithm is used.

It can be said that the proposed algorithm is more effective if problem sizes are larger.

Another merit of the algorithm is that a sequence of problems, $m\times1$, $m\times2$, $\cdots$, $m\times(n-1)$, are also strictly solved. Assume that machines are needed to be assigned to n-jobs but each job has a probability of cancellation. In such a case, arrange the jobs in increasing order of the probabilities, and the proposed algorithm may give flexible assignments which are adaptable to cancelations. On the contrary, if the $n+1$st job is jumped into the schedule, only the $n+1$st is needed for re-scheduling the previous one for n jobs (of course, this may be true even for the conventional algorithm, but it is not so natural expansion as in the proposed one).

Assignment problems often appear as subproblems in more complicated and more difficult-to-solve problems, such as independent assignment problems traveling salesman problems, etc. The effect of the computational saving, though it is not very remarkable, is important in saving the computational effort needed in order to solve such problems.

6. CONCLUSION

A new solution method for assignment problems is proposed, and the computational efficiency is theoretically estimated. The algorithm is based on the conventional Hungarian method and is very effective for large problems.

If an assignment which has no cycle is obtained in each stage $k(1 \leqq k \leqq n-1)$, then the resultant solution is a tour. This suggests the applicability of the proposed algorithm to the traveling salesman problems, because solving smaller problems is always easier than doing the larger problems.

7. REFERENCES

1) Ford, L. R. Jr. and Fulkerson, D. R. "Flows in Networks", Chap. III Princeton Univ. Press, 1962.
2) Little, J. D. C. et al. "An Algorithm for the Traveling Salesman Problem". Opns. Res., Vol. 11, No. 6 (1963), p. 972–989.
3) Kuhn, H. W. "The Hungarian Method for the Assignment Problem" Naval Research Logistics Quarterly, Vol. 2 (1955), p. 83–97.
4) Kuhn, H. W. "Variants of the Hungarian Method for Assignment Problems", Naval Research Logistics Quarterly, Vol. 3 (1956), p. 253–258.