Journal of The Korea Society of Computer and Information
Vol. 21 No. 1, pp. 131-138, January 2016
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2016.21.1.131

An Linear Bottleneck Assignment Problem (LBAP) Algorithm Using the Improving Method of Solution for Linear Minsum Assignment Problem (LSAP)

Sang-Un Lee*

Abstract

In this paper, we propose a simple linear bottleneck assignment problems (LBAP) algorithm to find the optimal solution. Generally, the LBAP has been solved by threshold or augmenting path algorithm. The primary characteristic of proposed algorithm is derived the optimal solution of LBAP from linear sum assignment problem (LSAP). Firstly, we obtains the solution for LSAP from the selected minimum cost of rows and moves the duplicated costs in row to unselected row with minimum increasing cost in direct and indirect paths. Then, we obtain the optimal solution of LBAP according to the maximum cost of LSAP can be move to less cost. For the 29 balanced and 7 unbalanced problem, this algorithm finds optimal solution as simple.

▸ Keywords : Threshold value, Bottleneck, Bottleneck assignment problem, Maxmin, Makespan

• First Author: Sang-Un Lee, Corresponding Author: Sang-Un Lee
*Sang-Un Lee (sulee@gwnu.ac.kr), Dept. of Multimedia Engineering, Gangneung-Wonju National University
• Received: 2015. 08. 31, Revised: 2015. 10. 20, Accepted: 2015. 10. 29.

I. Introduction

다수의 작업들 (Jobs, $J_i, i=1,2,\cdots,m$)과 기계 (Machines, $M_j, j=1,2,\cdots,n$)가 존재하며 각 작업을 특정 기계에서 수행하는 작업시간 (또는 비용, c_{ij})이 다르며, 하나의 작업은 반드시 하나의 기계에서만 작업이 이루어진다. 이 경우, 처리시간 (또는 비용)의 총합이 최소가 되는 최적 해 (optimal solution) $z=\sum_{i=1}^{m}\sum_{j=1}^{n}c_{ij}x_{ij},\ x_{ij}=1$를 찾도록 각 작업을 기계들에 배정하는 문제를 할당 문제 (assignment problem)라 한다[1]-[3]. 이러한 일반적인 할당문제를 선형 합 최소화 할당 문제 (linear sum or minsum assignment problem, LSAP)라 한다[1]-[3]. 반면에, 모든 작업을 동시에 수행할 때, 모든 작업을 가장 빨리 종료시키는 시간을 최소화시키는 해 $z_b=\min\max c_{ij}x_{ij}$를 찾는 문제를 선형 병목 할당문제 (linear bottleneck assignment problem, LBAP)라 한다[4]-[14]. 여기서의 병목은 일반적인 망의 병목현상을 의미하는 것이 아니라 작업을 동시에 수행하였을 때 가장 늦게 끝나는 시간으로 총 작업소요시간 (makespan)을 의미한다. 즉, 이 시간 이전에는 모든 작업을 종료시킬 수 없다.

$m\times n$ 비용 행렬에서 $m=n$인 경우를 균형 할당 문제 (balanced assignment problem, BAP), $m\neq n$인 경우를 불균형 할당 문제 (unbalanced assignment problem, UAP)라 한다[3].

LSAP를 해결하는 방법으로 거의 대부분은 헝가리안 알고리즘 (Hungarian algorithm)[1]-[3]을 적용하고 있으며, 일부는 유전자 알고리즘 (genetic algorithm)[15]을 적용한 경우도 있다. 헝가리안 알고리즘의 수행 복잡도는 $O(n^3)$으로 균형 할당 문제에 대해서는 최적 해 (optimal solution)를 항상 찾을 수 있다고 알려져 있다[1]. LBAP에 대해서는 수행 복잡도가 $O(n^2)$인 한계 알고리즘 (threshold algorithm)과 증대경로 알고리즘 (augmenting path algorithm) 등을 적용하고 있다[4-14].

본 논문은 병목 균형과 병목 불균형 할당 문제인 LBAP의 최적 해를 쉽고 빠르게 찾을 수 있는 알고리즘을 제안한다.

2장에서는 LSAP와 LBAP 개념과 더불어 LBAP 알고리즘을 고찰해 본다. 3장에서는 LSAP로부터 LBAP의 최적 해를 간단히 찾는 LS2LBAP 알고리즘을 제안한다. 4장에서는 다양한 균형과 불균형 할당 문제 사례들에 적용하여 제안된 알고리즘의 적용성을 검증한다.

II. Description and Related Works

LSAP는 식 (1)의 조건을 만족하는 최적 해를, LBAP는 식 (2)를 최소화시키는 최적 해를 찾는다.

$$z_s=\min\sum_{i=1}^{m}\sum_{j=1}^{n}c_{ij}x_{ij} \tag{1}$$

$$\begin{aligned}\text{s.t.}\quad &\sum_{i=1}^{m}x_{ij}=1\ \text{ for } j=1,2,\cdots,n.\\ &\sum_{j=1}^{n}x_{ij}=1\ \text{ for } i=1,2,\cdots,m.\\ &x_{ij}\geq 0,\ \text{for } \forall_{i,j}\end{aligned}$$

$$z_b=\min\max c_{ij}x_{ij} \tag{2}$$

$$\begin{aligned}\text{s.t.}\quad &\sum_{i=1}^{m}x_{ij}=1\ \text{ for } j=1,2,\cdots,n.\\ &\sum_{j=1}^{n}x_{ij}=1\ \text{ for } i=1,2,\cdots,m.\\ &x_{ij}\geq 0,\ \text{for } \forall_{i,j}\end{aligned}$$

LSAP의 최적 해를 찾는 헝가리안 알고리즘은 헝가리 수학자인 Harold Kuhn이 1955년에 제안하고, 1957년에 James Munkres가 보완하여 할당 알고리즘 또는 Kuhn-Munkres 알고리즘이라 부르며, 복잡도는 $O(n^3)$이다.

LBAP의 최적 해를 찾는 대표적인 알고리즘으로는 Gross가 제안한 그림 1의 Gross[11] 알고리즘과 Fulkerson et al.[12]이 제안한 그림 2의 한계 알고리즘이 있다. 이밖에도 증대경로나 벡터 알고리즘 등이 제시되었다. 한계치와 증대경로를 결합시킨 알고리즘에는 Gabow와 Tarjan[13], Pferschy[14]가 있으며, 복잡도는 $O(n^2)$이다.

Step 1. Find the feasible initial solution
$N=\{1,2,\cdots,n\}$, S_n : n's all possible permutation φ set
Start any solution among the $N!$'s feasible solutions.
Step 2. Make a Q matrix
$V=+\infty$
$c'_{ij}=\begin{cases}c_{ij}, & c_{ij}<V\\ M, & c_{ij}\geq V\end{cases}$, M: very large integer
Step 3. Get the solution of Q from transportation algorithm
$V=\max\{c'_{ij}: x_{ij}=1\}$
if $V=M$ then optimal solution=x'_{ij}
else if $V<M$ then $x'_{ij}=x_{ij}$, go to Step 2.

Fig. 1. Gross Algorithm

Step 1. Setting the threshold value c^*_{ij}.
① Ascending sort for c_{ij} or find the c^*_{ij} using binary search
② Find the c^*_{ij} using Hungarian algorithm
$c^* := \max\{\min r_i, \min c_j\}$
$$\overline{c_{ij}} := \begin{cases}1, & \text{if } c_{ij}>c^*_{ij}\\ 0, & \text{if } c_{ij}\leq c^*_{ij}\end{cases} \tag{3}$$
Step 2. Growth the bottleneck augmenting path without "0"

Fig. 2. Threshold Algorithm

표 1은 Kumar[16]에서 인용된 선형 할당 문제이다. 행은 일을, 열은 기계를, 행렬의 값 c_{ij}는 작업에 소요되는 비용 (또는 시간)이다. 4개의 작업을 4개의 기계에 중복되지 않게 할당하여 총 작업비용을 최소화시키는 제약조건을 만족하는 최적 해는 z_s이다. 반면에, 4개 작업을 동시에 4개의 기계에서

수행할 때 가장 빨리 작업을 종료시키도록 배정하는 최적 해는 z_b이다. 표 1에 대해 z_s와 z_b는 그림 3에 제시되어 있다. z_s는 $1+10+5+5$로 작업을 할당하였을 때 총 소요되는 시간은 21시간으로 최소가 된다. z_s로 작업을 동시에 수행한다면 10시간이 지나야 4개 작업을 모두 완료시킬 수 있다. 즉, $T_s = {}_{\max}(1,10,5,5) = 10$이다. 반면에, $z_b = 6+7+4+5 = 22$로 배정하여 동시에 수행하였을 때 병목인 7시간이면 모든 작업이 종료된다. 즉, $T_b = {}_{\max}(6,7,4,5) = 7$이다. 그러나 4개의 기계가 작동된 총 소요 시간은 22시간으로 z_s보다 오래 걸린다. 즉, $z_b \geq z_s, T_b \leq T_s$가 성립한다.

Table 1. A_1 Assignment Problem

c_{ij}		Machines			
		1	2	3	4
Jobs	1	1	4	6	3
	2	8	7	10	9
	3	4	5	11	7
	4	6	7	8	5

$$\begin{bmatrix} 1 & 4 & 6 & 3 \\ 8 & 7 & 10 & 9 \\ 4 & 5 & 11 & 7 \\ 6 & 7 & 8 & 5 \end{bmatrix} \rightarrow \underset{z_s = 1+10+5+5 = 21}{\begin{bmatrix} 1 & 4 & 6 & 3 \\ 8 & 7 & 10 & 9 \\ 4 & 5 & 11 & 7 \\ 6 & 7 & 8 & 5 \end{bmatrix}} \rightarrow \underset{z_b = 6+7+4+5 = 22}{\begin{bmatrix} 1 & 4 & 6 & 3 \\ 8 & 7 & 10 & 9 \\ 4 & 5 & 11 & 7 \\ 6 & 7 & 8 & 5 \end{bmatrix}}$$

Fig. 3. z_s and z_b of A_1 Assignment Problem

III. LSAP-Driven LBAP Algorithm

본 장에서는 헝가리안 알고리즘의 수행 복잡도 $O(n^3)$을 $O(n)$으로 향상시키는 이중경로 경쟁방식으로 LSAP의 해를 구한다. 다음으로, LSAP의 해를 개선하여 LBAP의 해를 구하는 알고리즘을 제안한다. 제안된 알고리즘을 LS2LBAP라 하며, 다음과 같이 수행된다.

$m > n$인 불균형 할당 문제는 편의상 $m < n$으로 행과 열을 교환한다.

Step 1. for $i = 1,2,\cdots,m$ $(m \leq n)$: 수행 복잡도 $O(n)$

${}_{\min}c_{i.(1)}$ 모두 선택

/* $|c_{.j}| \geq 2$이면 $c_{.j}$가 중복되지 않게 조정한다.

end

$|c_{.j}| \geq 2 \in S, |c_{.j}| = 1 \in I, |c_{.j}| = 0 \in T$

$|S| \geq 2$인 모든 열에 대해 S로 취급한다.

Step 2. 행 $i \in S$에 대해 ${}_{\min}c_{i.(2)} \in I$를 선택한다. 이 때 ${}_{\min}c_{i.(2)} \in S$는 고려하지 않으며, ${}_{\min}c_{i.(2)} \in T$이면 간접경로가 없는 것으로 간주. : 수행 복잡도 $O(n)$

Step 3. (1) $S \rightarrow T$의 직접경로에 대한 최소 비용 증가분 ${}_{\min}c_{ST}$를 계산한다.

(2) $S \rightarrow I \rightarrow T$의 간접경로에 대한 최소비용 증가분 ${}_{\min}c_{SIT}$를 계산한다.

(3) ${}_{\min}\{{}_{\min}c_{ST}, {}_{\min}c_{SIT}\}$을 선택하여 이동시킨다. 단, 동일한 비용증가분인 경우 선택된 ${}_{\max}c_{ij}$가 보다 작은 비용으로 이동시킨다.

불균형 할당문제는 $S = \{\phi\}$일 때까지 (1)~(3) 수행.

LSAP의 makespan을 T_s, 작업시간 총합을 z_s로, LBAP에 대해서는 각각 T_b, z_b라 하자.

Step 4. $T_s > T_b$ 개선 : 수행 복잡도 $O(n)$

(1) z_s에서 선택된 비용들 c_{ij}중에서 ${}_{\max}c_{ij} = c_{ij}^*$로 설정.

(2) 행렬에서 $c_{ij} > c_{ij}^*$와 미 선택된 $c_{ij} = c_{ij}^*$ 삭제. 선택된 행렬이 $m > n$인 경우, c_{ij}^*의 행이 $|c_i| \geq 2$이면 c_{ij}^* 열 삭제.

(3) c_{ij}^*가 $|c_i| = 1$또는 $|c_j| = 1$이면 해 개선 불가.

(4) c_{ij}^*가 $|c_i| = 1$또는 $|c_j| = 1$가 아니면

(4-1) 선택된 $c_{ij} < c_{ij}^*$가 $|c_i| = 1$이면 j열 비용 삭제, $|c_j| = 1$이면 i행 비용 삭제, 해당 행과 열을 제외시킴.

(4-2) $|c_i| = 1$또는 $|c_j| = 1$이 없으면 $|c_i| = 1$또는 $|c_j| = 1$가 발생할 때까지 c_{ij}^*부터 비용 내림차순으로 삭제. $|c_i| = 1$또는 $|c_j| = 1$가 동일행이나 열에 1개씩 발생하면 해당 비용으로 확정, go to (3).

단, 이때, $|c_i| = 1$또는 $|c_j| = 1$가 동일행이나 열에 중복 발생하면 최종적으로 삭제된 값을 선택한 후 go to (3).

(5) 남은 행이 $|c_i| = 0$이면 Step 4 종료.

Step 5. $z_s > z_b$ 개선 : 수행 복잡도 $O(n)$

행렬에서 z_s의 $c_{ij} > c_{ij}^*$와 미 선택된 $c_{ij} = c_{ij}^*$의 동일 행은 제외한 나머지 삭제.

z_s의 선택된 c_{ij}들 중에서 ${}_{\max}c_{ij}$를 제외한 c_{ij}들 쌍 $(c_{i_1j_1}, c_{i_2j_2})$에 대해

(1) $m = n$인 균형 할당문제 문제의 경우, $c_{i_2j_1} \leq c_{ij}^*, c_{i_1j_2} \leq c_{ij}^*$에 한해 $c_{i_1j_1} + c_{i_2j_2} > c_{i_2j_1} + c_{i_1j_2}$이면 $c_{i_1j_1} \rightarrow c_{i_2j_1}, c_{i_2j_2} \rightarrow c_{i_1j_2}$로 변경.

(2) $m < n$인 불균형 할당문제 문제의 경우, 미 선택된 열을 포함하여 해당 행의 비용을 감소시킬 수 있는 c_{ij}가 존재하면 해당 c_{ij}로 변경.

A_1 할당문제에 대해 제안된 알고리즘은 그림 4와 같이 수행된다. 첫 번째로, 각 작업 (행) $J_i, i = 1,2,\cdots,m$에서의 최소 비용 ${}_{\min}c_{i.(1)}$을 선택하면 $c_{11}, c_{22}, c_{31}, c_{44}$로 결정된다. 여기서 기

계 (열)에 대해 $|c_{\cdot j}| \geq 2$를 집합 S (start point), $|c_{\cdot j}| = 1$을 집합 I (intermediate point), $|c_{\cdot j}| = 0$을 집합 T (target point)라 하면 $S_j = \{1\}, I_j = \{2,4\}, T_j = \{3\}$이다. 따라서 M_1에 과다하게 할당된 $S_i = \{1,3\}$, $c_{11}, c_{31} \in S$ 중 어느 하나의 작업이 $M_3 \in T$로 이동되어야만 한다. 여기서 이동 방법은 S 집합의 2번째 최소 비용 ${}_{\min}c_{i\cdot(2)}$가 ${}_{\min}c_{i\cdot(2)} \in I$인 경우에 한해 $S \rightarrow I \rightarrow T$로 I를 경유하는 간접경로 증가 비용 c_{SIT}와 $S \rightarrow T$로의 직접경로 증가 비용 c_{ST}중에서 최소 증가비용으로 S에 속한 하나의 작업을 T 기계로 이동시키는 방법을 적용한다. 따라서 LSAP에 대해서는 $(J_3, M_1) = 4 \rightarrow (J_2, M_2) = 5, (J_2, M_2) = 7 \rightarrow (J_2, M_3) = 10$ 으로 이동되어 비용 증가분은 $+4$이다. 결국, $z_s = 1 + 10 + 5 + 5 = 21$, $T_s = {}_{\max}c_{ij} \in z_s = 10$을 얻는다. $c_{ij} \leq T_s$의 값으로 이루어진 행렬에서 $(J_2, M_3) = 10$을 비롯하여, 9, 8, 8을 삭제하면 $(J_1, M_3) = 6, (J_2, M_2) = 7$ 은 행이나 열이 1개만 존재하여 나머지 행이나 열의 비용이 삭제된다. 따라서 3,4행과 1,4열의 2×2 행렬만 남게 되며, 이 때 (4+5=9)<(7+6)=13으로 $(J_3, M_4) = 7, (J_4, M_1) = 6$ 을 삭제하면 LBAP의 $z_b = 6 + 7 + 4 + 5 = 22$, $T_b = 7$이 되어 $z_s < z_b$가 되지만 $T_s > T_b$ 로 해가 개선되었다.

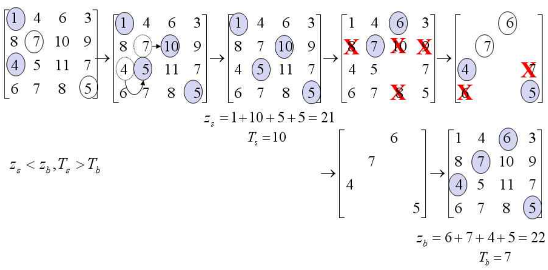
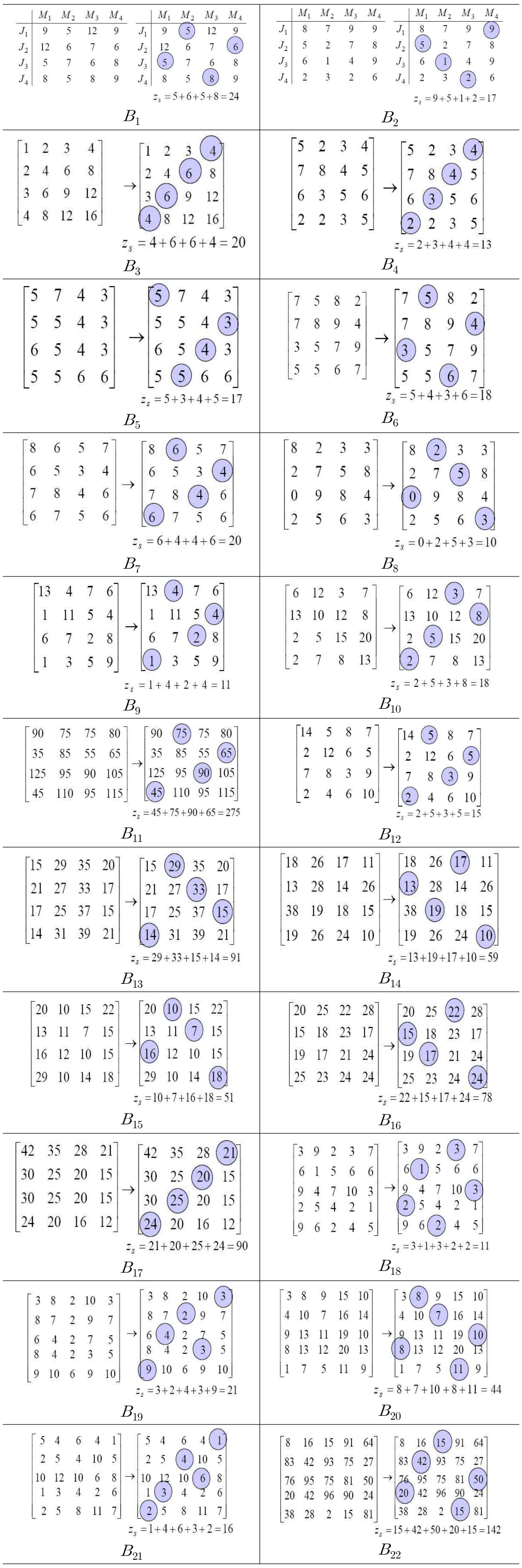


Fig. 4. LS2LBAP Algorithm for A_1 Problem

헝가리안 알고리즘은 행과 열의 최소치 선택, 각 c_{ij}의 잉여량 계산, 선을 그은 다음 $c_{ij} > 0$에 대한 $c_{ij} = c_{ij} - {}_{\min}c_{ij}$와 행과 열의 선이 교차된 c_{ij}에 대한 $c_{ij} = c_{ij} + c_{ij}$를 수행하는 복잡함을 갖고 있다. 반면에, 제안된 LS2LBAP 알고리즘은 단순히 행에 대해서만 최소치를 선택하고, S가 존재시 S에 대한 두 번째 최소치가 I에 속하는 경우에만 선택한다. 다음으로, 단순히 $S \rightarrow T$의 직접경로와 $S \rightarrow I \rightarrow T$의 간접경로 비용 증가분 중에서 최소치에 대해 S의 비용을 T로 이동시켜 LBAP의 해를 결정하였다. 이어서 LBAP의 ${}_{\max}c_{ij} = T_s$부터 내림차순으로 삭제하면서 행과 열에서 1개씩만 선택되도록 하여 LBAP의 z_b와 T_b를 구하였다.

IV. Applications and Evaluation

본 장에서는 Lee[17]에서 인용된 그림 5와 26개 균형 할당과 그림 6의 7개 불균형 할당 문제를 대상으로 LS2LBAP 알고리즘의 적용성을 평가해 본다.



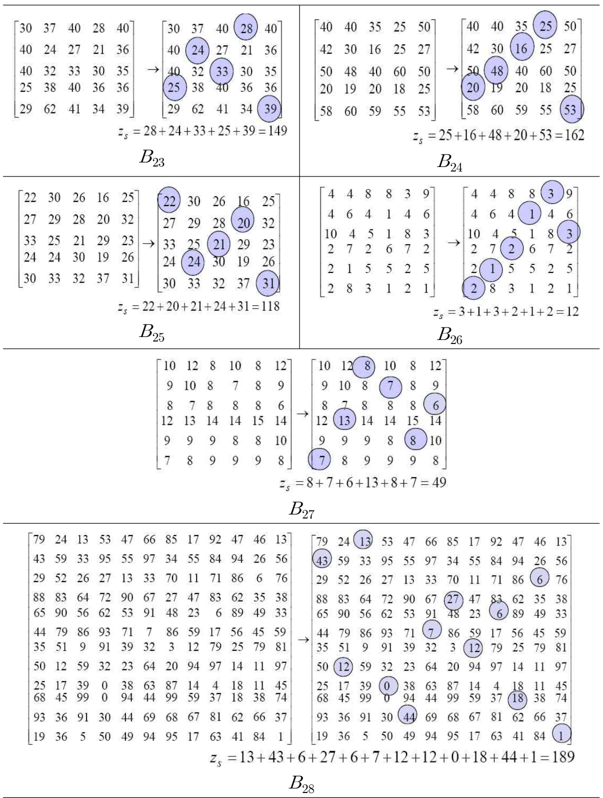


Fig. 5. Experimental Data for Balanced LBAP

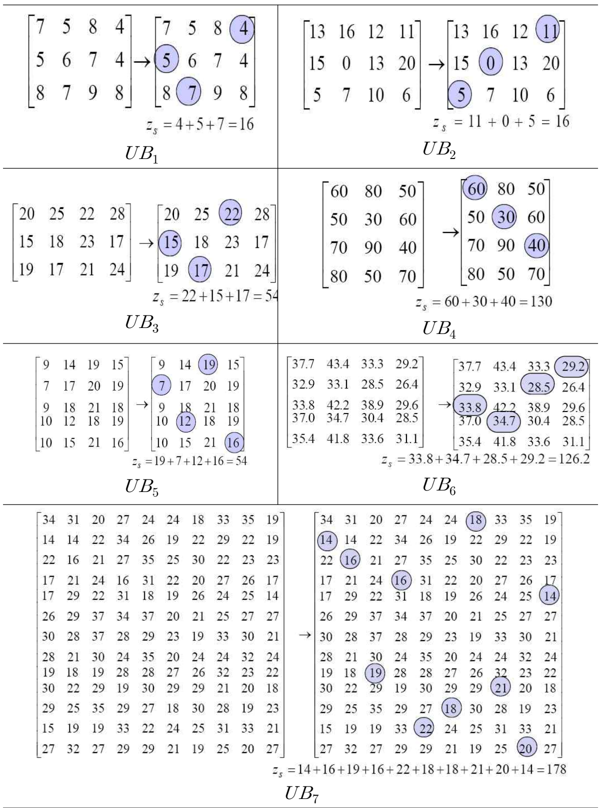


Fig. 6. Experimental Data for Unbalanced LBAP

대표적으로 B_1과 UB_1에 대해 LS2LBAP 알고리즘을 적용한 과정은 그림 7에 제시하였으며, 나머지 데이터들에 대해 LS2LBAP를 적용한 과정은 생략하고, 결과만을 그림 8과 9에 제시하였다.

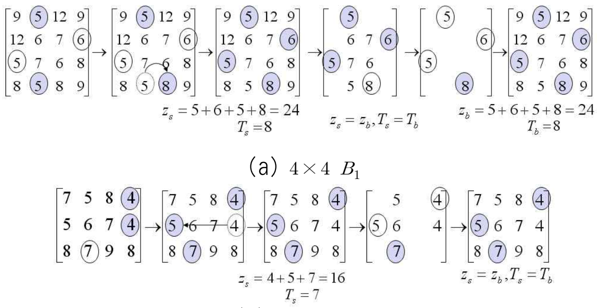
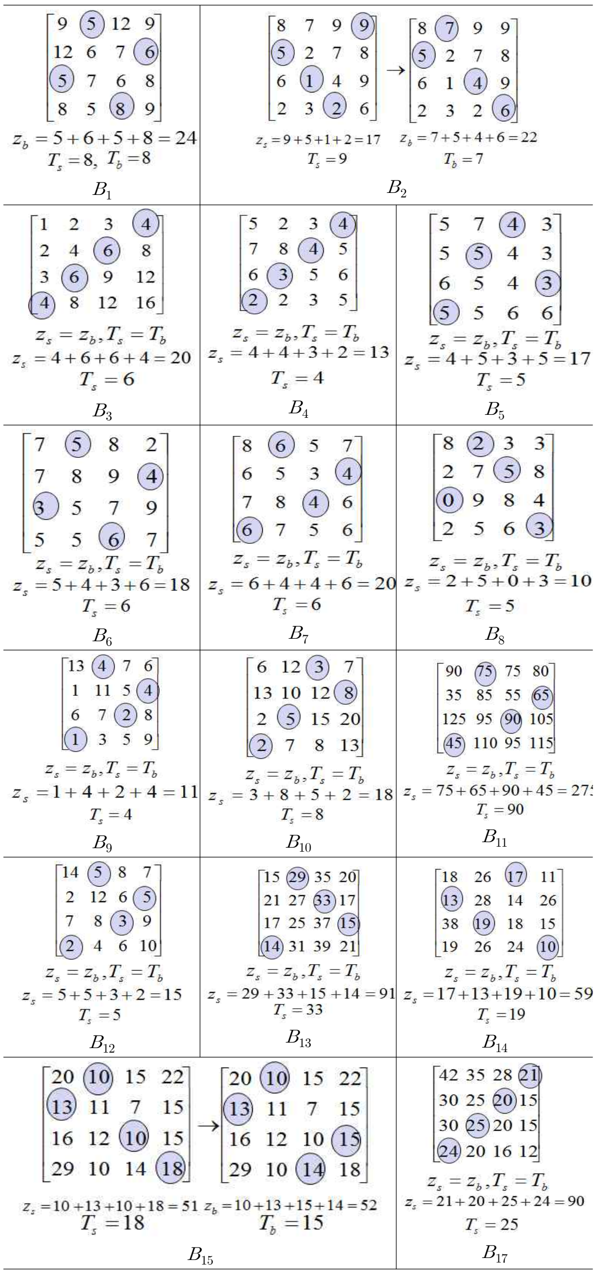


Fig. 7. LS2LBAP Algorithm for Unbalanced Assignment Problem



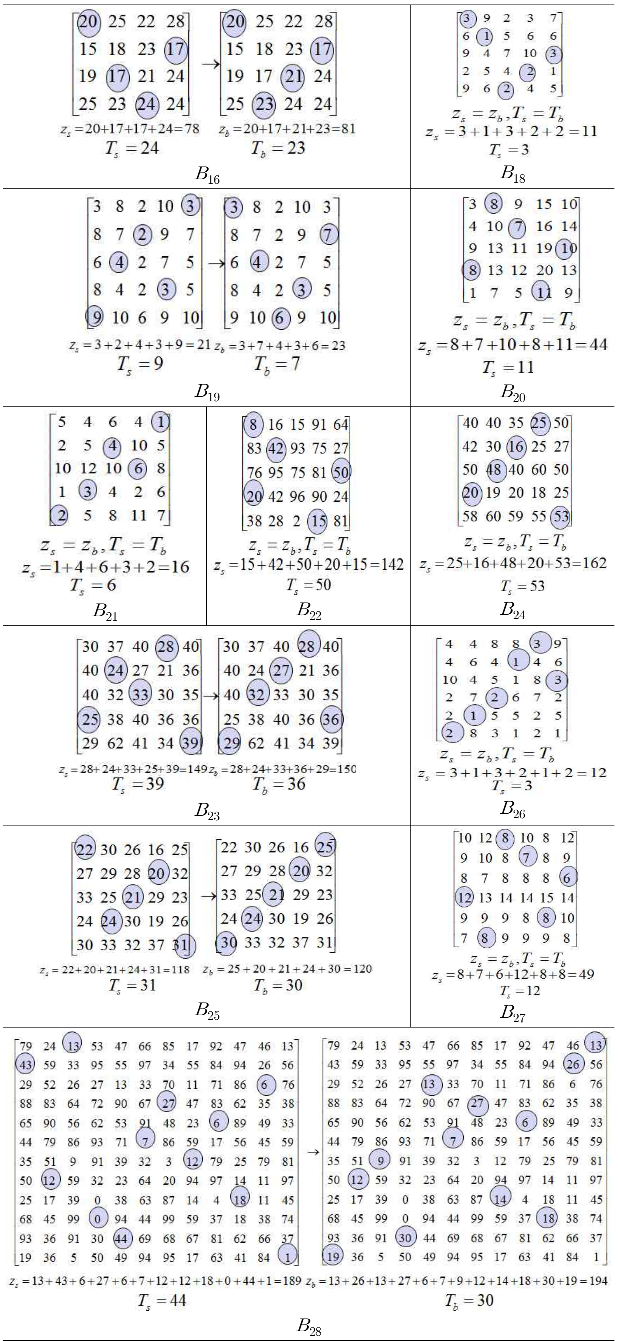


Fig. 8. LS2LBAP Algorithm for Balanced Assignment Problem

B_1의 경우 LSAP의 $T_s = 8$로 결정되었으며, $(J_4, M_3) = 8$의 행에는 $(J_4, M_2) = 5$, 열에는 $(J_3, M_3) = 6$이 존재하지만 $(J_4, M_3) = 8 \rightarrow (J_4, M_2) = 5$로 변경시킬 경우, $(J_1, M_2) = 5$의 다른 열에 값이 존재하지 않아 더 이상의 해 개선이 되지 않음을 알 수 있다. 따라서 $z_s = z_b$, $T_s = T_b$가 된다.

이와 같이 만약, LSAP의 해가 개선되면 $T_b < T_s$, $z_s < z_b$로 됨을 알 수 있다. 만약, LSAP의 해가 개선되지 않으면 $T_b = T_s$, $z_s = z_b$로 LSAP와 LBAP의 해는 동일한 값을 나타냄을 알 수 있다.

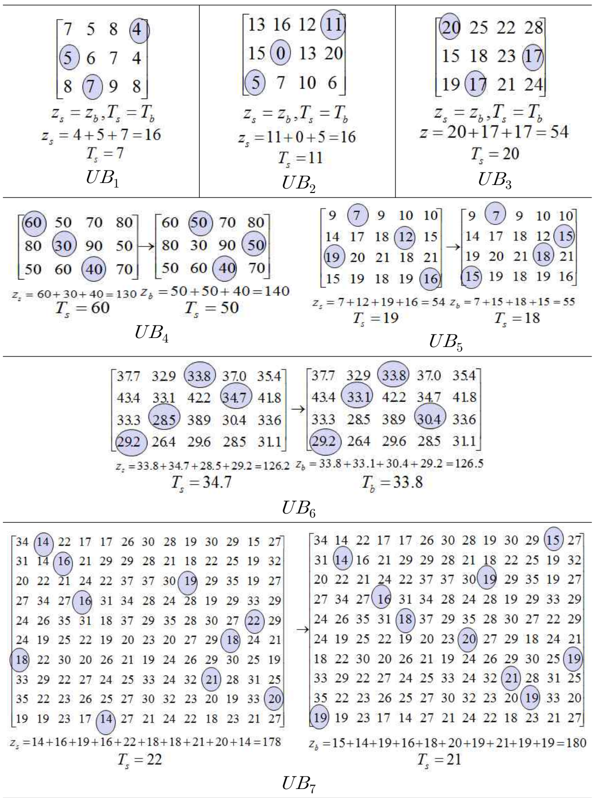


Fig. 9. LS2LBAP Algorithm for Unbalanced Assignment Problem

제안된 LS2LBAP 알고리즘으로 얻은 z_b, T_b를 z_s, T_s와 비교한 결과는 표 2에 제시되어 있다. 표에서 균형 할당문제 28개 중 LSAP의 $T_s > T_b$로 해가 개선된 경우는 8개, 불균형 할당문제에 대해서는 7개중 4개에서 해가 개선되었다.

Table 2. Compare LSAP and LBAP

	Problem	LSAP		LBAP		Solution improvement	
		z_s	T_s	z_b	T_b	$z_s = z_b$ $T_s = T_b$	$z_s < z_b$ $T_s > T_b$
Balanced Assignment	A1	**21**	**10**	**22**	**7**		O
	B1	24	8	24	8	O	
	B2	**17**	**9**	**22**	**7**		O
	B3	20	6	20	6	O	
	B4	13	4	13	4	O	
	B5	17	5	17	5	O	
	B6	18	6	18	6	O	
	B7	20	6	20	6	O	
	B8	10	5	10	5	O	
	B9	11	4	11	4	O	
	B10	18	8	18	8	O	
	B11	275	90	275	90	O	
	B12	15	5	15	5	O	
	B13	91	33	91	33	O	
	B14	59	19	59	19	O	
	B15	**51**	**18**	**52**	**15**		O
	B16	**78**	**24**	**81**	**23**		O
	B17	90	25	90	25	O	
	B18	11	3	11	3	O	
	B19	**21**	**9**	**23**	**7**		O
	B20	44	11	44	11	O	
	B21	16	6	16	6	O	
	B22	142	50	142	50		
	B23	**149**	**39**	**150**	**36**		O
	B24	162	53	162	53	O	
	B25	**118**	**31**	**120**	**30**		O
	B26	12	3	12	3	O	
	B27	49	12	49	12	O	
	B28	**189**	**44**	**194**	**30**		O
Unbalanced Assignment	UB1	16	7	16	7	O	
	UB2	16	11	16	11	O	
	UB3	54	20	54	20	O	
	BU4	**130**	**60**	**140**	**50**		O
	BU5	**54**	**19**	**55**	**18**		O
	BU6	**126.2**	**34.7**	**126.5**	**33.8**		O
	BU7	**178**	**22**	**180**	**21**		O

V. Conclusions

본 논문에서는 LSAP의 해를 $O(n)$의 수행 복잡도로 구하고, T_s를 보다 감소시킬 수 있는 LBAP의 해 T_b를 구하는 알고리즘을 제안하였다.

제안된 알고리즘은 단순히 행에 대해서만 최소치를 선택하고, 중복 선택된 열 S가 존재시 S에 대한 두 번째 최소치가 하나만 선택된 열 I에 속하는 경우에만 선택한다. 다음으로, 단순히 $S \rightarrow T$의 직접경로와 $S \rightarrow I \rightarrow T$의 간접경로 비용 증가분 중에서 최소치에 대해 S의 비용을 T (전혀 선택되지 않은 열)로 이동시켜 LBAP의 해를 결정하였다. 이어서 LBAP의 $_{\max}c_{ij} = T_s$부터 내림차순으로 삭제하면서 행과 열에서 1개씩만 선택되도록 하여 LBAP의 z_b와 T_b를 구하였다.

제안된 LS2LBAP 알고리즘을 병목 균형 할당 28개 문제와 병목 불균형 할당 7개 문제에 적용한 결과 현존하는 수행 복잡도 $O(n^2)$의 LBAP 알고리즘들에 비해 간단하면서도 빠르게 최적 해를 구할 수 있음을 보였다.

REFERENCES

[1] B. Rainer, M. Dell'Amico, and S. Martello, "Assignment Problems," pp. 73-144, SIAM, 2009.

[2] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," Naval Research Logistics Quarterly, Vol. 2, No. 1-2, pp. 83-97, Mar. 1955.

[3] L. Ntaimo, "Introduction to Mathematical Programming: Operations Research: Transportation and Assignment Problems," Vol. 1, 4th ed., Pacific Grove, CA : Thomson, 2005.

[4] F. D. Croce, V. T. Paschos, and A. Tsoukias, "An Improved General Procedure for Lexigraphic Bottleneck Problems," Operations Research Letters, Vol. 24, No. 4, pp. 187-194, May 1999.

[5] H. W. Corley and H. Golnabi, "Technical Note: A Generalized Bottleneck Assignment Problem," Journal of Optimization Theory and Applications, Vol. 36, No. 1, pp. 135-138, Jan. 1982.

[6] R. Burkard, M. Dell'Amico, and S. Martello, "Linear Assignment Problems, Section 6.6 Lexicographic Bottleneck Assignment Problem," pp. 198-202, SIAM, 2008.

[7] E. S. Page, "A Note on Assignment Problems," The Computer Journal, Vol. 6, No. 3, pp. 241-243, 1963.

[8] R. S. Garfinkel, "Technocal Note-An Improved Algorithm for the Bottleneck Assignment Problem," Operations Research, Vol. 19, No. 7, pp. 1747-1751, Nov. 1971.

[9] R. E. Burkard, "Optimierung und Kontrolle: Selected Topics on Assignment Problems," Karl-Franzens-Universität Graz & Technische Universität Graz, 1999.

[10] U. Derigs, "Alternate Strategies for Solving Bottleneck Assignment Problems-Analysis and Computational Results," Computing, Vol. 33, No. 2, pp. 95-106, Jun. 1984.

[11] O. Gross, "The Bottleneck Assignment Problem," Report No. P-1620, The Rand Corporation, Santa Monica, California, 1959.

[12] R. Fulkerson, I. Glickberg, and O. Gross, "A Production Line Assignment Problem," Technical Report RM-1102, The Rand Corporation, Santa Monica, California, 1953.

[13] H. N. Gabow and R. E. Tarjan, "Algorithms for two Bottleneck Optimization Problems," Journal of Algorithms, Vol. 9, No. 3, pp. 411-417, Sep. 1988.

[14] U. Pfershy, "The Random Linear Bottleneck Assignment Problem," Integer Programming and Combinatorial Optimization, Proceedings of 4th International IPCO Conference, Lecture Notes in Computer Science, Vol. 920, pp. 145-156, May 1995.

[15] K. Kinahan and J. Pryor, "Algorithm Animations for Practical Optimization: A Gentle Introduction," http://optlab-server.sce.carleton.ca/POAnimations2007/Default.html, 2007.

[16] D. N. Kumar, "Optimization Methods," http://www.ptel.iitm.ac.in/Courses/Webcourse-contents/IISc-BANG/OPTIMIZATIONMETHODS/pdf/Module_4/M4L3_LN.pdf, IISc, Bangalore, 2008.

[17] S. U. Lee, "Assignment Problem Algorithm Based on the First Selection Method of the Minimum Cost," Journal of IIBC, Vol. 13, No. 5, pp. 163-171, Oct. 2013.

Authors

Sang Un Lee received the B. Sc. degree in avionics from the Korea Aerospace University in 1997. He received the M. Sc. and Ph. D. degrees in Computer Science from Gyeongsang National University, Korea, in 1997 and 2001, respectively.

He is currently Professor with the Department of Multimedia Science, Gangneung-Wonju National University, Korea. He is interested in software quality assurance and reliability modeling, software engineering, software project management, neural networks, and algorithm.