http://dx.doi.org/10.7236/JIIBC.2013.13.6.211
JIIBC 2013-6-27

선형 병목할당 문제의 역-삭제 알고리즘

Linear Bottleneck Assignment Problem Based on Reverse-delete Algorithm

이상운*

Sang-Un, Lee

요 약 본 논문은 선형 병목할당 문제의 최적해를 간단히 찾는 알고리즘을 제안하였다. 일반적으로 병목할당 문제의 최적해는 한계 또는 증대경로 알고리즘으로 구한다. 제안된 알고리즘은 2단계를 수행하는 역-삭제 알고리즘이다. 첫 번째로, 행 또는 열의 개수가 1개가 될 때까지 최대 비용을 삭제하여 초기해를 구한다. 두 번째로 한계치 보다 큰 값이 초기해로 선택되었으면 해를 개선하는 과정을 수행하였다. 제안된 알고리즘을 28개의 병목 균형 할당 문제와 7개의 병목 불균형 할당 문제에 적용한 결과 최적해를 쉽게 찾는데 성공하였다.

Abstract This paper proposes an algorithm that easily finds an optimal solution for linear bottleneck assignment problems. It is either threshold or augmenting path algorithm that is generally used to solve the bottleneck assignment problem. This paper proposes a reverse-delete algorithm that follows 2 steps. Firstly, the algorithm deletes the maximum cost in a given matrix until it renders a single row or column. Next, the algorithm improves any solution that contains a cost exceeding the threshold value c_{ij}^{*}. Upon its application to 28 balanced assignment problems and 7 unbalanced problems, the algorithm is found to be both successful and simple.

Key Words : Threshold value, Bottleneck, Bottleneck assignment problem, Threshold value

Ⅰ. 서 론

선형 할당 문제 (linear assignment problem LAP)는 크게 2가지로 분류된다. 다수의 작업 (job, $J_i, i=1,2,\cdots,m$)과 다수의 기계 (machine, $M_j, j=1,2,\cdots,n$)가 존재하며, 임의의 작업을 임의의 기계에서 수행하는 비용 (또는 시간, c_{ij})은 차이가 있으며, 모든 작업을 모든 기계에 중복없이 하나씩 할당 ($x_{ij}=1$)하는 문제이다. 이 경우, 총 수행시간이 최소가 되는 최적해 $z_s=\min\sum_{i=1}^{m}\sum_{j=1}^{n}c_{ij}x_{ij}$를 찾도록 x_{ij}를 할당하는 문제를 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 시간 (최장 수행시간)을 최소화시키는 해 $z_b=\min\max c_{ij}x_{ij}$를 찾는 문제를 선형 병목 할당문제 (linear bottleneck or minmax assignment problem, LBAP)라 한다[4-14]. 여기서의 병목은 일반적인

*정회원, 강릉원주대학교 과학기술대학 멀티미디어공학과
접수일자 : 2013년 5월 12일, 수정완료 : 2013년 11월 27일
게재확정일자 : 2013년 12월 13일

Received: 12 May, 2013 / Revised: 27 November, 2013 / Accepted: 13 December, 2013
*Corresponding Author: sulee@gwnu.ac.kr
Dept. of Multimedia Eng., Gangneung-Wonju National University, Korea

망의 병목현상을 의미하는 것이 아니라 작업을 동시에 수행하였을 때 가장 늦게 끝나는 시간을 의미한다. 즉, 이 시간 이전에는 모든 작업을 종료시킬 수 없다. 2차원 할당문제 (Quadratic assignment problem, QAP)는 보다 어려워 NP-난제 (NP-hard)로 알려져 있다. 본 논문에서는 LBAP에 한정하여 알고리즘을 제안한다.

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

LSAP를 해결하는 방법으로 거의 대부분은 헝가리안 알고리즘 (Hungarian algorithm)[1-3]을 적용하고 있으며, 일부는 유전자 알고리즘 (genetic algorithm)[4]을 적용한 경우도 있다. 헝가리안 알고리즘의 수행 복잡도는 $O(n^3)$으로 균형 할당 문제에 대해서는 최적해 (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[5-15].

본 논문은 병목 균형과 불균형 할당 문제 뿐 아니라 병목 최단경로의 최적해를 쉽고 빠르게 찾을 수 있는 알고리즘을 제안한다.

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

II. LSAP와 LBAP

LSAP는 식 (1)의 조건을 만족하는 최적해를 찾으며, LBAP는 식 (2)를 최소화시키는 최적해를 찾는다.

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

$$s.t. \quad \sum_{i=1}^{m} x_{ij} = 1 \text{ for } j = 1,2,\cdots,n.$$

$$\sum_{j=1}^{n} x_{ij} = 1 \text{ for } i = 1,2,\cdots,m.$$

$$x_{ij} \geq 0, \text{ for } \forall_{i,j}$$

$$z_b = \min \max c_{ij}x_{ij} \quad (2)$$

$$s.t. \quad \sum_{i=1}^{m} x_{ij} = 1 \text{ for } j = 1,2,\cdots,n.$$

$$\sum_{j=1}^{n} x_{ij} = 1 \text{ for } i = 1,2,\cdots,m.$$

$$x_{ij} \geq 0, \text{ for } \forall_{i,j}$$

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 최적해를 찾는 대표적인 알고리즘으로는 Gross가 제안한 그림 1의 Gross[12] 알고리즘과 Fulkerson, Glicksberg와 Gross가 제안한 그림 2의 한계 알고리즘이 있다. 이밖에도 증대경로 또는 벡터 알고리즘 등이 제시되었다. 한계치와 증대경로를 결합시킨 알고리즘에는 Gabow와 Tarjan[14], Pferschy[15]가 있으며, 복잡도는 $O(n^2)$이다. 이 밖에도 이상운[16,17]은 일반적인 할당 문제에 대한 연구를 수행하였다.

Step 1. 실현 가능한 초기 해
$N = \{1,2,\cdots,n\}$, S_n : n의 모든 순열 (permutation) φ 집합
$N!$의 실현 가능한 해 중에서 임의의 해로 시작.

Step 2. Q 행렬 작성

$$V = +\infty$$

$$c_{ij}^{'} = \begin{cases} c_{ij}, & c_{ij} < V \\ M, & c_{ij} \geq V \end{cases}, \ M\text{: 매우 큰 양의 정수}$$

Step 3. 수송문제 알고리즘으로 Q의 해를 얻음

$$V = \max\left\{c_{ij}^{'} : x_{ij} = 1\right\}$$

if $V = M$ then 최적해=$x_{ij}^{'}$

else if $V < M$ then $x_{ij}^{'} = x_{ij}$, go to Step 2.

그림 1. Gross 알고리즘
Fig. 1. Gross algorithm

Step 1. 한계치 c_{ij}^{*} 설정.

① c_{ij}를 오름차순으로 정렬 또는 이진탐색으로 c_{ij}^{*}를 찾음

② 헝가리안 알고리즘 적용, c_{ij}^{*}를 찾음

$$c^{*} := \max\left\{\min r_i, \min c_j\right\}$$

$$\overline{c_{ij}} := \begin{cases} 1, & \text{if } c_{ij} > c_{ij}^{*} \\ 0, & \text{if } c_{ij} \leq c_{ij}^{*} \end{cases} \quad (3)$$

Step 2. 할당이 "0"을 포함하지 않도록 병목 증대경로 (bottleneck augmenting path)를 성장시킴.

그림 2. 한계 알고리즘
Fig. 2. Threshold algorithm

그림 3은 Kumar[18]에서 인용된 선형 할당 문제이다. 행은 일을, 열은 기계를, 행렬의 값 c_{ij}은 작업에 소요되는 비용 (또는 시간)이다. 4개의 작업을 4개의 기계에 중

복되지 않게 할당하여 총 작업비용을 최소화시키는 제약조건을 만족하는 최적해는 z_s이다. 반면에, 4개 작업을 동시에 4개의 기계에서 수행할 때 가장 빨리 작업을 종료시키도록 배정하는 최적해는 z_b이다. 그림 3에 대해 z_s와 z_b는 그림 4에 제시되어 있다. z_s는 1+10+5+5로 작업을 할당하였을 때 총 소요되는 시간은 21시간으로 최소가 된다. z_s로 작업을 동시에 수행한다면 10시간이 지나야 4개 작업을 모두 완료시킬 수 있다. 반면에, $z_b=6+7+4+5=22$로 배정하여 동시에 수행하였을 때 병목인 7시간이면 모든 작업이 종료된다. 그러나 4개의 기계가 작동된 총 소요시간은 22시간으로 z_s보다 오래 걸린다. 즉, $z_b \geq z_s$이다.

c_{ij}		기계			
		1	2	3	4
작업	1	1	4	6	3
	2	8	7	10	9
	3	4	5	11	7
	4	6	7	8	5

그림 3. A_1 **할당 문제**
Fig. 3. A_1 **Assignment problem**

$$\begin{bmatrix}1&4&6&3\\8&7&10&9\\4&5&11&7\\6&7&8&5\end{bmatrix} \rightarrow \begin{bmatrix}(1)&4&6&3\\8&7&(10)&9\\4&(5)&11&7\\6&7&8&(5)\end{bmatrix} \rightarrow \begin{bmatrix}1&4&(6)&3\\(8)&(7)&10&9\\(4)&5&11&7\\6&7&8&(5)\end{bmatrix}$$

$z_s = 1+10+5+5 = 21$ $z_b = 6+7+4+5 = 22$

그림 4. A_1**의** z_s**와** z_b
Fig. 4. z_s **and** z_b **for** A_1

Ⅲ. 역-삭제 알고리즘

본 장에서는 LBAP에 대해 균형과 불균형 할당 문제와 더불어 s에서 t로의 병목 최단경로에 간단히 적용할 수 있는 알고리즘을 제안한다. 제안된 알고리즘은 다음 정리에 기반하고 있다.

[정리 1] 병목값 b에 대해 어떠한 배정을 하던지 간에 상관없이 항상 $b < c_{ij}^*$를 얻지 못한다. 왜냐하면 c_{ij}^*는 특정 작업의 최소 소요 시간 또는 특정 기계의 최소 소요시간이 되기 때문에 더 이상의 시간을 단축시킬 수 없다.

[정리 2] 병목값 b에 대해 항상 $b \geq c_{ij}^*$이다. 따라서 c_{ij}^*는 b의 하한값 (lower bound value)이다.

제안된 알고리즘은 2단계를 수행한다. Step 1에서는 행 또는 열의 개수가 1개가 될 때까지 행렬의 최대비용 $\max c_{ij}$를 계속 삭제한다. 만약, 행 또는 열에서 1개가 선택되면 해당 값의 나머지 행 또는 열의 값을 삭제한다. 이 과정을 모든 행과 열의 값이 1개가 존재할 때까지 수행한다. 만약, 선택되지 않은 행 또는 열이 존재하면 교차점 값 c_{ij}를 선택한다. 선택된 c_{ij}로 초기해 S를 얻는다. 만약, $\max c_{ij} \in S$에 대해 $\max c_{ij} > c_{ij}^*$이면 Step 2에서 $\max c_{ij}$를 다른 c_{ij}로 이동시켜 해를 개선할 수 있는지 검증한다. 만약, $\max c_{ij} = c_{ij}^*$이면 초기해 S가 z_b호 확정되고 Step 2를 수행하지 않는다.

제안된 알고리즘을 $n \times n$의 병목 균형할당과 $m \times n$ $(m < n)$의 병목 불균형 할당문제에 적용하는 방법은 그림 5에 제시되어 있다. 제안된 알고리즘은 최대값부터 내림차순으로 삭제하는 방법으로 역-삭제 (reverse-delete) 알고리즘이라 칭한다.

Step 1. **최대값** $\max c_{ij}$ **삭제**
/* $\max c_{ij}$: 행렬의 최대값
한계치 (threshold value) $c_{ij}^* = \max\{\min r_i, \min c_j\}$
$i = 1,2,\cdots,n, j = 1,2,\cdots,n$
$s = 0$
if $n \times n$ then /* 균형할당 문제
  while $s = n$
    if $|r_i| = 1$ or $|c_j| = 1$ then
      c_{ij} 선택, r_i, c_j 값 삭제
    else if $\forall (|r_i| > 1 \cap |c_j| > 1)$ then $\max c_{ij}$ 삭제.
    if $|r_i| = 0 \cap |c_j| = 0$ and $s < n$ then
      미 선택된 행과 열 교차점 c_{ij} 선택.
  end
else if $m \times n$ $(m < n)$ then /* 불균형할당 문제
  while $s = m$
    if $|r_i| = 1$ then c_{ij} 선택, r_i, c_j 값 삭제
    else if $\forall |r_i| > 1$ then $\max c_{ij}$ 삭제.
    if $|r_i| = 0$ and $s < n$ then
      미선택된 행과 열 교차점 c_{ij} 선택.
  end
초기해 $z = \sum_{i=1}^{n} c_{ij}$
Step 2. **해 개선**
if $\exists \max(c_{ij}) \in z > c_{ij}^*$ then
  $c_{ij} \leftarrow \max(c_{ij})$ 로 이동, $\hat{z}$ 계산
  /* $c_{ij} < \max(c_{ij})$
  if $\max(c_{ij}) > c_{ij}^*$ and $\max c_{ij}$ 감소 then
    z를 $\hat{z}$로 대체
  else z 확정

그림 5. 병목할당문제의 역-삭제 알고리즘
Fig. 5. Reverse-delete algorithm for bottleneck assignment problem

제안된 알고리즘과 한계 알고리즘과의 차이점은 다음과 같다.

(1) 한계 알고리즘은 초기해를 얻기 위해 순열을 구하고, 식 (3)을 계산한다. 반면에, 역-삭제 알고리즘은 단지 행렬의 최대값을 삭제하는 방법으로 간단히 계산한다.
(2) 한계 알고리즘은 최적해를 얻기 위해 증대경로를 찾는데 반해, 제안된 알고리즘은 $\exists \max(c_{ij}) \in z > c_{ij}^{*}$ 이면 해 개선과정을 수행하며, 그렇지 않으면 초기해를 최적해로 확정한다.

추가적으로, 제안된 알고리즘을 s에서 t로의 병목 최단경로를 구하는 방법은 그림 6에 제시되어 있다. 망의 출발점 s와 도착점 t가 주어진 방향 그래프에서, s에서 t로의 병목 최단경로는 망의 최대 값 $_{\max}c_{ij}$를 삭제하면서 찾는다. 이 때 s는 유입이 없고, 유출만 존재해야 하며, t는 유출이 없고, 유입만 존재해야 한다. 또한, s,t를 제외한 모든 정점 v_i에 대해서는 유입과 유출차수가 1이 되어야 한다. 만약, 유입 또는 유출이 없으면 해당 정점은 삭제된다. 또한, t에서 s로의 역방향 흐름은 성립하지 않으므로 이들 호도 삭제된다.

Network $G(V,E), s, t$
Step 1. **최대값** $_{\max}c_{ij}$ **삭제**
/* $d_G^+(v_i)$: v_i 정점의 유출차수
/* $d_G^-(v_i)$: v_i 정점의 유입차수
while $d_G^+(v_i)=1 \cap d_G^-(v_i)=1$
(단, $d_G^+(s)=1, d_G^-(t)=1$)
$_{\max}c_{ij}$ 삭제.
(t,s)의 역방향 호 삭제.
if $d_G^+(v_i) \geq 1 \cap d_G^-(v_i)=0$ then
v_i 삭제
else if $d_G^+(v_i)=0 \cap d_G^-(v_i) \geq 1$then
v_i 삭제
end
최적해 $z = path(c_{ij})$

그림 6. 병목 최단경로의 역-삭제 알고리즘
Fig. 6. Reverse-delete algorithm for bottleneck shortest path

역-삭제 알고리즘을 A_1 문제에 적용한 결과는 그림 7에 제시되어 있다. Step 1에서 행렬의 최대값 11,10,9,8을 삭제한 결과 2행에서 $(2,2)=7$, 3열에서 $(1,3)=6$만 존재하여 선택된다. 따라서 2열과 1행이 삭제된다. 다음으로 최대값 $(3,4)=7$이 삭제되면 3행에서 $(3,1)=4$와 4열에서 $(4,4)=5$만이 존재하여 4개가 모두 선택되었다. Step 1을 수행한 결과 초기해 $z_b=6+7+4+5=22$를 얻었으며, 이 과정에서 병목값 $b=7$ (2행의 최소값)도 포함되어 있음을 알 수 있다. 따라서 Step 2가 수행되지 않고 초기값을 최적해로 결정하였다. Step 1을 $c_{ij}^{*}=7=\max\{1,7,4,5,1,4,6,3\}$으로 선택하고, $c_{ij} > c_{ij}^{*}$인 값과 c_{ij}^{*}의 행과 열을 모두 삭제하는 방법으로도 얻을 수 있다. 이 경우도 11,10,9,8을 삭제하는 방법과 동일하게 수행된다.

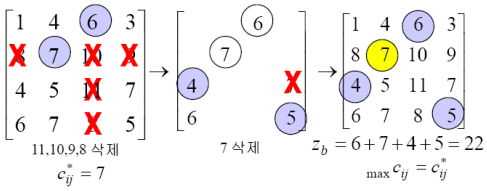


그림 7. A_1 문제의 역-삭제 알고리즘 적용
Fig. 7. Reverse-delete algorithm for A_1 problem

Croce et al.[4]에서 인용된 그림 8의 병목 최단경로를 찾는 문제에 그림 6의 역-삭제 알고리즘을 적용하여 보자. 여기서 $s=A, t=G$이다. 즉, A에서 출발하여 G까지 가는 병목 최단경로를 찾는 문제이다. Croce et al.[4]는 SAP 최단경로 알고리즘인 Dijkstra 알고리즘을 적용해 $z_b=A-B-C-E-F-G$를 구하였다. 일반적으로 최단경로는 $A-B-F-G=2+4+3=9$이다. 제안된 알고리즘을 적용한 결과는 그림 9에 제시되어 있다.

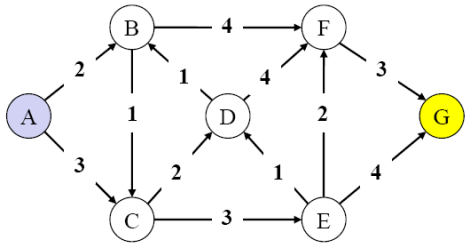
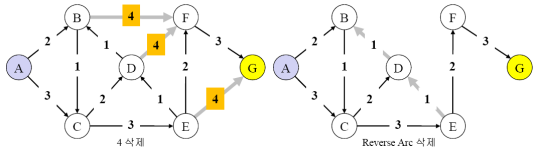


그림 8. 병목 최단경로 문제
Fig. 8. Bottleneck shortest path problem



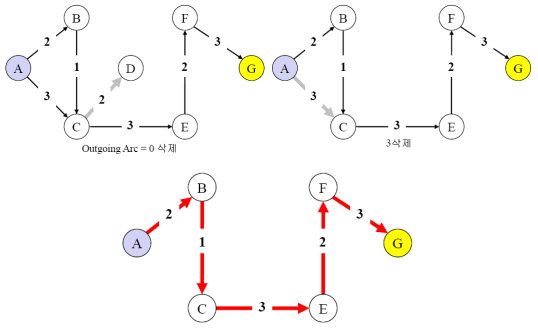


그림 9. 병목 최단경로 문제의 역-삭제 알고리즘
Fig. 9. Reverse-delete algorithm for bottleneck shortest path problem

Ⅳ. 알고리즘 적용성 평가

본 장에서는 그림 10의 27개의 균형 병목 할당 문제와 그림 11의 7개의 불균형 병목 할당 문제를 대상으로 제안된 알고리즘의 적용성을 평가해 본다.

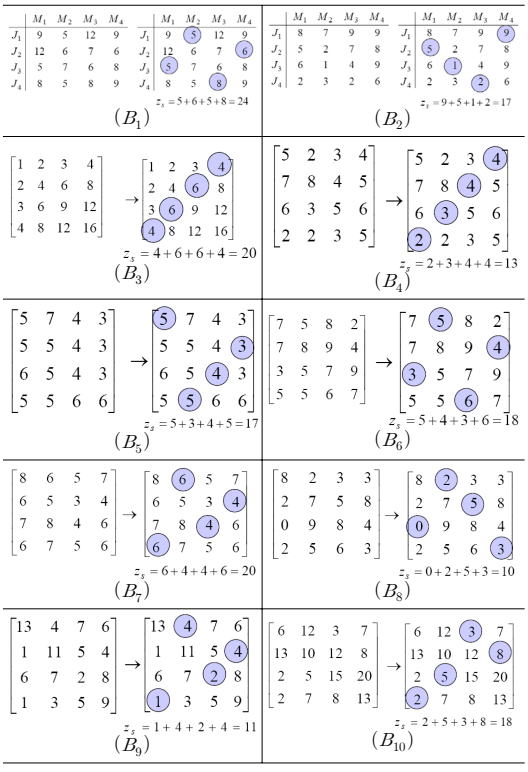


그림 10. 균형 병목 할당 실험 데이터
Fig. 10. Experimental data for balanced bottleneck assignment

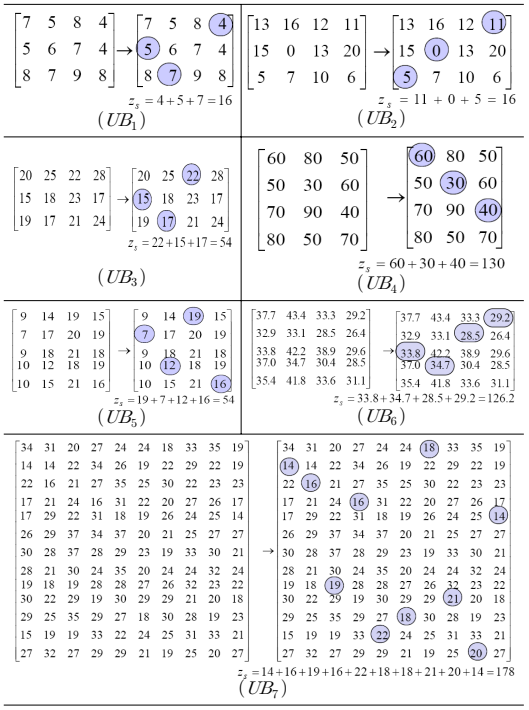
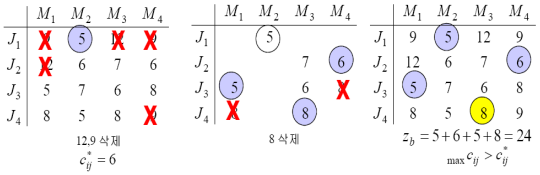


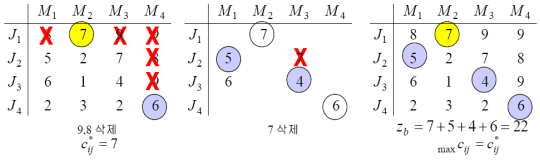
그림 11. 불균형 병목 할당 실험 데이터
Fig. 11. Experimental data for unbalanced bottleneck assignment

4×4비용행렬은 B_1에서 B_{17}까지 17개 데이터이며, 5×5비용행렬은 B_{18}에서 B_{25}까지 8개 데이터, 6×6비용행렬은 B_{26}과 B_{27}의 2개 데이터로 구성되어 있다[3,5,8,19-32].

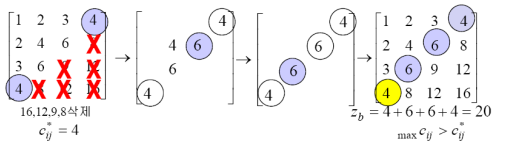
4×4, 5×5, 6×6비용행렬에 대해 역-삭제 알고리즘을 적용한 결과는 각각 그림 12, 그림 13과 그림 14에 제시되어 있다. 27개 균형 할당 문제에 대해 제안된 역-삭제 알고리즘은 모두 최적해를 찾았다.



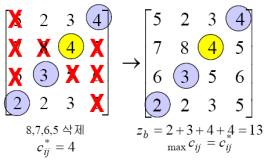
(a) B_1 할당 문제



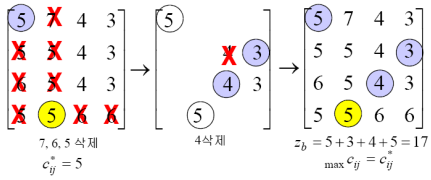
(b) B_2 할당 문제



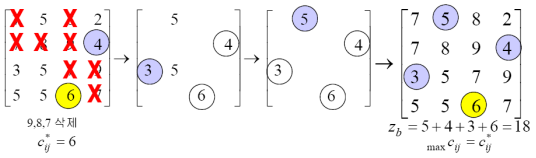
(c) B_3 할당 문제



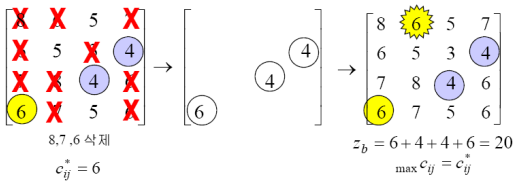
(d) B_4 할당 문제



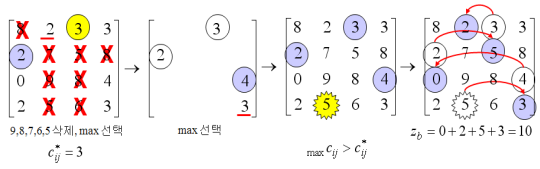
(e) B_5 할당 문제



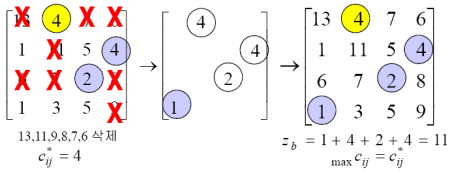
(f) B_6 할당 문제



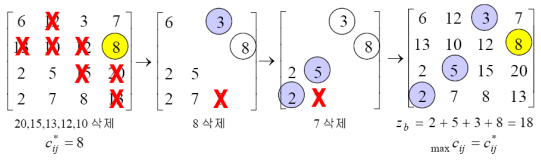
(g) B_7 할당 문제



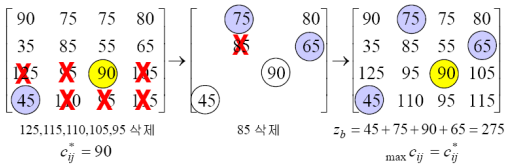
(h) B_8 할당 문제



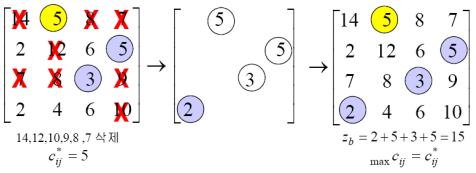
(i) B_9 할당 문제



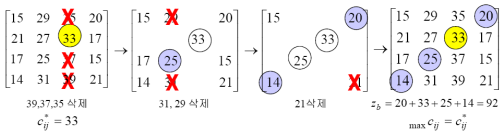
(j) B_{10} 할당 문제



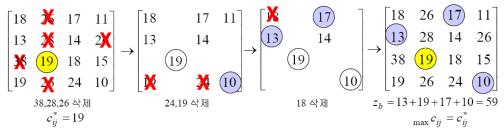
(k) B_{11} 할당 문제



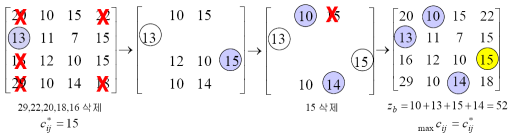
(l) B_{12} 할당 문제



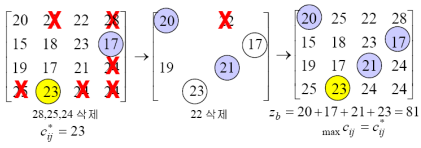
(m) B_{13} 할당 문제



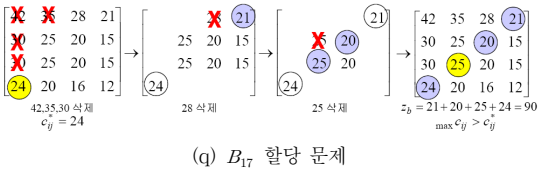
(n) B_{14} 할당 문제



(o) B_{15} 할당 문제

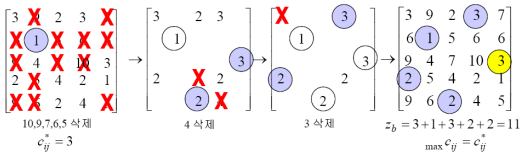


(p) B_{16} 할당 문제

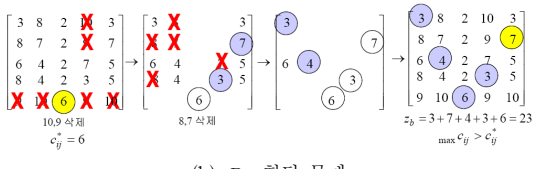


(q) B_{17} 할당 문제

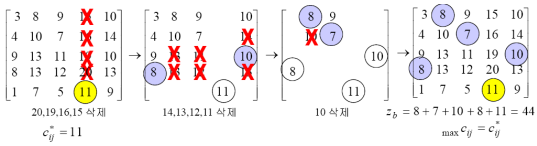
그림 12. 4×4 문제의 역-삭제 알고리즘
Fig. 12. Reverse-delete algorithm for 4×4 problem



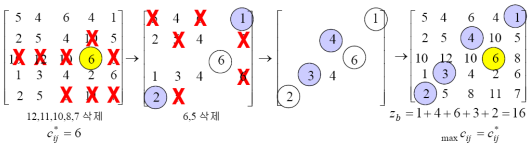
(a) B_{18} 할당 문제



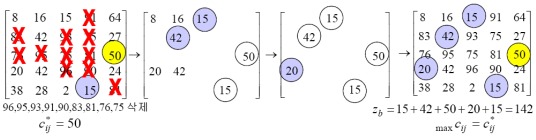
(b) B_{19} 할당 문제



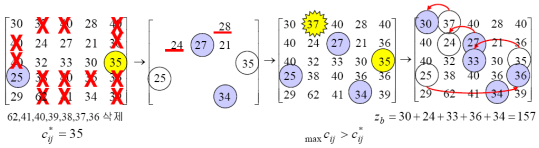
(c) B_{20} 할당 문제



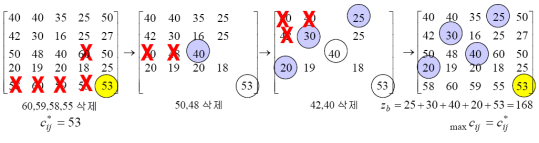
(d) B_{21} 할당 문제



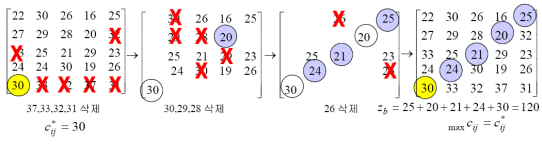
(e) B_{22} 할당 문제



(f) B_{23} 할당 문제

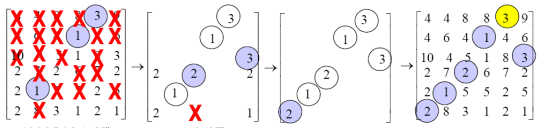


(g) B_{24} 할당 문제

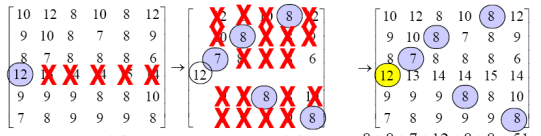


(h) B_{25} 할당 문제

그림 13. 5×5 문제의 역-삭제 알고리즘
Fig. 13. Reverse-delete algorithm for 5×5 problem



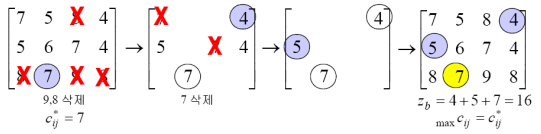
(a) B_{26} 할당 문제



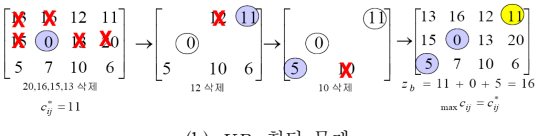
(b) B_{27} 할당 문제

그림 14. 6×6 **문제의 역-삭제 알고리즘**
Fig. 14. Reverse-delete algorithm for 6×6 problem

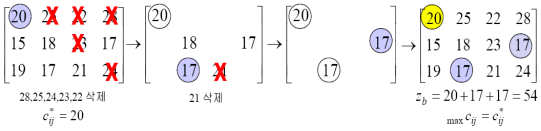
7개 불균형 할당 문제[4,19,22,23,27,33,34]에 대해 최적 할당 알고리즘을 적용한 결과 3×4비용행렬은 그림 15에, 4×3비용행렬은 그림 16에, 5×4비용행렬은 그림 17에, 13×10비용행렬은 그림 18에 제시하였다. 7개의 불균형 할당 문제에 대해 제안된 역-삭제 알고리즘은 모두 최적해를 쉽게 찾는데 성공하였다.



(a) UB_1 할당 문제



(b) UB_2 할당 문제



(c) UB_3 할당 문제

그림 15. 3×4 **문제의 역-삭제 알고리즘**
Fig. 15. Reverse-delete algorithm for 3×4 problem

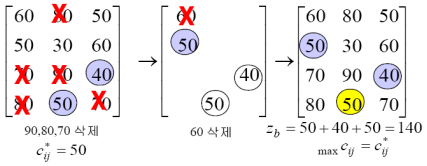
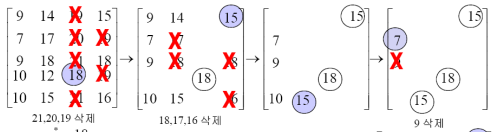
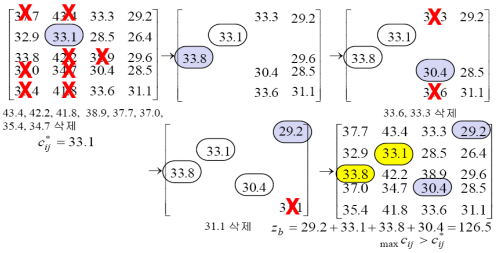


그림 16. 4×3 **문제의 역-삭제 알고리즘**
Fig. 16. Reverse-delete algorithm for 4×3 problem



(a) UB_5 할당 문제



(b) UB_6 할당 문제

그림 17. 5×4 **문제의 역-삭제 알고리즘**
Fig. 17. Reverse-delete algorithm for 5×4 problem

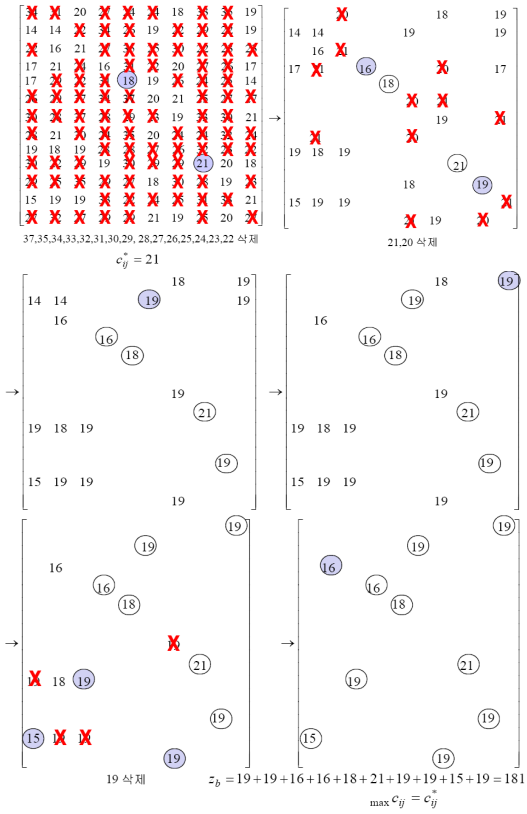


그림 18. 13×10 **문제의 역-삭제 알고리즘**
Fig. 18. Reverse-delete algorithm for 13×10 problem

V. 결 론

본 논문에서는 행렬의 최대비용을 삭제하는 역-삭제 알고리즘을 적용하여 병목 할당문제의 최적해를 구하였다. 제안된 알고리즘은 행 또는 열에 1개의 데이터만 존재할 때까지 행렬의 최대 비용을 선택하는 방법을 적용하여 초기 해를 구하였다. 초기해에 대해 한계값 c_{ij}^{*}를 초과하는 값이 존재하면 할당 값을 재조정하는 2단계를 수행하였다.

제안된 알고리즘을 병목 균형 할당 27 문제와 병목 불균형 할당 7 문제에 적용한 결과 모든 할당 문제에 대해 최적해를 찾는데 성공하였다.

References

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

[2] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," Naval Research Logistics Quarterly, Vol. 2, pp. 83-97, 1955.

[3] L. Ntaimo, "*Introduction to Mathematical Programming: Operations Research: Transportation and Assignment Problems*", Vol. 1, 4th edition, W. L. Winston and M. Venkataramanan, 2005.

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

[5] 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.

[6] 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.

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

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

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

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

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

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

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

[14] 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.

[15] U. Pfershy, "The Random Linear Bottleneck Assignment Problem," RAIRO Operations Research, Vol. 30, No. 2, pp. 127-142, Feb 1996.

[16] Sang-Un Lee, "The Simplified Solution for Assignment Problem," Journal of the Institute of Internet, Broadcasting and Communication, Vol. 12, No. 5, pp. 141-151, 2012.

[17] Sang-Un Lee, "The Assignment Problem Algorithm Based on the First Selection Method of the Minimum Cost," Journal of the Institute of Internet, Broadcasting and Communication, Vol. 13, No. 5, pp. 163-171, 2013.

[18] D. N. Kumar, "Optimization Methods," IISc, Bangalore, 2008.

[19] Rai Foundation Colleges, "Information Research," Bachelor of Business Administration, Business

Administration, 2008.
[20] S. Noble, "Lectures 15: The Assignment Problem," Department of Mathematical Sciences, Brunel University, 2000.
[21] A. Dimitrios, P. Konstantinos, S. Nikolaos, and S. Angelo, "*Applications of a New Network-enabled Solver for the Assignment Problem in Computer-aided Education*," Journal of Computer Science, Vol. 1, No. 1, pp. 19-23, Jan 2005.
[22] R. M. Berka, "A Tutorial on Network Optimization," http://home.eunet.cz/berka/o/English/networks/node8.html, 1997.
[23] M. S. Radhakrishnan, "AAOC C222: Optimization," Birla Institute of Technology & Science, 2006.
[24] R. Burkard, M. D. Amico, and S. Martello, "*Assignment Problems*," SIAM Monographs on Discrete Mathematics and Applications, 2006.
[25] S. C. Niu, "Introduction to Operations Research," School of Management, The University of Texas at Dallas, 2004.
[26] W. Snyder, "The Linear Assignment Problem," Department of Electrical and Computer Engineering, North Carolina State University, 2005.
[27] M. E. Salassi, "AGEC 7123: Operations Research Methods in Agricultural Economics: Standard LP Form of the Generalized Assignment Problem," Department of Agricultural Economics and Agribusiness, Louisiana State University, 2005.
[28] K. Wayne, "Algorithm Design," http://www.cs.princeton.edu/~wayne/kleinberg-tardos/07assignment.pdf, 2005.
[29] J. Havlicek, "Introduction to Management Science and Operation Research," http://orms.czu.cz/text/transproblem.html, 2007.
[30] R. Sedgewick and K. Wayne, " Computer Science 226: Data Structures and Algorithms, Princeton University, 2002.
[31] J. E. Beasley, "Operations Research and Management Science: OR-Notes," Department of Mathematical Sciences, Brunel University, West London, 2004.
[32] D. Doty, "Munkres' Assignment Algorithm: Modified for Rectangular Matrices," KCVU, Murray State University, Dept. of Computer Science and Information Systems, 2008.
[33] M. A. Trick, "Network Optimizations for Consultants," http://mat.gsia.cmu.edu/mstc/networks/ networks.html, 1996.
[34] Optimalon Software, "Transportation Problem (Minimal Cost)," http://www.optimalon.com/examples/transport.htm, 2008.

저자 소개

이 상 운(정회원)

- 1987년 : 한국항공대학교 항공전자공학과 (학사)
- 1997년 : 경상대학교 컴퓨터과학과 (석사)
- 2001년 : 경상대학교 컴퓨터과학과 (박사)
- 2003년 : 강원도립대학 컴퓨터응용과 전임강사
- 2004년~2007년 2월 : 국립 원주대학 여성교양과 조교수
- 2007년 3월~현재 : 강릉원주대학교 멀티미디어공학과 부교수

<주관심분야 :소프트웨어 프로젝트 관리, 개발 방법론, 분석과 설계 방법론, 시험 및 품질보증, 소프트웨어 신뢰성, 그래프 알고리즘>

- E-mail : sulee@gwnu.ac.kr