Journal of The Korea Society of Computer and Information
Vol. 17 No. 9, September 2012
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci/2012.17.9.139

할당 문제의 최적 알고리즘

이 상 운*

The Optimal Algorithm for Assignment Problem

Sang-Un, Lee *

요 약

본 논문에서는 할당 문제의 최적해를 간단히 찾을 수 있는 알고리즘을 제안하였다. 일반적으로 할당 문제의 최적해는 Hungarian 알고리즘으로 구한다. 제안된 알고리즘은 Hungarian 알고리즘의 4단계 수행 과정을 1단계로 단축시켰으며, 행과 열의 최소 비용만을 선택하여 비용을 감소시키는 최적화 과정을 거쳐 최적해를 구하였다. 제안된 알고리즘을 27개의 균형 할당 문제와 7개의 불균형 할당 문제에 적용한 결과 유전자 알고리즘으로 찾지 못한 최적해를 찾는데 성공하였다. 제안된 알고리즘은 Hungarian 알고리즘의 수행 복잡도 $O(n^3)$을 $O(n)$으로 향상시켰다. 따라서 제안된 알고리즘은 Hungarian 알고리즘을 대체하여 할당 문제에 일반적으로 적용할 수 있는 알고리즘으로 널리 활용될 수 있을 것이다.

▸ Keywords : 헝가리안 알고리즘, 균형 할당, 불균형 할당, 최소 비용, 최적 해

Abstract

This paper suggests simple search algorithm for optimal solution in assignment problem. Generally, the optimal solution of assignment problem can be obtained by Hungarian algorithm. The proposed algorithm reduces the 4 steps of Hungarian algorithm to 1 step, and only selects the minimum cost of row and column then gets the optimal solution simply. For the 27 balanced and 7 unbalanced assignment problems, this algorithm finds the optimal solution but the genetic algorithm fails to find this values. This algorithm improves the time complexity $O(n^3)$ of Hungarian algorithm to $O(n)$. Therefore, the proposed algorithm can be general algorithm for assignment problem replace Hungarian algorithm.

▸ Keywords : Hungarian Algorithm, Balanced Assignment, Unbalanced Assignment, Minimum Cost, Optimal Solution

• 제1저자 : 이상운 *
• 투고일 : 2012. 05. 21, 심사일 : 2012. 06. 21, 게재확정일 : 2012. 08. 14.
* 강릉원주대학교 멀티미디어공학과 (Dept. of Multimedia Eng., Gangneung-Wonju National University)

I. 서 론

할당문제 (assignment problem)는 수송문제 (transportation problem)의 특별한 경우로, 다수의 공급처 (source, $S_i, i=1,2,\cdots,m$)와 수요처 (demand, $D_j, j=1,2,\cdots,n$)가 존재하며 모든 공급량과 요구량이 항상 1인 경우이다. 또한, 수송비용 (c_{ij})이 모두 다르며, 한 공급처에서 반드시 한 수요처로만 수송이 이루어져야만 한다. 이 경우 총 수송비용의 합이 최소가 되는 최적해 $z=\sum_{i=1}^{m}\sum_{j=1}^{n}c_{ij}x_{ij}$를 찾도록 $x_{ij}=1$을 할당하는 문제이다.[1-3]

$m\times n$ 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]

할당 문제를 해결하는 방법으로 거의 대부분은 Hungarian 알고리즘[1-3]을 적용하고 있으며, 일부는 유전자 알고리즘[4]을 시도하는 경우도 있다. Hungarian 알고리즘의 수행 복잡도는 $O(n^3)$로 균형 할당 문제에 대해서는 최적해 (optimal solution)를 항상 찾을 수 있다고 알려져 있다.[1] 그러나 균형 할당문제에 대해서도 항상 최적해를 찾지 못하는 경우도 발생하여 최적해를 얻었는지 검증하는 과정이 필요하다. 또한, 불균형 할당 문제에 대해서는 최적의 해법을 찾지 못할 수 있기 때문에 알고리즘을 적용하기 전에 비용이 모두 0인 가상 (dummy)의 행이나 열을 추가하여 균형 할당을 만들어야만 한다.[3] 또한, Hungarian 알고리즘은 마지막으로 얻은 0의 비용들 중에서 각 열에서 중복되지 않게 1개씩만 선택하여야 하는 불편함이 따르며, 0을 포함하는 최소한의 선을 m개를 긋는 과정이 반복적으로 수행되어야만 한다.

본 논문은 비용 행렬의 크기와 무관하며, 균형과 불균형 할당 문제와 무관한 모든 경우에 대해 항상 최적해를 찾는 단순하면서도 일반적으로 적용할 수 있는 알고리즘을 제안한다. 제안된 알고리즘은 Hungarian 알고리즘에 비해 보다 단순히 적용하면서도 최적해를 빠르게 찾을 수 있어 최적 할당 알고리즘 (optimal assignment algorithm)이라 부른다. 제안되는 알고리즘의 수행 복잡도는 Hungarian 알고리즘의 수행 복잡도 $O(n^3)$을 $O(n)$으로 현저히 감소시키는 효과를 얻었다.

2장에서는 할당 문제의 최적해를 찾는 대표적인 Hungarian 알고리즘을 살펴보고 문제점을 고찰해 본다. 3장에서는 최적의 해법을 쉽게 찾는 최적 할당 알고리즘을 제안한다. 4장에서는 최적 할당 알고리즘을 다양한 균형과 불균형 할당 문제 사례들에 적용하여 최적해를 찾는지 평가해 본다.

II. 관련연구와 연구 배경

할당 문제는 수송 문제의 특별한 경우로 공급량과 요구량이 모두 1인 경우이다. 하나의 공급처는 반드시 비용을 최소로 하는 하나의 수요처만을 선택하여야만 한다. 또한, 하나의 수요처는 반드시 하나의 공급처만을 선택해야만 한다. 이는 기계에 일을 부여하는 경우, 사람에게 임무를 부여하는 경우 등에 일반적으로 적용하기 때문에 할당 문제로 부르기도 한다. 할당 문제는 식 (1)의 조건을 만족하는 최적해를 찾는다.

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

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

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

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

할당 문제의 최적 해를 찾는 Hungarian 알고리즘은 헝가리 수학자인 Harold Kuhn이 1955년에 제안하고, 1957년에 James Munkres가 보완하였다. 따라서 할당 알고리즘 또는 Kuhn-Munkres 알고리즘이라 부르기도 한다.[2] Hungarian 알고리즘은 그림 1과 같이 수행된다.

```
m×n (m=n) 비용 행렬
Step 1. /* 수행 복잡도 : O(n^2)
        for i=1 to m
            최소 비용 min c_i를 찾음. /* 각 행에서 최소 비용 min c_i을 찾음.
            for j=1 to n
                c_ij = c_ij − min c_i . /* 각 열의 비용에서 최소 비용을 감산함.
            end
        end
        for j=1 to n
            최소 비용 min c_j를 찾음. /* 각 열에서 최소 비용 min c_i을 찾음.
            for i=1 to m
                c_ij = c_ij − min c_j . /* 각 행의 비용에서 최소 비용을 감산함.
            end
        end
Step 2. /* 수행 복잡도 : O(n^2)
        모든 c_ij = 0 를 포함하는 최소한의 선을 그음.
        /* 선의 수 : |M|
        if |M| = m then go to Step 4.
        else if |M| < m then go to Step 3.
Step 3. /* 수행 복잡도 : O(n^3)
        선에 포함되지 않은 c_ij > 0 들인 감소된 비용 행렬 (Reduced Cost
        Matrix)에서 최소 비용 min c_ij 를 찾음.
        c_ij > 0 : c_ij = c_ij − min c_ij .
        행과 열의 선이 교차된 c_ij : c_ij = c_ij + c_ij .
        go to Step 2.
Step 4. 각 열에서 0이 중첩되지 않게 1개씩 선택, 선택된 셀의 비용을 모두
        더하여 최적해 z를 얻음.
```

그림 1. Hungarian 알고리즘
Fig. 1. Hungarian Algorithm

표 1은 Kumar[5]에서 인용된 할당 문제이다. 행은 일을, 열은 기계를, 행렬의 값 c_{ij}은 작업 수행비용이다. 4개의 작업을 4개의 기계에 중복되지 않게 할당하여 총 작업비용을 최소화시키는 제약조건을 만족하는 최적해를 찾아야 한다.

표 1. A_1 할당 문제
Table 1. A_1 Assignment Problem

c_{ij}		Machine				공급량 (s_i)
		1	2	3	4	
Job	1	1	4	6	3	1
	2	8	7	10	9	1
	3	4	5	11	7	1
	4	6	7	8	5	1
요구량 (d_j)		1	1	1	1	

Kumar[5]가 표 1에 대해 Hungarian 알고리즘을 적용하여 최적해를 찾은 결과는 그림 2와 같다. Step 2를 2회, Step 3을 1회 수행하여 0을 모두 포함하는 최소한의 선을 m개 얻었으며, 작업을 $x_{11}=1, x_{23}=1, x_{32}=1, x_{44}=1$로 할당하여 Step 4에서 최적해 $z=\min\sum_{i=1}^{m}\sum_{j=1}^{n}c_{ij}x_{ij}=1+10+5+5=21$ 을 얻었다.

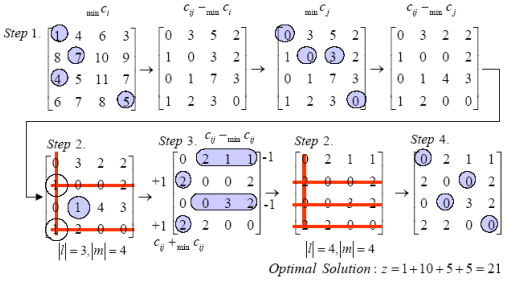


그림 2. A_1의 Hungarian 알고리즘 적용
Fig. 2. Hungarian Algorithm for A_1

Hungarian 알고리즘은 다음 문제점들이 존재한다.

(1) 불균형 할당문제인 경우 부정확한 결과를 얻을 수 있어 가상 행이나 열을 추가하여 $m\times n\,(m=n)$인 균형 할당을 만드는 과정이 필요하다.

(2) 비용 행렬이 크고 0을 다수 포함하는 경우 0을 모두 포함하는 최소한의 라인을 긋는 방법이 어렵다.

(3) 최종적으로 얻은 0 값들에 대해 각 열을 기준으로 중첩되지 않게 1개씩만 선택하는 어려움이 있다.

(4) Hungarian 알고리즘의 수행 복잡도는 $O(n^3)$로 큰 비용행렬인 경우 최적 해법을 도출하기 위해서는 Step 2와 Step 3를 반복 수행하여 최적해를 얻는데 많은 시간이 소요된다.

III. 최적 할당 알고리즘

본 장에서는 Hungarian 알고리즘의 수행 복잡도 $O(n^3)$를 $O(n)$으로 획기적으로 향상시키면서, 균형과 불균형 할당 문제 모두에 즉시 적용할 수 있는 최적 할당 알고리즘을 제안한다.

최적 할당 알고리즘은 Step 1에서 각 행의 최소 비용 $(_{\min}c_i)$과 열의 최소 비용 $(_{\min}c_j)$만을 선택한다. 선택된 $_{\min}c_i$와 $_{\min}c_j$들 중에서 행 또는 열에서 1개만 존재하는 셀 $(s_i=0 \text{ or } s_j=0)$을 결정하고, 행과 열에서 선택된 개수인 s_i와 s_j를 다시 계산한다. 만약, 선택되지 않은 $\{i,j\}$ 셀 $(s_i=0 \text{ or } s_j=0)$의 비용 c_{ij}들이 다수 존재시 이 셀들만을 대상으로 하는 축소된 비용행렬에 대해 반복 수행한다.

Step 2에서는 선택된 $c_{ij}\neq(_{\min}c_i, _{\min}c_j)$를 대체하는 경우와 선택되지 않은 $c_{ij}c_{ij}=(_{\min}c_i, _{\min}c_j)$로 대체 선택시 비용을 감소시킬 수 있으면 초기해를 개선하여 최적해를 얻는다.

제안 알고리즘은 초기 해를 구하는 과정과 최적해를 얻었는지 검증하는 과정이 단순한 장점이 있다. 이 알고리즘을 "최적 할당 알고리즘"이라 하자. 최적 할당 알고리즘의 상세한 내용은 그림 3에 제시되어 있다.

제안된 알고리즘은 초기해 도출 과정에서 행에 대해 최소값을 $O(n)$ 복잡도로 수행하고, 열에 대해 최소값을 $O(n)$ 복잡도로 수행하기 때문에 $O(2n)=O(n)$이다. Step 1의 마지막에 2개 이상 선택된 축소된 행렬을 대상으로 Step 1을 재수행하는 과정에서도 수행 복잡도는 동일하다. 또한, 최적화 과정은 $m\times n$ 행렬의 모든 값들을 대상으로 하지 않고, 단지 출발 지점에서 목적지 까지 최대 n개의 간선들을 대상으로 하기 때문에 이들 간선은 $O(n)$의 복잡도로 찾을 수 있다. 따라서, 제안된 알고리즘은 수행 복잡도가 $O(n)$이다.

표 1의 A_1에 최적 할당 알고리즘을 적용한 과정은 그림 4에 제시되어 있다. Step 1에서 행과 열의 최소 비용을 찾아 $s_i=1$, $s_j=1$인 셀을 선택하였으며, Step 2는 $c_{ij}=(_{\min}c_i, _{\min}c_j)$ 미선택 셀 $(1,1)=1$이 존재하여 $(1,3)=6$이 $(1,1)=1$로 교체되고 이 결과로 $(3,1)=4$가 $(3,2)=5$로, $(2,2)=7$이 $(2,3)=10(2,3)=$으로 이동하여 $-5+1+3=-1$을 감소시키는 결과를 얻어 최적해 $z=21$을 찾았다. 결국, 행과 열의 최소 비용만을 찾고 행과 열의 최소 비용이 선택되었는지 검증하는 과정을 거치면 최적해를 구할 수 있다.

Step 1. 초기 해 도출 /* 동일 비용 다수 존재시 첫 번째 비용 선택
for $i=1$ to n /* 수행 복잡도 : $O(n)$
i 행의 최소 비용 $_{\min}c_i$을 찾음, $s_i = s_i + 1$, $s_j = s_j + 1$.
end
for $j=1$ to n /* 수행 복잡도 : $O(n)$
j열의 최소 비용 $_{\min}c_j$을 찾음, $s_i = s_i + 1$, $s_j = s_j + 1$.
end
$(s_i = 1 \mid s_j = 1)$인 비용 c_{ij} 선택, s_i, s_j 다시 계산.
if $m = n$ 행렬 $(\forall_i, s_i = 1) \cap (\forall_j, s_j = 1)$, $m > n$ 행렬 $(\forall_j, s_j = 1)$, 또는 $m < n$ 행렬 $(\forall_i, s_i = 1)$ then Step 2 수행
else if $(s_i \geq 2 \mid s_j \geq 2)$ then 행 또는 열의 최소 비용 선택, 나머지 삭제.
$s_i = 0$, $s_j = 0$ 확인.
if $(|s_i = 0| = 1) \cap (|s_j = 0| = 1)$ then $\{i, j\}$ 셀 비용 c_{ij} 선택
else if $(|s_i = 0| = 1) \cap (|s_j = 0| \geq 2)$ 또는 $(|s_i = 0| \geq 2) \cap (|s_j = 0| = 1)$ then $\{i, j\}$ 셀의 최소비용 $_{\min}c_{ij}$ 선택
else if $(|s_i = 0| \geq 2) \cap (|s_j = 0| \geq 2)$ then $\{i, j\}$ 셀들의 축소비용 행렬을 대상으로 Step 1 재 수행.

/* 행과 열이 모두 최소 비용인 셀 : $(_{\min}c_i, _{\min}c_j)$
Step 2. 최적 해 검증 /* 수행 복잡도 : $O(n)$
if $m = n \mid m < n$ 비용행렬 then /* 행 비용 이동
if $c_{ij} \neq (_{\min}c_i, _{\min}c_j)$ 선택 then
i 행에서 c_{ij}를 c_{ij} 보다 작은 비용 c_{ik}로 이동시 비용 감소분이 k열에 선택된 비용 c_{lk}가 l행의 다른 비용으로 이동시 비용 증가분보다 크면 $c_{ij} \rightarrow c_{ik}$, c_{lk}를 다른 비용으로 이동.
if $c_{ij} = (_{\min}c_i, _{\min}c_j)$ 미 선택 then
i 행에서 선택된 비용 c_{ij}를 c_{ij}로 이동시 비용 감소분이 j열에서 선택된 c_{lj}을 l행의 다른 비용으로 이동시 비용 증가분보다 크면 $c_{ik} \rightarrow c_{ij}$, c_{lj}를 다른 비용으로 이동.
if $m > n$ 비용행렬 then /* 열 비용 이동
if $c_{ij} \neq (_{\min}c_i, _{\min}c_j)$ 선택 then
j열에서 c_{ij}를 c_{ij} 보다 작은 비용 c_{kj}로 이동시 비용 감소분이 k행 선택된 비용 c_{ik}를 l열의 다른 비용으로 이동시 발생하는 비용 증가분보다 크면 $c_{ij} \rightarrow c_{kj}$, c_{ik}를 다른 비용으로 이동.
if $c_{ij} = (_{\min}c_i, _{\min}c_j)$ 미 선택 then
j열에서 선택된 비용 c_{kj}를 c_{ij}로 이동시 비용 감소분이 i행에서 선택된 c_{il}을 l열의 다른 비용으로 이동시 비용 증가분보다 크면 $c_{kj} \rightarrow c_{ij}$, c_{il}을 다른 비용으로 이동.

그림 3 최적 할당 알고리즘
Fig. 3 Optimal Assignment (OA) Algorithm

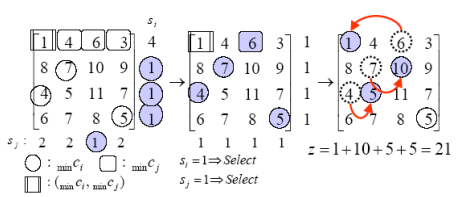


그림 4. A_1 문제의 최적 할당 알고리즘 적용
Fig. 4. OA Algorithm for A_1 Problem

Step 1에서 행과 열의 최소 비용을 찾아 $s_i = 1$, $s_j = 1$인 셀을 선택하였으며, Step 2는 $c_{ij} = (_{\min}c_i, _{\min}c_j)$ 미선택 셀 $(1,1) = 1$이 존재하여 $(1,3) = 6$이 $(1,1) = 1$로 교체되고 이 결과로 $(3,1) = 4$가 $(3,2) = 5$로, $(2,2) = 7$이 $(2,3) = 10$으로 이동하여 $-5 + 1 + 3 = -1$을 감소시키는 결과를 얻어 최적해 $z = 21$을 찾았다. 결국, 행과 열의 최소 비용만을 찾고 행과 열의 최소 비용이 선택되었는지를 검증하는 과정을 거치면 최적해를 구할 수 있다.

결국, 제안된 알고리즘은 Hungarian 알고리즘의 Step 1 ~ Step 4를 Step 1으로 단순화시켰음을 알 수 있다. 또한, Hungarian 알고리즘의 Step 1에서 $c_{ij} = c_{ij} - _{\min} c_i$ 와 $c_{ij} = c_{ij} - _{\min} c_j$를 수행하지 않으며, Step 2의 선을 긋지도 않고, Step 3의 $c_{ij} > 0$에 대한 $c_{ij} = c_{ij} - _{\min} c_{ij}$와 행과 열의 선이 교차된 c_{ij}에 대한 $c_{ij} = c_{ij} + c_{ij}$도 수행하지 않는 단순함이 있다.

IV. 알고리즘 적용성 평가

본 장에서는 26개의 균형 할당과 7개의 불균형 할당 문제를 대상으로 최적 할당 알고리즘의 적용성을 평가해 본다.

1. 균형 할당 문제

표 2의 26개 균형 할당 문제를 대상으로 최적 할당 알고리즘의 적용성을 평가한다.

표 2. 균형 할당 실험 데이터
Table 2. Experimental Data of Balanced Assignment Problems

$$\begin{bmatrix}1&2&3&4\\2&4&6&8\\3&6&9&12\\4&8&12&16\end{bmatrix}\ (B_1) \quad \begin{bmatrix}5&2&3&4\\7&8&4&5\\6&3&5&6\\2&2&3&5\end{bmatrix}\ (B_2) \quad \begin{bmatrix}5&7&4&3\\5&5&4&3\\6&5&4&3\\5&5&6&6\end{bmatrix}\ (B_3) \quad \begin{bmatrix}7&5&8&2\\7&8&9&4\\3&5&7&9\\5&5&6&7\end{bmatrix}\ (B_4) \quad \begin{bmatrix}8&6&5&7\\6&5&3&4\\7&8&4&6\\6&7&5&6\end{bmatrix}\ (B_5)$$

$$\begin{bmatrix}8&2&3&3\\2&7&5&8\\0&9&8&4\\2&5&6&3\end{bmatrix}\ (B_6) \quad \begin{bmatrix}13&4&7&6\\1&11&5&4\\6&7&2&8\\1&3&5&9\end{bmatrix}\ (B_7) \quad \begin{bmatrix}6&12&3&7\\13&10&12&8\\2&5&15&20\\2&7&8&13\end{bmatrix}\ (B_8) \quad \begin{bmatrix}90&75&75&80\\35&85&55&65\\125&95&90&105\\45&110&95&115\end{bmatrix}\ (B_9) \quad \begin{bmatrix}14&5&8&7\\2&12&6&5\\7&8&3&9\\2&4&6&10\end{bmatrix}\ (B_{10})$$

$$\begin{bmatrix}15&29&35&20\\21&27&33&17\\17&25&37&15\\14&31&39&21\end{bmatrix}\ (B_{11}) \quad \begin{bmatrix}18&26&17&11\\13&28&14&26\\38&19&18&15\\19&26&24&10\end{bmatrix}\ (B_{12}) \quad \begin{bmatrix}20&10&15&22\\13&11&7&15\\16&12&10&15\\29&10&14&18\end{bmatrix}\ (B_{13}) \quad \begin{bmatrix}20&25&22&28\\15&18&23&17\\19&17&21&24\\25&23&24&24\end{bmatrix}\ (B_{14}) \quad \begin{bmatrix}42&35&28&21\\30&25&20&15\\30&25&20&15\\24&20&16&12\end{bmatrix}\ (B_{15})$$

$$\begin{bmatrix}3&9&2&3&7\\6&1&5&6&6\\9&4&7&10&3\\2&5&4&2&1\\9&6&2&4&5\end{bmatrix}\ (B_{16}) \quad \begin{bmatrix}3&8&2&10&3\\8&7&2&9&7\\6&4&2&7&5\\8&4&2&3&5\\9&10&6&9&10\end{bmatrix}\ (B_{17}) \quad \begin{bmatrix}3&8&9&15&10\\4&10&7&16&14\\9&13&11&19&10\\8&13&12&20&13\\1&7&5&11&9\end{bmatrix}\ (B_{18}) \quad \begin{bmatrix}5&4&6&4&1\\2&5&4&10&5\\10&12&10&6&8\\1&3&4&2&6\\2&5&8&11&7\end{bmatrix}\ (B_{19})$$

$$\begin{bmatrix}8&16&15&91&64\\83&42&93&75&27\\76&95&75&81&50\\20&42&96&90&24\\38&28&2&15&81\end{bmatrix}\ (B_{20}) \quad \begin{bmatrix}30&37&40&28&40\\40&24&27&21&36\\40&32&33&30&35\\25&38&40&36&36\\29&62&41&34&39\end{bmatrix}\ (B_{21}) \quad \begin{bmatrix}40&40&35&25&50\\42&30&16&25&27\\50&48&40&60&50\\20&19&20&18&25\\58&60&59&55&53\end{bmatrix}\ (B_{22}) \quad \begin{bmatrix}22&30&26&16&25\\27&29&28&20&32\\33&25&21&29&23\\24&24&30&19&26\\30&33&32&37&31\end{bmatrix}\ (B_{23})$$

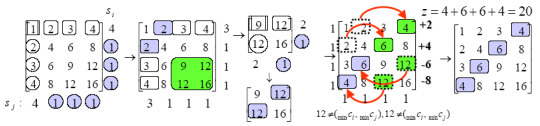
$$\begin{bmatrix}4&4&8&8&3&9\\4&6&4&1&4&6\\10&4&5&1&8&3\\2&7&2&6&7&2\\2&1&5&5&2&5\\2&8&3&1&2&1\end{bmatrix}\ (B_{24}) \quad \begin{bmatrix}10&12&8&10&8&12\\9&10&8&7&8&9\\8&7&8&8&8&6\\12&13&14&14&15&14\\9&9&9&8&8&10\\7&8&9&9&9&8\end{bmatrix}\ (B_{25})$$

79	24	13	53	47	66	85	17	92	47	46	13
43	59	33	95	55	97	34	55	84	94	26	56
29	52	26	27	13	33	70	11	71	86	6	76
88	83	64	72	90	67	27	47	83	62	35	38
65	90	56	62	53	91	48	23	6	89	49	33
44	79	86	93	71	7	86	59	17	56	45	59
35	51	9	91	39	32	3	12	79	25	79	81
50	12	59	32	23	64	20	94	97	14	11	97
25	17	39	0	38	63	87	14	4	18	11	45
68	45	99	0	94	44	99	59	37	18	38	74
93	36	91	30	44	69	68	67	81	62	66	37
19	36	5	50	49	94	95	17	63	41	84	1

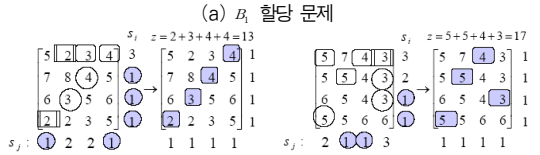
(B_{26})

4×4 비용행렬은 B_1에서 B_{15}까지 15개 데이터이며, 5×5 비용행렬은 B_{16}에서 B_{23}까지 8개 데이터, 6×6 비용행렬은 B_{24}에서 B_{25}까지 2개 데이터, 12×12 비용행렬은 B_{26}의 1개 데이터로 구성되어 있다. B_1은 [6], B_2는 [7], B_3는 [8], B_4는 [9], B_5는 [10], B_6는 [11], B_7은 [12], B_8은 [6], B_9는 [13], B_{10}은 [3,14], $B_{11}\sim B_{15}$는 [6], B_{16}, B_{17}은 [10], B_{18}은 [15], B_{19}는 [16], B_{20}은 [17], B_{21}, B_{22}는 [6], B_{23}은 [18], B_{24}는 [19], B_{25}는 [6], B_{26}은 [20]에서 인용되었다.

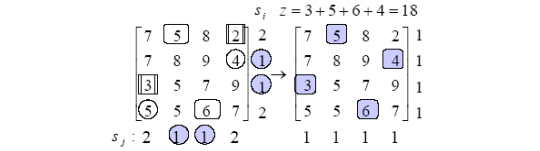
4×4, 5×5, 6×6, 12×12 비용행렬에 대한 최적 할당 알고리즘을 적용한 결과는 각각 그림 5, 그림 6, 그림 7과 그림 8에 제시되어 있다. 26개 균형 할당 문제에 대해 제안된 최적 할당 알고리즘은 모두 최적해를 찾았다. B_{26}은 Dantzig[20] 원문에서 최적의 해를 제시하지 않았다. 반면에 제안된 최적 할당 알고리즘은 최적해 $z=189$을 찾을 수 있었다.



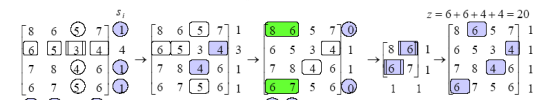
(a) B_1 할당 문제



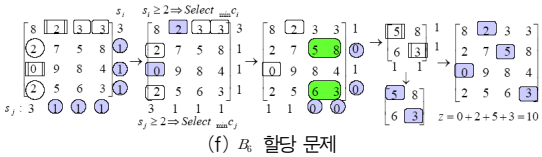
(b) B_2 할당 문제 (c) B_3 할당 문제



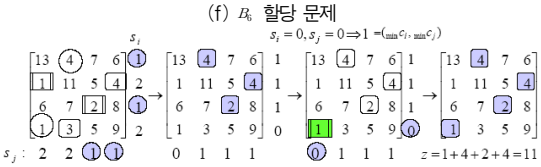
(d) B_4 할당 문제



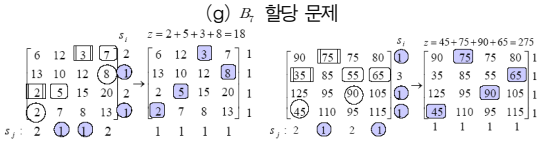
(e) B_5 할당 문제



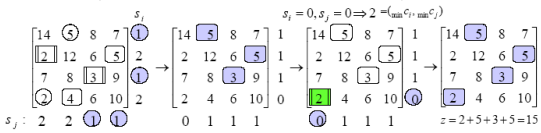
(f) B_6 할당 문제



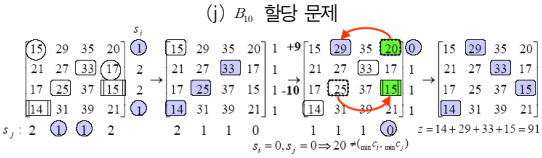
(g) B_7 할당 문제



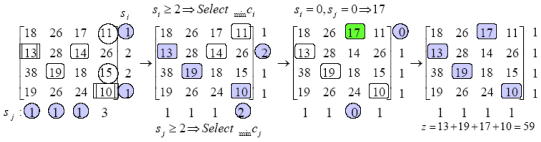
(h) B_8 할당 문제 (i) B_9 할당 문제



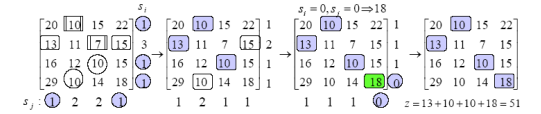
(j) B_{10} 할당 문제



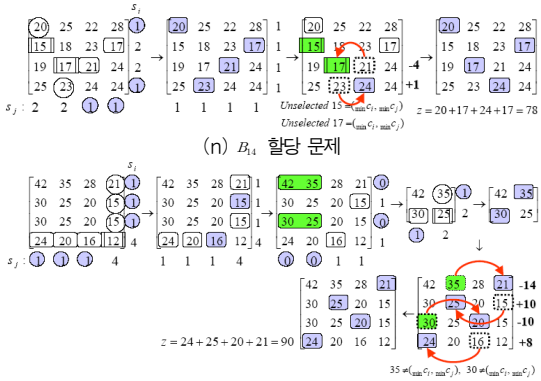
(k) B_{11} 할당 문제



(l) B_{12} 할당 문제



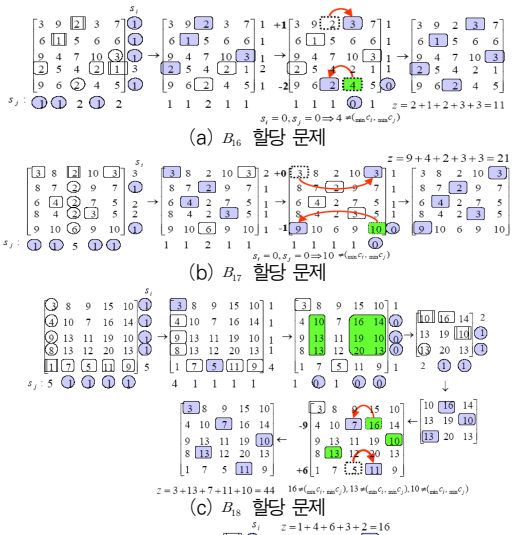
(m) B_{13} 할당 문제



(n) B_{14} 할당 문제

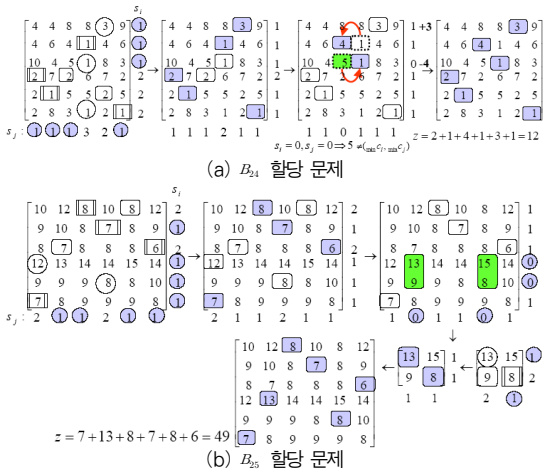
(o) B_{15} 할당 문제

그림 5. 4×4 문제의 최적 할당 알고리즘 적용
Fig. 5. OA Algorithm for 4×4 Matrix Problem



(a) B_{16} 할당 문제 (b) B_{17} 할당 문제 (c) B_{18} 할당 문제 (d) B_{19} 할당 문제 (e) B_{20} 할당 문제 (f) B_{21} 할당 문제 (g) B_{22} 할당 문제 (h) B_{23} 할당 문제

그림 6. 5×5 문제의 최적 할당 알고리즘 적용
Fig. 6. OA Algorithm for 5×5 Matrix Problem



(a) B_{24} 할당 문제 (b) B_{25} 할당 문제

그림 7. 6×6 문제의 최적 할당 알고리즘 적용
Fig. 7. OA Algorithm for 6×6 Matrix Problem

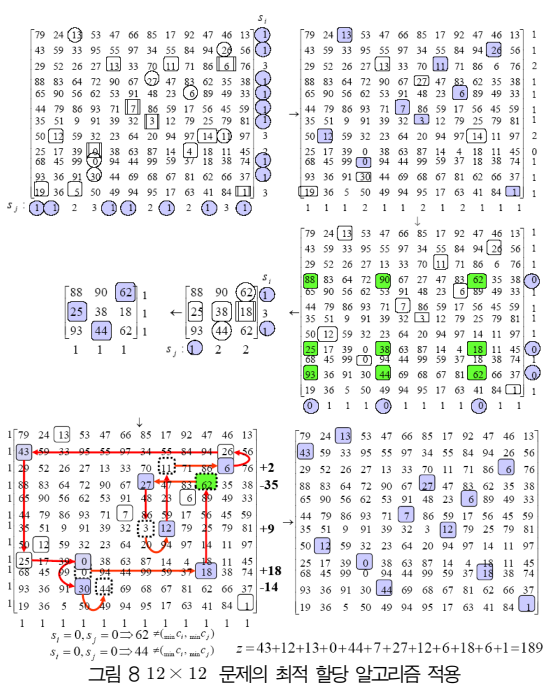


그림 8 12×12 문제의 최적 할당 알고리즘 적용
Fig. 8. OA Algorithm for 12×12 Matrix Problem

2. 불균형 할당 문제

본 절에서는 표 3의 7개 불균형 할당 문제를 대상으로 최적 할당 알고리즘의 적용성을 평가해 본다. UB_1은 [10]에서, UB_2는 [9,14,21]에서, UB_3, UB_4, UB_5는 [6]에서 UB_6는 [22]에서, UB_7는 [4]에서 인용되었다.

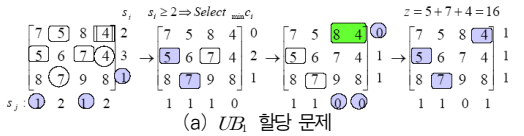
7개 불균형 할당 문제에 대해 최적 할당 알고리즘을 적용

한 결과 3×4 비용행렬은 그림 9에, 4×3 비용행렬은 그림 10에, 5×4 비용행렬은 그림 11에, 13×10 비용행렬은 그림 12에 제시하였다.

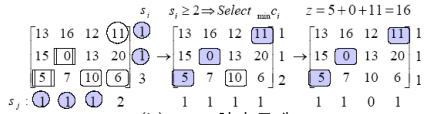
표 3. 불균형 할당 실험 데이터
Table 3. Experimental Data of Un-balanced Assignment Problems

(UB_1)	(UB_2)	(UB_3)	(UB_4)
$\begin{bmatrix}7&5&8&4\\5&6&7&4\\8&7&9&8\end{bmatrix}$	$\begin{bmatrix}13&16&12&11\\15&0&13&20\\5&7&10&6\end{bmatrix}$	$\begin{bmatrix}20&25&22&28\\15&18&23&17\\19&17&21&24\end{bmatrix}$	$\begin{bmatrix}60&80&50\\50&30&60\\70&90&40\\80&50&70\end{bmatrix}$

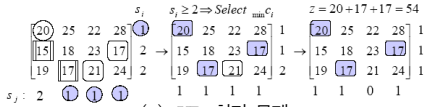
(UB_5)	(UB_7)
$\begin{bmatrix}9&14&19&15\\7&17&20&19\\9&18&21&18\\10&12&18&19\\10&15&21&16\end{bmatrix}$	$\begin{bmatrix}34&31&20&27&24&24&18&33&35&19\\14&14&22&34&26&19&22&29&22&19\\22&16&21&27&35&25&30&22&23&23\\17&21&24&16&31&22&20&27&26&17\\17&29&22&31&18&19&26&24&25&14\\26&29&37&34&37&20&21&25&27&27\\30&28&37&28&29&23&19&33&30&21\\28&21&30&24&35&20&24&24&32&24\\19&18&19&28&28&27&26&32&23&22\\30&22&29&19&30&29&29&21&20&18\\29&25&35&29&27&18&30&28&19&23\\15&19&19&33&22&24&25&31&33&21\\27&32&27&29&29&21&19&25&20&27\end{bmatrix}$
$\begin{bmatrix}37.7&43.4&33.3&29.2\\32.9&33.1&28.5&26.4\\33.8&42.2&38.9&29.6\\37.0&34.7&30.4&28.5\\35.4&41.8&33.6&31.1\end{bmatrix}$	
(UB_6)	



(a) UB_1 할당 문제



(b) UB_2 할당 문제



(c) UB_3 할당 문제

그림 9. 3×4 할당 문제의 최적 할당 알고리즘 적용
Fig. 9. OA Algorithm for 3×4 Matrix Problem

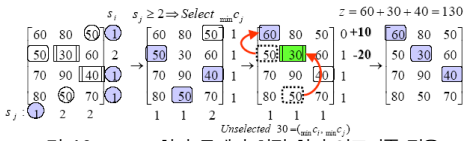
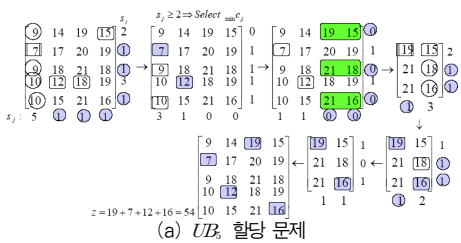
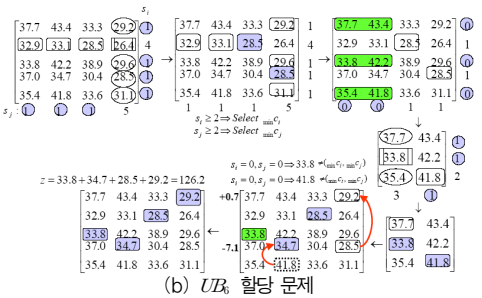


그림 10. 4×3 할당 문제의 최적 할당 알고리즘 적용
Fig. 10. OA Algorithm for 4×3 Matrix Problem



(a) UB_5 할당 문제



(b) UB_6 할당 문제

그림 11. 5×4 할당 문제의 최적 할당 알고리즘 적용
Fig. 11. OA Algorithm for 5×4 Matrix Problem

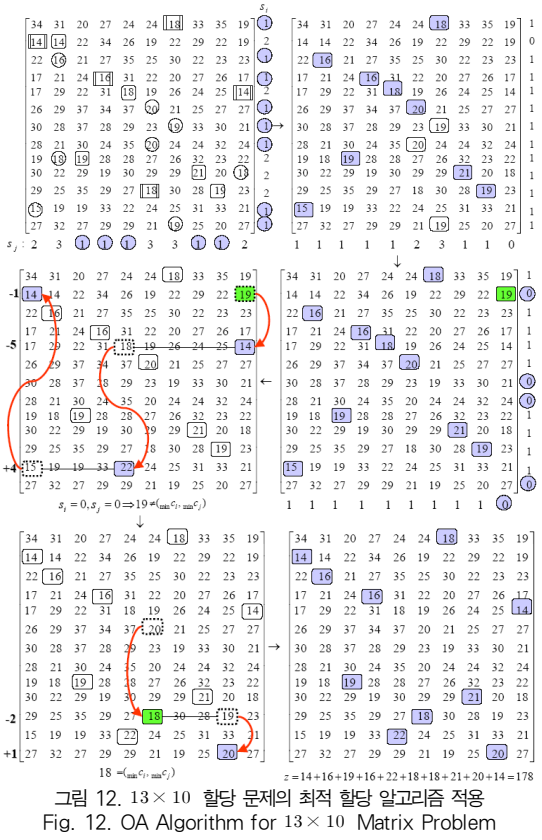


그림 12. 13×10 할당 문제의 최적 할당 알고리즘 적용
Fig. 12. OA Algorithm for 13×10 Matrix Problem

7개 불균형 할당 문제에 대해 제안된 최적 할당 알고리즘은 모두 최적해를 쉽게 찾는데 성공하였다. UB_7은 Kinahan와 Pryor[4] 원문에서 유전자 알고리즘을 적용한 결과 최적의 해법으로 $z=323$이라 제시하고 있다. 반면에 제안된 최적 할당 알고리즘은 최적 해 $z=179$를 찾는데 성공하였다. 결국, 유전자 알고리즘으로 할당문제를 해결하려고 시도하였으나 실패하였음을 알 수 있다.

V. 결론

본 논문에서는 할당 문제의 최적해법을 찾는 최적 할당 알고리즘을 제안하였다. 제안된 알고리즘은 Hungarian 알고리즘의 Step 1 ~ Step 4를 1개의 Step으로 단순화시켰으며, 추가로, 해를 개선하는 Step 2를 수행하였다.

제안된 최적 할당 알고리즘을 균형 할당 26 문제와 불균형 할당 7 문제에 적용한 결과 모든 할당 문제에 대해 최적해를 찾는데 성공하였다.

제안된 알고리즘은 Hungarian 알고리즘의 수행 복잡도 $O(n^3)$ 을 $O(n)$ 으로 획기적으로 개선하였으며, 균형 할당 문제에 대해 최적해를 찾지 못하는 Hungarian 알고리즘의 단점을 보완하였다. 또한, 불균형 할당 문제에 대해 가상 행과 열을 추가하는 과정도 생략되었다.

결국 제안된 최적 할당 알고리즘은 주어진 문제에 대해 단순히 행과 열의 최소비용을 선택하여 최적해를 간단히 찾을 수 있는 알고리즘으로 할당 문제를 해결하는 Hungarian 알고리즘을 대체하여 널리 활용될 수 있을 것이다.

추후, Step 2의 최적해 검증과정을 보다 효율적으로 수행하는 방법을 연구할 계획이다.

참고문헌

〔1〕 Wikipedia, "Assignment Problem," http://en.wikipedia.org/wiki/Assignment_problem, Wikimedia Foundation Inc., 2008.

〔2〕 Wikipedia, "Hungarian Algorithm," http://en.wikipedia.org/wiki/Hungarian_algorithm, Wikimedia Foundation Inc., 2008.

〔3〕 L. Ntaimo, "Introduction to Mathematical Programming: Operations Research: Transportation and Assignment Problems", Vol. 1, 4th edition, by W. L. Winston and M. Venkataramanan, http://ie.tamu.edu/INEN420/INEN420_2005Spring/SLIDES/Chapter 7.pdf, 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〕 D. N. Kumar, "Optimization Methods," http://www.nptel.iitm.ac.in/Courses/Webcourse-contents/IISc-BANG/OPTIMIZATIONMETHODS/pdf/Module_4/M4L3_LN.pdf, IISc, Bangalore, 2008.

〔6〕 Rai Foundation Colleges, "Information Research," Bachelor of Business Administration, Business Administration, http://www.rocw.raifoundation.org/management/bba/OperationResearch/lecture-notes/, 2008.

〔7〕 S. Noble, "Lectures 15: The Assignment Problem," Department of Mathematical Sciences, Brunel University,http://people.brunel.ac.uk/~mastsdn/combopt/handout8.html, 2000.

〔8〕 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, 2005.

〔9〕 R. M. Berka, "A Tutorial on Network Optimization, "http://home.eunet.cz/berka/o/English/networks/node8.html, 1997.

〔10〕 M. S. Radhakrishnan, "AAOC C222: Optimization," Birla Institute of Technology & Science, http://discovery.bits-pilani.ac.in/discipline/math/msr/aaoc222/ppt/assgn1.ppt, 2006.

〔11〕 R. Burkard, M. D. Amico, and S. Martello, "Assignment Problems, http://www.assignmentproblems.com/HA4applet.htm, SIAM Monographs on Discrete Mathematics and Applications, 2006.

〔12〕 S. C. Niu, "Introduction to Operations Research," http://www.utdallas.edu/~scniu/OPRE-6201/documents/TP2-Initialization.pdf, School of Management, The University of Texas at Dallas, 2004.

〔13〕 W. Snyder, "The Linear Assignment Problem," Department of Electrical and Computer Engineering, North Carolina State University, http://www4.ncsu.edu/~wes/Assignment Problem.pdf,2005.

〔14〕 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, http://www.agecon.

lsu.edu/WebClasses/AGEC_7123/2004-Materials/Ovhd-18.pdf, 2005.
〔15〕 K. Wayne, "Algorithm Design," http://www.cs.princeton.edu/~wayne/kleinberg-tardos/07assignment.pdf, 2005.
〔16〕 J. Havlicek, "Introduction to Management Science and Operation Research," http://orms.czu.cz/text/transproblem.html, 2007.
〔17〕 R. Sedgewick and K. Wayne, "Computer Science 226: Data Structures and Algorithms," Princeton University, http://www.cs.princeton.edu/courses/archive/spr02/cs226/assignments/assign.html, 2002.
〔18〕 J. E. Beasley, "Operations Research and Management Science: OR-Notes," Department of Mathematical Sciences, Brunel University, West London, http://people.brunel.ac.uk/~mastjjb/jeb/or/contents.html, 2004.
〔19〕 D. Doty, "Munkres' Assignment Algorithm: Modified for Rectangular Matrices," KCVU, Murray State University, Dept. of Computer Science and Information Systems, http://www.public.iastate.edu/~ddoty/Hungarian Algorithm.html, 2008.
〔20〕 G. B. Dantzig, "Linear Programming and Extensions," USAF Project RAND, R-366-PR, The RAND Corporation, Santa Monica, California, U.S., https://www.rand.org/pubs/reports/2007/R366part2.pdf, 1963.
〔21〕 M. A. Trick, "Network Optimizations for Consultants," http://mat.gsia.cmu.edu/mstc/networks/networks.html, 1996.
〔22〕 Optimalon Software, "Transportation Problem (Minimal Cost)," http://www.optimalon.com/examples/transport.htm,2008.

저 자 소 개

이 상 운 (Sang-Un, Lee)
1983년~1987년 : 한국항공대학교 항공전자공학과(학사)
1995년~1997년 : 경상대학교 컴퓨터과학과(석사)
1998년~2001년 : 경상대학교 컴퓨터과학과(박사)
2003.3~현재 : 강릉원주대학교 멀티미디어공학과 부교수
관심분야 : 소프트웨어 프로젝트 관리, 소프트웨어 개발 방법론, 소프트웨어 신뢰성, 신경망, 뉴로-퍼지, 그래프 알고리즘
e-mail : sulee@gwnu.ac.kr