http://dx.doi.org/10.7236/JIWIT.2012.12.5.141
JIWIT 2012-5-18

할당 문제의 단순한 해법

AThe Simplified Solution for Assignment Problem

이상운*

Sang-Un, Lee

요 약 본 논문은 할당 문제의 최적해를 찾는 대표적인 Hungarian 보다 간단히 최적해를 구하는 알고리즘을 제안하였다. Hungarian 알고리즘은 행과 열의 최소 비용을 선택하고 각 비용에서 최소 비용을 뺀다. 다음으로 0을 모두 포함하는 최소한의 선이 행의 개수 가 될 때까지 수행한다. 반면에 제안된 알고리즘은 단지 행에 대해 최소 비용을 선택한다. 다음으로 열에 대해 2개 이상 선택된 열을 출발지로, 하나도 선택되지 않은 열을 목적지로하여 출발지의 비용에서 목적지의 최소 비용과의 차이인 기회비용이 가장 큰 비용은 고정시키고 기회비용이 보다 작은 비용들을 다음으로 큰 비용으로 이동시키는 방법을 적용하였다. 제안된 방법을 25개의 균형 할당과 7개의 불균형 할당 문제에 적용하여 Hungarian 알고리즘과 동일한 최적해를 구하는데 성공하였다. 제안된 알고리즘은 Hungarian 알고리즘의 수행 복잡도 $O(n^3)$를 $O(n^2)$로 개선하였으며, 불균형을 균형 할당 문제로 변환시키는 과정도 수행하지 않는 단순한 알고리즘이다. 따라서 할당 문제의 Hungarian 알고리즘을 대체시킬 수 있을 것이다.

Abstract This paper suggests more simple algorithm than Hungarian algorithm for assignment problem. Hungarian algorithm selects minimum cost of row and column, and subtracts minimum cost from each cost. Then, performs until the number of minimum lines with 0 equals the number of rows. But, the proposed algorithm selects the minimum cost for each rows only. From the start point with over 2 to the target point with null selects in column, fixes the maximum opportunity cost that the difference of the cost of starting point and target point, and moves the cost less than opportunity cost th more than previous cost. For the 25 balance and 7 unbalance assignment problems, This algorithm gets the optimal solution same as Hungarian algorithm. This algorithm improves the time complexity $O(n^3)$ of Hungarian algorithm to $O(n^2)$, and do not performs the transformation process from unbalance to balance assignment in Hungarian algorithm. Therefore, this algorithm can be alter Hungarian algorithm in assignment problem.

Key Words : Hungarian Algorithm, Balanced Assignment, Unbalanced Assignment, Transportation Problem, Opportunity Cost

Ⅰ. 서 론

할당 문제 (assignment problem)는 수송 문제 (transportation problem)의 특별한 경우이다. 수송 문제는 다수의 공급처와 수요처가 존재하며, 공급량과 요구량, 수송 단위당 소요 비용이 모두 다른 경우, 공급량과 요구량

*정회원, 강릉원주대학교 과학기술대학 멀티미디어공학과
접수일자 : 2012년 5월 18일, 수정완료 : 2012년 8월 20일
게재확정일자 : 2012년 10월 12일

Received: 18 May 2012 / Revised: 20 August 2012 / Accepted: 12 October 2012
*Corresponding Author: sulee@gwnu.ac.kr
Dept. of Multimedia Eng., Gangneung-Wonju National University, Korea

을 모두 만족시키면서 수송될 최소 비용의 합을 찾는 문제이다. 반면에, 할당 문제는 공급과 요구량이 모두 1인 경우로, 한 공급원에서 하나의 요구 부서로만 수송이 이루어져하며, 수송비용 합을 최소화 하는 최적해를 찾는 문제이다.[1-3]

할당 문제는 $m \times n$ 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 방법으로는 일반적으로 거의 대부분은 Hungarian 알고리즘[1-3]을 적용하고 있으며, 일부는 유전자 알고리즘[4]을 시도하는 경우도 있다.

Hungarian 알고리즘은 $m \times n$ 비용 행렬에 대해 수행 복잡도는 $O(n^3)$로 알려져 있으며, $O(n^4)$로 구현한 사례도 존재한다.[2,5] Hungarian 알고리즘은 행의 각 비용에서 최소 비용과의 차이를 계산하고, 열의 각 비용에서 최소 비용과의 차이를 계산하여 최소 비용 0이 포함된 축소된 비용 행렬 (reduced cost matrix)을 얻는다. 축소된 비용 행렬에서 0을 포함하는 최소한의 선이 m개가 존재하면 알고리즘을 종료한다. 만약 그렇지 않으면 선으로 포함되지 않은 비용들 중 최소 비용을 확인하여 최소비용과의 차이를 계산하여 다시 선을 긋는다.[3] Hungarian 알고리즘을 비용 행렬이 매우 큰 경우 (예로, $m \geq 7$), 최소한의 라인을 긋는 방법과 마지막으로 얻은 축소된 비용 행렬에서 각 열이 중첩되지 않게 0을 1개씩만 선택하는 방법에 어려움이 있다. 따라서 할당 알고리즘의 사례는 대부분 $m \leq 6$인 경우만 제시하고 있다.

Hungarian 알고리즘은 균형 할당 문제에 대해서는 최적의 해를 항상 찾을 수 있다. 그러나 불균형 할당 문제는 최적의 해를 찾지 못할 수 있기 때문에 알고리즘을 적용하기 전에 균형 할당을 만들기 위하여 비용이 모두 0인 행이나 열(dummy point)을 추가하여 균형 할당을 만들어야만 한다.[3]

비록, 비용 행렬이 큰 경우 $m > 5$)라도, 할당 문제에 대한 최적의 해를 쉽게 찾기 위해서는 Hungarian 알고리즘보다 수행 복잡도를 현저히 줄이면서, 적용이 쉬운 알고리즘을 개발할 필요가 있다. 또한, 불균형 할당 문제에 대해서도 균형 할당으로 만들지 않고 직접 최적의 해를 찾는 알고리즘이 절실히 요구된다.

이상운[6,7]은 행과 열의 최소 비용을 선택하는 방법을 적용하는 경우와 행렬의 최대 비용을 역으로 삭제하는 방법에 대해 알고리즘을 제안하였다. 본 논문에서는 이들 연구 결과 보다 간단한 방법을 제안한다.

본 논문은 비용 행렬의 크기와 무관하며, 균형과 불균형 할당 문제와 무관한 모든 경우에 대해 일반적으로 적용할 수 있는 단순한 할당 알고리즘을 제안한다. 제안되는 알고리즘의 수행 복잡도는 Hungarian 알고리즘의 수행 복잡도 $O(n^3)$를 $O(n^2)$으로 감소시키는 효과를 얻었다.

2장에서는 할당 문제의 최적해를 찾는 대표적인 Hungarian 알고리즘을 살펴보고 문제점을 고찰해 본다. 3장에서는 최적의 해를 쉽게 찾는 단순 할당 알고리즘을 제안한다. 4장에서는 제안된 단순 할당 알고리즘을 균형과 불균형 할당 문제에 대한 다양한 사례들에 적용하여 최적의 해를 찾는지 평가해 본다.

II. 관련 연구와 연구 배경

수송 문제는 최소 비용 흐름 문제 (minimum cost flow problem)의 특별한 경우로 다수의 공급처 (공장)와 수요처 (물류창고)가 존재하고, 공급량, 요구량, 수송 단위당 비용이 모두 다른 경우가 적용된다. 즉, m개의 공장에서 서로 다른 공급량 s_i를, n개의 물류창고에서 서로 다른 요구량 d_j를 갖고 있다. 또한, 공급처에서 물류창고로 수송되는 서로 다른 단위당 수송비용 c_{ij}가 존재한다. 이 경우, 주어진 공급량으로부터 요구량을 만족시키도록 최소 비용을 찾는 문제로, 각 공급처로부터 물류창고에 수송될 양을 x_{ij}라 하면, 식 (1)의 최적해를 찾는 문제이다. 여기서 모든 x_{ij}는 정수 값을 가져야만 한다.

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

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

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

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

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 선택

해야만 한다. 이는 기계에 일을 부여하는 경우, 사람에게 임무를 부여하는 경우 등에 일반적으로 적용하기 때문에 할당 문제로 부르기도 한다. 할당 문제는 식 (2)의 최적해를 찾는 문제이다.

$$z = \min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \qquad (2)$$

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

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

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

할당 문제를 선형 할당 문제 (linear assignment problem)라고도 한다. 그림 1은 Synder[8]에서 인용된 할당 문제이다.

	D_1	D_2	D_3	D_4
S_1	90	75	75	80
S_2	35	85	55	65
S_3	125	95	90	105
S_4	45	110	95	115

그림 1. A_1 **할당 문제**
Fig 1. A_1 **Assignment Problem**

행은 공장 (또는 기계, 사람)을, 열은 물류창고 (또는 임무, 일)을, 행렬의 값 c_{ij}은 수송 (또는 수행) 단위 당 소요되는 비용이다. 4개의 공장에서 4개의 물류창고로 수송해야 하는 경우, 각 공장은 하나의 물류창고에만 수송해야 한다. 또한, 각 물류창고는 한 공장에서만 공급 받을 수 있다. 한 공장에서 각 물류창고로 수송하는데 소요되는 비용이 존재한다. 이 경우 총 소요비용을 최소화시키는 제약조건을 만족하는 최적해를 찾는 문제는 선형 방정식을 풀어야 하기 때문에 선형계획법 (linear program)이나 정수계획법 (integer program)이라 한다. 즉, 식 (3)을 만족하는 최적해를 찾아야만 한다.

$$\begin{aligned} \min z = {} & 90x_{11} + 75x_{12} + 75x_{13} + 80x_{14} + \\ & 35x_{21} + 85x_{22} + 55x_{23} + 65x_{24} + \\ & 125x_{31} + 95x_{32} + 90x_{33} + 105x_{34} + \\ & 45x_{41} + 110x_{42} + 95x_{43} + 115x_{44} \end{aligned} \qquad (3)$$

$$s.t. \quad \begin{aligned} & x_{11} + x_{12} + x_{13} + x_{14} = 1 \qquad (\textit{Supply Constraints}) \\ & x_{21} + x_{22} + x_{23} + x_{24} = 1 \\ & x_{31} + x_{32} + x_{33} + x_{34} = 1 \\ & x_{41} + x_{42} + x_{43} + x_{44} = 1 \end{aligned}$$

$$\begin{aligned} & x_{11} + x_{21} + x_{31} + x_{41} = 1 \qquad (\textit{Demand Constraints}) \\ & x_{12} + x_{22} + x_{32} + x_{42} = 1 \\ & x_{13} + x_{23} + x_{33} + x_{43} = 1 \\ & x_{14} + x_{24} + x_{34} + x_{44} = 1 \end{aligned}$$

할당 문제의 최적해를 찾는 대표적인 Hungarian 알고리즘은 1955년에 헝가리 수학자인 Harold Kuhn이 처음 제안하였으며, 1957년에 James Munkres가 보완하였다. 따라서 Hungarian 알고리즘을 Munkres 할당 알고리즘 또는 Kuhn-Munkres 알고리즘이라 부르기도 한다.[2]

할당 문제의 최적해를 찾는 대표적인 Hungarian 알고리즘은 그림 2와 같이 수행된다.

$m \times n$ $(m = n)$ 비용 행렬
Step 1. /* 수행 복잡도 : $O(n^2)$
for $i = 1$ to m
최소 비용 $\min c_i$를 찾음. /* 각 행에서 최소 비용 $\min c_i$을 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 $\min c_j$를 찾음. /* 각 열에서 최소 비용 $\min c_i$을 찾음.
for $i = 1$ to m
$c_{ij} = c_{ij} - \min c_j$. /* 각 행의 비용에서 최소 비용을 감산함.
end
end
/* Reduced Cost Matrix를 얻음.
Step 2. /* 수행 복잡도 : $O(n)$
Reduced Cost Matrix에 대해 모든 $c_{ij} = 0$를 포함하는 최소한의 선을 그림.
/* 선의 수 : $|l|$
if $|l| = m$ then 알고리즘 종료
else if $|l| < m$ then go to Step 3.
Step 3. /* 수행 복잡도 : $O(n^2)$
Reduced Cost Matrix에서 선으로 그려지지 않은 $c_{ij} > 0$들 중에서 최소 비용 $\min c_{ij}$를 찾음.
선으로 그려지지 않은 $c_{ij} > 0$들에 대해 $c_{ij} = c_{ij} - \min c_{ij}$.
2개의 선이 교차된 c_{ij}들에 대해 $c_{ij} = c_{ij} + c_{ij}$.
go to Step 2.

그림 2. Hungarian 알고리즘
Fig 2. Hungarian Algorithm

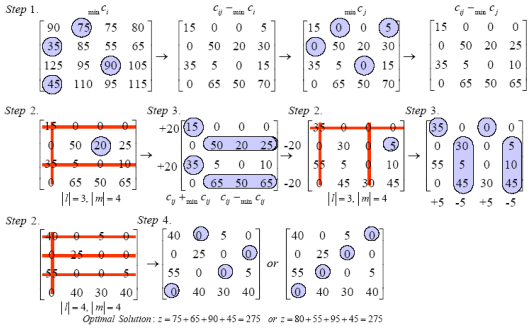


그림 3. A_1 **할당문제의 Hungarian 알고리즘 적용**
Fig 3. Hungarian Algorithm for A_1 **Assignment Problem**

그림 1에 대해 Hungarian 알고리즘을 적용한 결과는 그림 3과 같다. 결국, 최적해 $z = \min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} = 80+55+95+45 = 275$를 얻는다.[9]

Hungarian 알고리즘은 다음과 같은 문제점이 존재한다.

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

(2) 축소된 비용 행렬에서 0을 모두 포함하는 최소한의 선을 긋는 방법을 전산화하기 어렵다.

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

(4) Hungarian 알고리즘의 수행 복잡도는 $O(n^3)$로 실제 $m \geq 7$인 경우 최적해를 도출하기 위해서는 많은 시간이 소요된다.

Ⅲ. 단순 할당 알고리즘

본 장에서는 Hungarian 알고리즘의 수행 복잡도 $O(n^3)$를 $O(n^2)$로 향상시키면서, 균형과 불균형 할당 문제 모두에 즉시 적용할 수 있는 단순한 알고리즘을 제안한다.

그림 1의 A_1 할당 문제에 대해, 그림 4와 같이 각 행(S_i)에서의 최소 비용 $(\min c_i)$을 선택하면 $x_{12}=1$, $x_{21}=1$, $x_{33}=1$, $x_{41}=1$이 배정되어 $z = c_{12}x_{12} + c_{21}x_{21} + c_{33}x_{33} + c_{41}x_{41} = 245$를 얻는다. 이 경우, 모든 열$(D_j)$에서 1개씩만 선택되면 $z = \min z$로 최소의 비용이 소요되는 최적해가 된다. 그러나 이 단계만으로 최적해를 찾는 경우는 거의 없다.

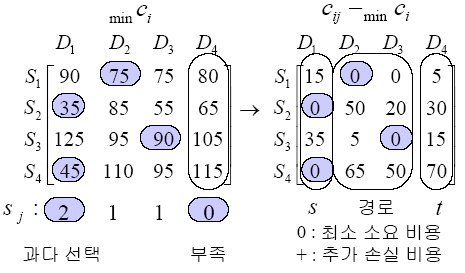


그림 4. 행 최소 비용에서 이동시 손실 비용
Fig 4. The Lost cost moving from Minimum Cost of Row

A_1의 경우도, D_1은 2개, D_2, D_3는 1개씩, D_4는 0개가 선택되었다. 각 행에서의 $\min c_i$ 선택은 $z = \min z$, 모든 공급(행)은 1개씩만 할당 조건은 만족시킨다. 그러나 모든 요구 (열)에서의 1개씩 선택 조건을 만족시키지 못한다. 결국, $z = 245$는 최적해가 아님을 알 수 있다.

할당 문제는 모든 행과 열이 1개씩만 선택해야만 하는 조건을 만족시켜야 한다. 따라서 중복 선택된 열은 1개만 남기고 나머지는 다른 곳으로 이동시켜야 한다. 이때 이동시 손실 비용 (loss Cost)을 최소한으로 증가시켜야만 한다. 즉, D_1의 x_{21}과 x_{41}중 1개가 D_2, D_3, D_4중 어느 한 열로 이동해야만 한다. 이 문제를 해결 가능한 하나의 방법은 $\min\{c_{21}(35),\ c_{41}(45)\}$인 $x_{21}=1$을 D_1에 배정하고, $\max\{c_{21}(35), c_{41}(45)\}$인 $x_{41}=0$으로, 다음 최소 비용 $c_{43}(95)$인 D_3로 이동하여 $x_{43}=1$을 배정한다. 이 경우, D_3가 다시 2개가 선택되었으므로, $\min\{c_{33}(90), c_{43}(95)\}$인 $x_{33}=1$을 D_3에 배정하고, $\max\{c_{33}(90), c_{43}(95)\}$인 $x_{43}=0$으로, 다음 최소 비용 $c_{42}(110)$인 D_2로 이동하여 $x_{42}=1$을 배정한다. 이 경우, D_2가 다시 2개가 선택되어 어느 하나를 이동시켜야만 한다. $\min\{c_{12}(75), c_{42}$ $c_{42}(110)\}$인 $x_{12}=1$을 D_2에 배정하고, $\max\{c_{12}(75), c_{42}($ $110)\}$인 $x_{42}=0$으로, 다음 최소 비용 $c_{44}(115)$인 D_4로 이동하여 $x_{44}=1$을 배정한다. 이 과정을 거치면 모든 행과 열이 1개씩만 선택되는 조건을 만족시킨다. 그러나 $z = 75+35+90+115 = 315$로 최적해 $z = 275$인 최적해를 만족시키지 못한다. 따라서 이 방법은 적용할 수 없다.

본 논문에서는 이 문제를 해결하는 방법으로 $s_j \geq 2$인 D_1을 s (Source), $d_j = odd$인 D_4를 t (destination)이라 하고, s에서 t로의 손실 비용 오름차순으로 도달 가능한 경로를 찾아 손실 비용의 합이 최소가 되는 경로를 선택하는 방법을 적용한다. s에서 t로의 손실 비용 오름차순으로 도달 가능한 경로를 찾아 손실 비용의 합이 최소가 되는 경로를 결정한 결과는 그림 5와 같다.

그림 5에서 손실 비용을 최소화 시킬 수 있는 경로는 S_2가 D_2에 선택된 $x_{21}=1, x_{24}=0$를 D_4로 이동시켜 $x_{21}=0, x_{24}=1$로 하는 경로이다. 결국, S_2가 D_1에 할당된 것을 D_4로 이동시키면 최적해를 찾을 수 있다. 이 경우 S_2는 처음 할당비용 $c_{21}x_{21}=35$이 $c_{24}x_{24}=65$로 30이 증가한다. 결국, $x_{12}=x_{24}=x_{33}=x_{41}=1$, $x_{ij}=0, (\neq x_{12}, x_{24}, x_{33}, x_{41})$으로 할당되어 최적 할당 비용 $z = \min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} = 75+65+90+45 = 275$이 된다. 이 방법은 이론적으로는 매우 타당하지만 중복 선

택된 열이 다수 존재시 s에서 t로의 가능한 경로를 모두 찾아 손실비용 합을 구하는 과정이 어렵다.

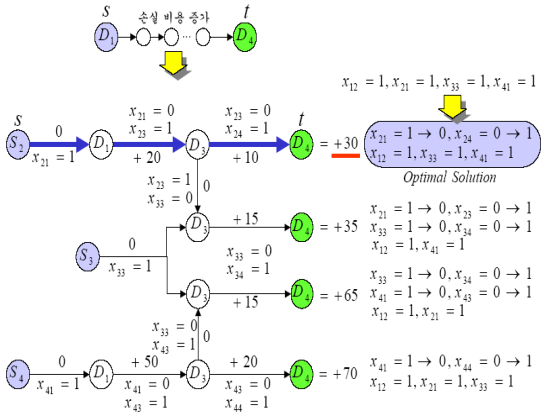


그림 5. s에서 t로의 손실 비용 최소화 경로 결정
Fig 5. The minimum Path of Lost Cost from s to t

본 논문에서는 s에서 t로의 가능한 경로를 모두 찾아 손실비용의 합을 계산하는 복잡한 과정을 채택하지 않고, 단순히 $s_j \geq 2$인 s의 $_{\min}c_i, i=1,2,\cdots$을 t의 최소 비용 $_{\min}t_i$과의 차이를 비교하여 이동 여부를 결정하는 방법을 적용한다. 즉, s에서 $_{\min}c_i, i=1,2,\cdots$가 선택된 경우, t의 비용들 중 최소 비용($_{\min}t_1$)까지 도달하는데 최대 손실이 발생하는 $\max\{(_{\min}t_1 - _{\min}c_1),(_{\min}t_2 - _{\min}c_2),\cdots\}$가 j열에 남고, 나머지 $_{\min}c_i$들을 다음 최소 비용 열로 이동시키는 방법을 적용한다.

제안 알고리즘은 이동 여부를 단순히 결정하며, 알고리즘 구현이 쉬워 "단순 할당 알고리즘 (simple assignment algorithm)"이라 하자. 단순 할당 알고리즘의 상세한 내용은 그림 6에 제시되어 있다.

그림 1의 A_1에 단순 할당 알고리즘을 적용한 과정은 그림 7에 제시되어 있다. 단순 할당 알고리즘은 첫 번째로 각 행의 최소 비용을 선택한 결과 $x_{12}=x_{21}=x_{33}=x_{41}=1$이 할당되어 1열이 2개가 선택되었다. 1열의 $x_{21}=1, x_{41}=1$에 대해 t의 최소 비용과의 차이를 비교하여 $x_{21}=0 \rightarrow x_{23}=1$로 이동하였다. 다음으로 3열에 2개가 선택되어 다시 $x_{23}=1, x_{33}=1$에 대해 t의 최소 비용과의 차이를 비교하여 $x_{23}=0 \rightarrow x_{24}=1$로 이동하였다. 즉, 단지 2회의 이동만으로 최적의 해를 찾는데 성공하였다.

Ⅳ. 알고리즘 적용성 평가

1. 균형 할당 문제

본 절에서는 그림 8의 24개 균형 할당 문제를 대상으로 단순 할당 알고리즘의 적용성을 평가해 본다. 데이터들은 행의 최소 비용을 선택시 $s_j \geq 2$가 되는 j열이 1개인 경우, 2개인 경우와 모든 최소 비용이 1개의 열에 선택된 경우로 구분하였다. $B_1 \sim B_{10}$의 10개 데이터는 $s_j \geq 2$가 되는 j열이 1개 선택된 경우이며, $B_{11} \sim B_{15}$의 5개 데이터는 모든 최소 비용이 1개의 열에 선택된 경우이며, $B_{16} \sim B_{24}$의 9개 데이터는 $s_j \geq 2$가 되는 j열이 2개 선택된 경우이다.

$m \times n$ 행렬
[$m=n$, $m<n$]
$s_j=0, j=1,2,\cdots n$. /* j열에서 선택된 비용 수
for $i=1$ to m /* 수행 복잡도 : $O(n)$
  i 행의 최소 비용 선택, $s_j=s_j+1$. /* 만약 동일 값이 존재하면 처음 값 선택.
end
if $m=n, \forall_j s_j=1$ then 알고리즘 종료 /* 모든 열의 최소비용 1개씩 선택
else if $m<n$, $\exists_j (s_j=1)=\lfloor m \rfloor$ then 알고리즘 종료
else if $\exists_j s_j \geq 2$ then /* 최소 비용이 2개 이상 선택된 열이 존재
  /* 수행 우선 순위 결정 : $s_j \geq 2$인 열들에서 선택된 최대 비용 비교, 최대 비용이 적은 열부터 수행. (Max-Min 우선 수행 기법)
  for $\forall_j s_j=1$ $(m=n)$ or $\exists_j (s_j=1)=\lfloor m \rfloor$ $(m<n)$ /* 수행 복잡도 : $O(n^2)$
    /* i 행 : 선택 최소비용 c_{i0}, 2번째 최소 비용 c_{i1}.
    /* 미 선택 열 : 최소비용 u_{i1}, 두 번째 최소 비용 u_{i2}.
    for $i=1$ to s_j
      $d_i(uc)=u_{i1}-c_{i0}$ /* 미 선택 열 최소비용 - 첫 번째 최소 비용
      $d_i(cc)=c_{i1}-c_{i0}$ /* 2번째 최소비용 - 첫 번째 최소 비용
      $d_i(u2c)=u_{i2}-c_{i0}$ /* 미 선택 열 두 번째 최소비용 - 첫 번째 최소 비용
    end
    $_{\max}d_i(uc)$, $_{\max}d_i(cc)$, $_{\max}d_i(u2c)$ 확인.
    /* 이동 순서 결정 : $_{\max}d_i(uc)$, $_{\max}d_i(cc)$, $_{\max}d_i(u2c)$, $_{\max}c_{i0}$ 순서.
    /* 이동 값 결정 : 손실비용 최대값은 고정, 나머지 비용들을 두 번째 최소비용 (c_{i1})으로 이동.
    if $d_1(uc)=d_2(uc)$ then
      if $d_1(cc)<d_2(cc)$ then c_{10}를 c_{11}으로 이동
      else if $d_1(cc)=d_2(cc)$ then
        if $c_{10}<c_{20}$ then c_{10}를 c_{11}으로 이동
        else if $d_1(cc)=d_2(cc)$ then
          if $c_{10}<c_{20}$ then c_{10}를 c_{11}로 이동
          else if $c_{10}=c_{20}$ then c_{10}를 c_{11}으로 이동
    else if $d_1(uc)<d_2(uc)$ then c_{10}를 c_{11}으로 이동.
    if $d_1(uc)=d_2(uc)=d_3(uc)$ then
      if $d_1(u2c)>d_2(u2c)>d_3(u2c)$ then
        c_{20}, c_{30}를 c_{21}, c_{31}으로 이동
    else if $d_1(uc)>d_2(uc)=d_3(uc)$ then c_{20}, c_{30}를 c_{21}, c_{31}으로 이동
    else if $d_1(uc)>d_2(uc)>d_3(uc)$ then c_{20}, c_{30}를 c_{21}, c_{31}으로 이동.
    if $d_1(uc)>d_2(uc)>d_3(uc)>d_4(uc)$ then c_{20}, c_{30}, c_{40}를 c_{21}, c_{31}, c_{41}으로 이동.
  end

[$m>n$]
행 (i)을 열 (j)로, 열 (j)을 행 (i)으로 바꾸어 알고리즘 수행.
/* 수행 우선 순위 결정 : $s_i \geq 2$인 행들의 최대 비용 비교, 최대 비용이 큰 행부터 수행. (Max-Max 우선 수행 기법)

그림 6. 단순 할당 알고리즘
Fig 6. Simple Assignment Algorithm

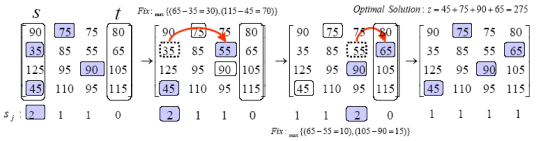


그림 7. A_1 할당 문제의 단순 할당 알고리즘 적용 최적해
Fig 7. The Optimal Solution of Simple Assignment Algorithm for A_1

B_1[5], B_2[3,9], B_3[10], B_4[11], B_5[12], B_6, B_7[13], B_8[14], B_9[15], B_{10}[16], B_{11}, B_{12}[13], B_{13}[17], B_{14}[18], B_{15}[17], B_{16}[13], B_{17}[19], B_{18}, B_{19}[13], B_{20}[17], B_{21}[13], B_{22}[20], B_{23}, B_{24}[13]에서 인용되었다.

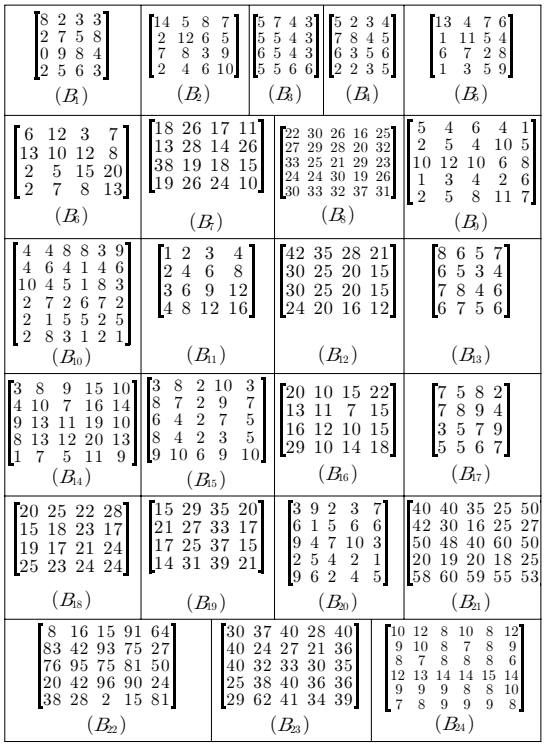
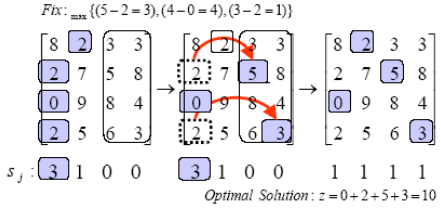
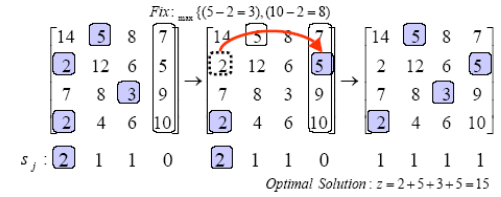


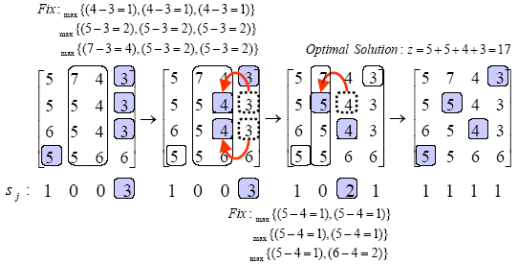
그림 8. 균형 할당 실험 데이터
Fig 8. Experimental data for Balance Assignment Problem



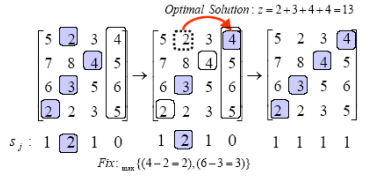
(a) B_1 할당 문제



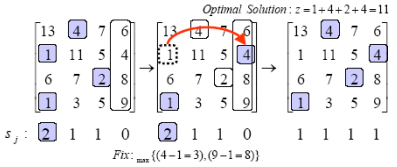
(b) B_2 할당 문제



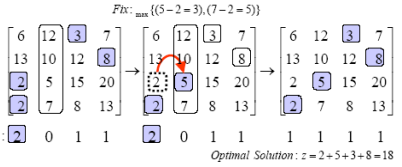
(c) B_3 할당 문제



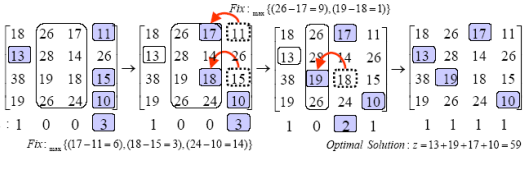
(d) B_4 할당 문제



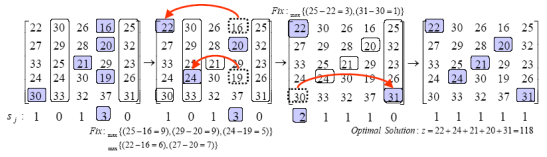
(e) B_5 할당 문제



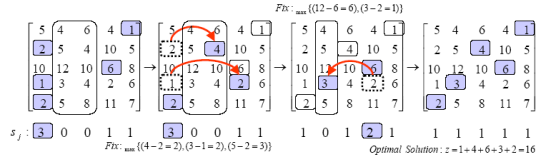
(f) B_6 할당 문제



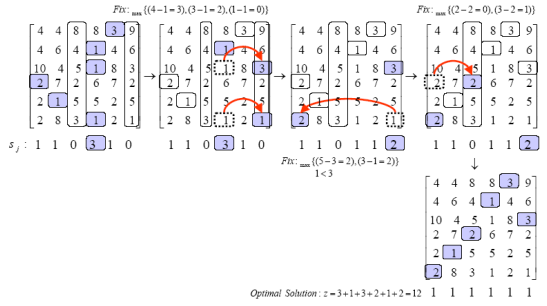
(g) B_7 할당 문제



(h) B_8 할당 문제



(i) B_9 할당 문제

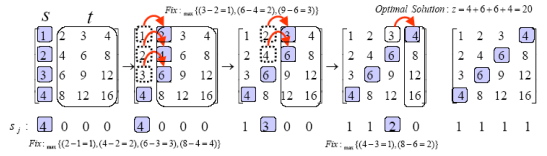


(j) B_{10} 할당 문제

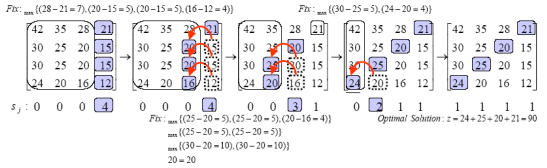
그림 9. 2개 이상 선택된 열이 1개인 경우
Fig 9. Case of No. 1 Column for selects more than 2

$s_j \geq 2$가 되는 j열이 1개인 경우인 $B_1 \sim B_{10}$ 의 10개 데이터에 단순 할당 알고리즘을 적용한 결과는 그림 8)에 제시되어 있다. B_1, B_2, B_4는 1회 이동으로, B_3, B_7, B_8, B_9는 2회 이동으로, B_{10}은 3회 이동으로 최적해를 찾는데 성공하였다.

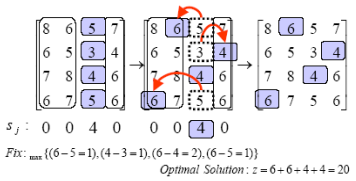
모든 최소 비용이 1개의 열에 선택된 경우인 $B_{11} \sim B_{15}$의 5개 데이터에 대해 단순 할당 알고리즘을 적용한 결과는 그림 10에 제시되어 있다. B_{11}, B_{12}는 3회 이동으로, B_{13}, B_{15}는 1회 이동으로, B_{14}는 4회 이동으로 최적해를 찾는데 성공하였다.



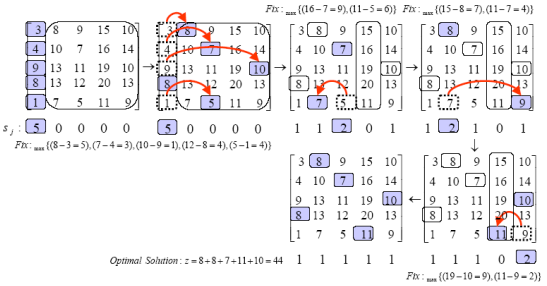
(a) B_{11} 할당 문제



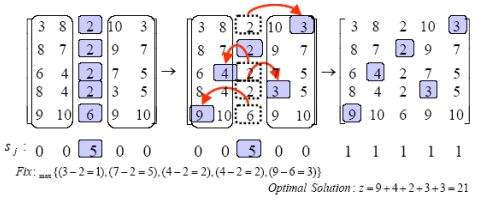
(b) B_{12} 할당 문제



(c) B_{13} 할당 문제



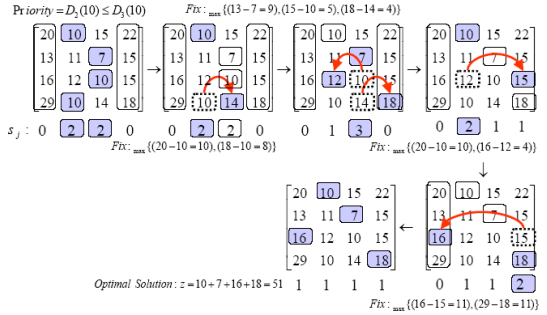
(d) B_{14} 할당 문제



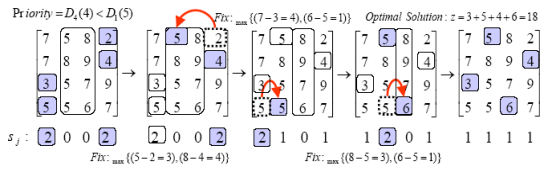
(e) B_{15} 할당 문제

그림 10 1개 열이 모두 선택된 경우
Fig 10. Case of All selected in 1 Column

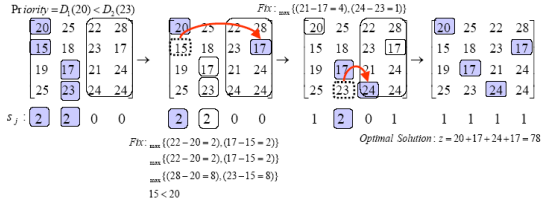
$s_j \geq 2$가 되는 j열이 2개 선택된 경우의 $B_{17} \sim B_{24}$의 8개 데이터에 대해 단순 할당 알고리즘을 적용한 결과는 그림 11에 제시되어 있다. B_{16}, B_{19}, B_{23}은 4회 이동으로, B_{17}은 3회 이동으로, $B_{18}, B_{20}, B_{21}, B_{22}, B_{24}$는 2회 이동으로 최적해를 찾는데 성공하였다.



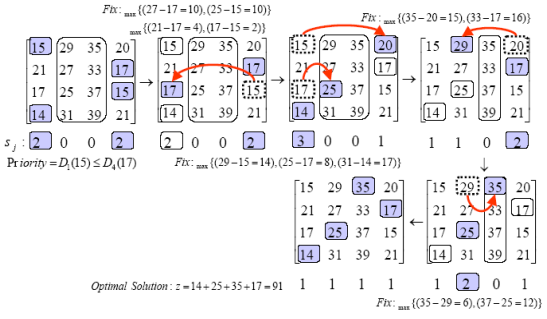
(a) B_{16} 할당 문제



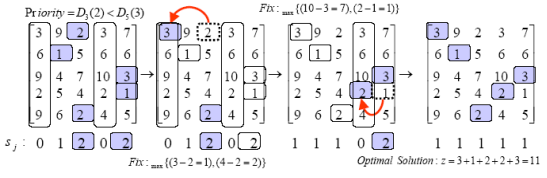
(b) B_{17} 할당 문제



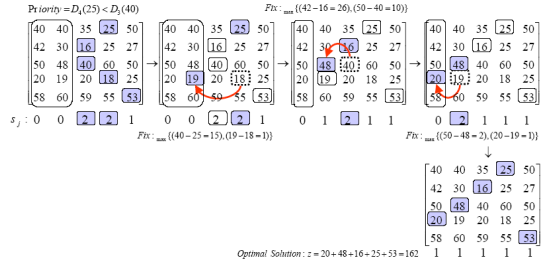
(c) B_{18} 할당 문제



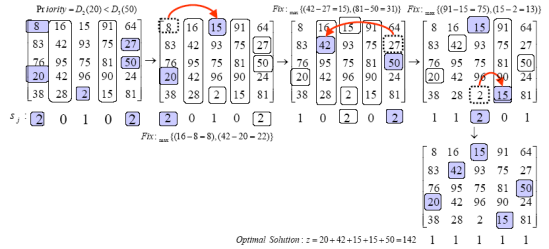
(d) B_{19} 할당 문제



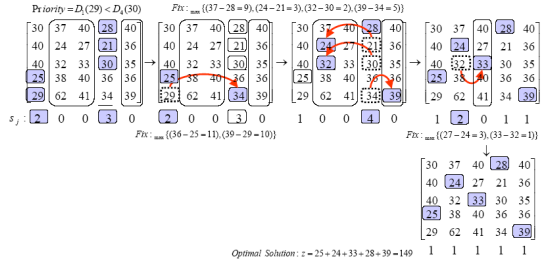
(e) B_{20} 할당 문제



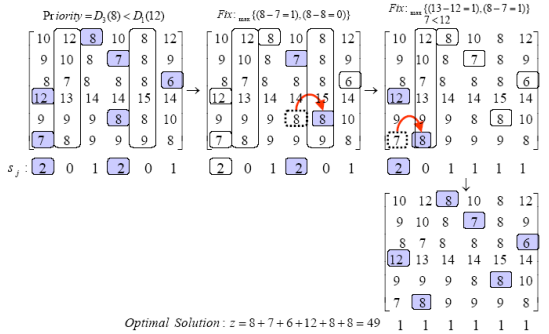
(f) B_{21} 할당 문제



(g) B_{22} 할당 문제



(h) B_{23} 할당 문제



(i) B_{24} 할당 문제

그림 11. 2개 이상 선택된 열이 2개인 경우
Fig 11. Case of More Than 2 Column for selects more than 2

24개의 균형 할당 문제에 대해 모두 Hungarian 알고리즘을 적용하여 최적의 해를 찾았다. 또한, 제안된 단순 할

당 알고리즘도 모두 최적의 해를 도출하는데 성공하였다.

2. 불균형 할당 문제

본 절에서는 그림 12의 7개 불균형 할당 문제를 대상으로 단순 할당 알고리즘의 적용성을 평가해 본다. UB_1 [17], UB_2 [9,19,21], UB_3, UB_4, UB_5 [13] UB_6 [22], UB_7 [4]에서 인용되었다. 7개 불균형 할당 문제에 대해 단순할당 알고리즘을 적용한 결과는 그림 13에 제시하였다.

7개의 불균형 할당 문제에 대해 제안된 단순 할당 알고리즘은 모두 최적의 해를 찾는데 성공하였다. UB_2, UB_3, UB_4는 1회, UB_2는 0회, UB_5, UB_6는 2회, UB_7은 3회 이동을 수행하였다. UB_7 [4]는 원문에서 유전자 알고리즘을 적용한 결과 최적의 해로 $z=323$이라 제시하고 있다. 반면에 제안된 단순 할당 알고리즘은 단 3회 이동으로 최적해 $z=179$를 찾는데 성공하였다. 결국, 유전자 알고리즘으로 할당문제를 해결하려고 시도하였으나 실패하였음을 알 수 있다.

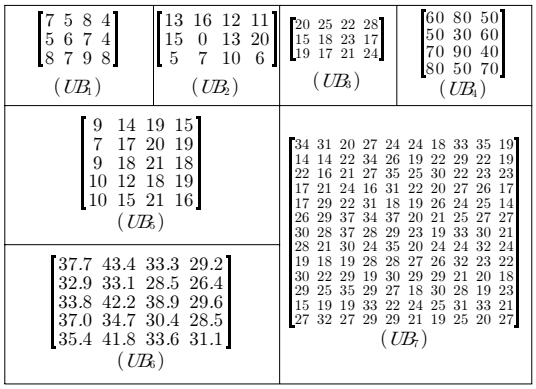
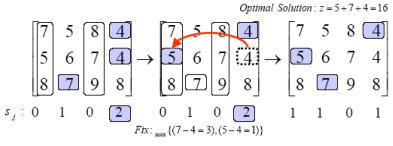
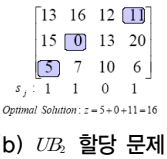


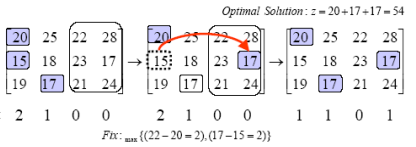
그림 12. 불균형 할당 실험 데이터
Fig 12. Experimental Data for Unbalanced Assignment Problem



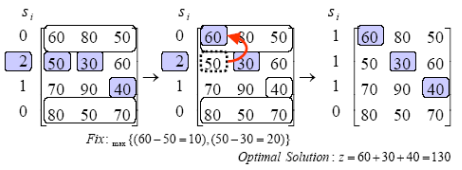
(a) UB_1 할당 문제



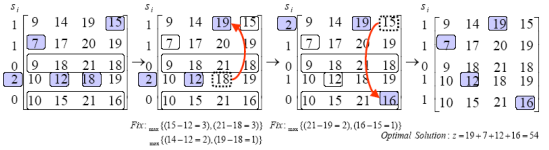
(b) UB_2 할당 문제



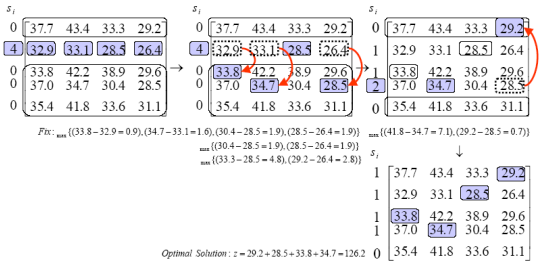
(c) UB_3 할당 문제



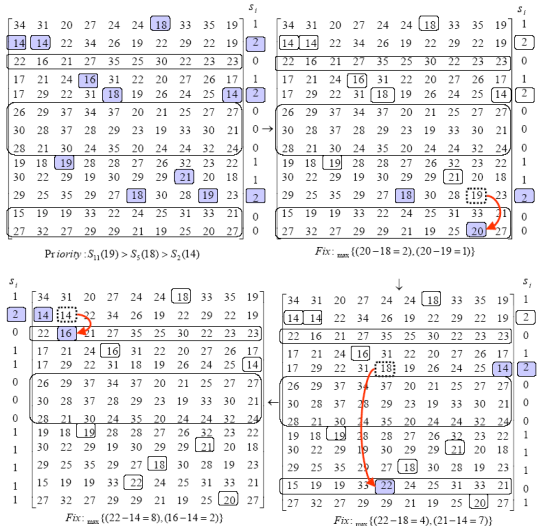
(d) UB_4 할당 문제



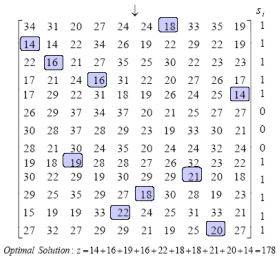
(e) UB_5 할당 문제



(f) UB_6 할당 문제



(g) UB_7 할당 문제



(g) UB_7 할당 문제

그림 13 불균형 할당 문제의 단순 할당 알고리즘 적용
Fig 13. Simple Assignment Algorithm for Unbalanced Assignment Problems

3. 알고리즘 성능 평가

본 논문에서 실험에 적용된 균형 할당 25개와 불균형 할당 7개 데이터에 대해 이동 횟수를 평가한 결과는 표 1에 제시되어 있다. 단순 할당 알고리즘은 중복 선택된 할당 이동을 최대 4회, 대부분은 1회나 2회 수행으로 최적해를 간단히 얻을 수 있었다.

표 1. 단순 할당 알고리즘의 이동 횟수
Table 1. The Number of Moving for Simple Assignment Algorithm

구분	행렬	데이터 수	이동 횟수				
			0회	1회	2회	3회	4회
균형 할당	4×4	15	0	6	4	3	2
	5×5	8	0	1	5	0	2
	6×6	2	0	0	1	1	0
불균형 할당	3×4	3	1	2	0	0	0
	4×3	1	0	1	0	0	0
	5×4	2	0	0	2	0	0
	13×10	1	0	0	0	1	0
계		32	1	10	12	5	4

Ⅳ. 결론 및 향후 연구과제

본 논문에서는 할당 문제의 최적해를 찾는 단순한 알고리즘을 제안하였다.

제안된 알고리즘은 각 행에서 최소 비용을 선택하여 2개 이상이 선택된 열을 출발 (s)로, 선택되지 않은 열을 목적지 (t)로 설정하였다. 출발의 선택된 최소 비용에서 목적지의 최소 비용까지의 차이 (손실비용)이 최대가 되는 행의 비용은 남고, 다른 모든 행의 비용은 다음 최소 비용으로 이동시키는 방법을 적용하였다. 제안된 방법은 출발에서 목적지까지의 최소 손실비용 증가를 갖는 비용이 이동하는 방법이다. 이는 가능한 경로들 중 경로 길이 합이 최소가 되는 경로로 비용을 이동시키는 방법이라 할 수 있다.

제안된 알고리즘을 25개의 균형 할당 문제와 7개의 불균형 할당 문제에 적용한 결과 중복 선택된 할당을 평균 1 또는 2회 이동시켜 최적해를 찾는데 성공하였다.

제안된 알고리즘은 대표적인 Hungarian 알고리즘의 수행 복잡도 $O(n^3)$을 $O(n^2)$로 개선하였다. 또한, 균형뿐 아니라 불균형 할당 문제에 Dummy Point를 추가하는 과정을 거치지 않고 주어진 문제로부터 직접 최적해를 찾을 수 있는 장점을 갖고 있다.

본 논문에서는 수송 문제의 특별한 경우인 공급량과 요구량이 모두 1인 할당문제만을 연구하였다. 추후 공급량과 요구량이 모두 다른 보다 복잡한 수송문제를 간단히 해결할 수 있는 알고리즘을 연구할 예정이다.

참 고 문 헌

[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] R. Burkard, M. D. Amico, and S. Martello, *"Assignment Problems*, http://www.assignmentproblems.com/HA4 applet.htm, SIAM Monographs on Discrete Mathematics and Applications, 2006.

[6] S. U. Lee, "The Optimal Algorithm for Assignment Problem," Journal of Korea Society of Computer Information, Vol. 17, No. 9, pp. 139-147, Sep, 2012.

[7] S. U. Lee, "A Reverse-delete Algorithm for Assignment Problem," Journal of Korean Institute of Information Technology, Vol. 10, No. 8, pp. 117-126, Aug, 2012.

[8] 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.

[9] 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.

[10] 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.

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

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

[14] 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.

[15] J. Havlicek, "Introduction to Management Science and Operation Research," http://orms.czu.cz/text/transproblem. html, 2007.

[16] 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.

[17] 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.

[18] K. Wayne, "Algorithm Design," http://www.cs.princeton.edu/~wayne/kleinberg-tardos/07assignment.pdf, 2005.

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

[20] 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.

[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.

저자 소개

이 상 운

- 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

<관심분야 :소프트웨어 프로젝트 관리, 개발 방법론, 소프트웨어 척도, 분석과 설계 방법론, 시험 및 품질보증, 소프트웨어 신뢰성, 신경망, 뉴로-퍼지, 그래프 알고리즘>

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