Journal of the Society of Korea Industrial and Systems Engineering
Vol. 34, No. 3, pp.90－96, September 2011.

최대후회 최소화 임계 경로 탐색 알고리듬

강준규*† · 윤협상**

*성결대학교 산업경영공학부
**대구가톨릭대학교 경영정보학과

A Heuristic Algorithm to Find the Critical Path Minimizing the Maximal Regret

Jun-Gyu Kang*† · Hyoup-Sang Yoon**

*Department of Industrial and Management Engineering, Sungkyul University
**Department of Management Information Systems, Catholic University of Daegu

Finding the critical path (or the longest path) on acyclic directed graphs, which is well-known as PERT/CPM, the ambiguity of each acr's length can be modeled as a range or an interval, in which the actual length of arc may realize. In this case, the min-max regret criterion, which is widely used in the decision making under uncertainty, can be applied to find the critical path minimizing the maximum regret in the worst case. Since the min-max regret critical path problem with the interval arc's lengths is known as *NP-hard*, this paper proposes a heuristic algorithm to diminish the maximum regret. Then the computational experiments shows the proposed algorithm contributes to the improvement of solution compared with the existing heuristic algorithms.

Keywords : Min-max Regret, Interval Data, Robust Shortest Path

1. 서 론

DAG(directed acyclic graph) 상에서의 최장경로문제는 PERT/CPM(Program Evaluation and Review Techniques/ Critical Path Method)으로 일반적으로 잘 알려져 있는데, PERT에서 가지(arc)의 길이는 각 활동(activity)의 소요시간을 나타내고, 프로젝트의 완료시간을 알기 위해 임계경로(critical path) 혹은 최장경로(longest path)를 찾는다. 현실적으로 각 활동(activity)의 소요시간에는 불확실성(uncertainty, imprecision)이 존재하여, PERT/CPM에서는 그 소요시간의 불확실성을 확률적으로 베타분포를 따른다고 가정하고 있으나, 실제 이 불확실성은 정보의 부족과 같은 이유로 통계적 모형으로 수립되기 어려운 경우가 많다. 이에 Kouvelis and Yu[12]는 네트워크상의 가지 길이의 불확실성을 모형화하는 가장 간단한 방법으로, 각 가지의 길이가 범위(interval data)의 형태로 주어진 경우로 가정하고, 이를 해결하기 위해 최대후회 최소화 기준(min-max regret criterion)의 적용을 제안한다.

Karaşan et al.[10]은 가지의 길이가 범위(interval)로 주어진 경우 최대후회 최소화 최단경로문제에 대한 혼합정수계획모형을 제시하고, 임계경로 상에 존재할 가능성

논문접수일 : 2011년 07월 19일　게재확정일 : 2011년 09월 10일
† 교신저자 kangjungyu@hotmail.com
※ 이 논문은 성결대학교 연구비지원에 의한 연구과제임.

이 없는 dominated arc를 판별하기 위한 pre-processing 기법을 제안하였다. 이후 Montemanni and Gambardella [13]는 Karaşan et al.[10]의 MILP 모형에 기초하여 수행시간을 단축할 수 있는 branch-and-bound algorithm을 제안했고, 이어서 Montemanni and Gambardella[14]는 Karaşan et al.[10]의 MILP 모형에 Benders decomposition approach를 이용하여 branch-and-bound algorithm의 수행속도를 향상시키려 하였다. 반면에, Kang and Brissaud[8]은 동적계획법 모형에 기반을 둔 최적해 탐색 알고리듬을 제안하였다. Chanas and Zieliński[2]에 의하면 Karaşan et al.[10]이 제안한 dominancy의 판별문제는 strongly NP-complete이며, Averbakh and Lebedev[1] and Zieliński[15]에 의하면 DAG상에서 최대후회 최소화 임계경로문제는 NP-hard이다.

주어진 문제가 *NP-hard*이므로 현실적인 접근법으로써, Kang et al.[9]과 Montemanni and Gambardella[13]는 k-최장경로(최장경로부터 k번째 최장경로까지)를 생성하여 그 중 최대후회가 최소인 경로를 선택하는 휴리스틱 알고리듬을 제안했는데, Kang et al.[9]은 k-최장경로를 생성하기 위해 branch-and-cut 방법을 이용하고, Montemanni and Gambardella[13]는 Eppstein[4]의 알고리듬을 이용하여 k-최단경로를 비교적 효율적으로 찾았다. 그러나 Chanas and Zieliński[2]가 지적한 대로, k개의 최적해가 될 가능성이 있는(non-dominated) 경로를 생성하는 문제 역시 *NP-hard*이므로 large-scale 최대후회 최소화 임계경로 문제에서 효과적일 것으로 기대할 수 없다. Kasperski and Zielinski[11]는 극단적으로 간단한 근사해법으로 범위의 중간값을 사용하여 전통적인 확정적 문제(deterministic problem)의 알고리듬으로 해를 찾는 방법을 제시하기도 하였다. 최근에 Conde [3] and Kang[6]은 동적계획법 모형에 기반을 두고 이를 단순화하여 Dijkstra's 알고리듬을 응용한 휴리스틱 알고리듬을 제안하였고, 이후 Kang[7]은 Conde[3]의 알고리듬을 개선한 알고리듬을 제안하였다. 본 연구는 현재까지 알려진 휴리스틱 알고리듬 중 해의 정확도가 가장 높은 Kang[7]의 알고리듬을 바탕으로 해의 성능을 개선 방법을 제안한다.

본 논문은 구성은 다음과 같다. 제 2장에서 최대후회최소화 임계경로문제와 Kang[7]의 알고리듬을 소개하고 제 3장에서 이를 기반으로 해의 개선 방법을 제안하며, 제 4장에서는 제안된 개선 방법의 효과를 검증하기 위한 컴퓨터 모의실험에 사용된 네트워크 모형과 그 결과를 제시하고, 그리고 마지막 제 5장에 결론이 이어진다.

2. 문제 및 기존 알고리듬

가지의 가중치가 범위로 주어진 네트워크를 다음과 같이 모형화한다. 임의의 DAG G = (N, A), N은 마디(node)의 집합, |N| = n, 그리고, A는 가지(arc)의 집합, |A| = m상에서 가지의 가중치가 다음과 같이 주어진 경우를 가정하자. 그래프 상의 임의의 가지(i, j)의 가중치 c_{ij}는 그 값의 범위(interval range, $c_{ij} \in [l_{ij}, u_{ij}]$)만이 알려져 있다. 임의의 c_{ij}는 구간 $[l_{ij}, u_{ij}]$에서 어느 값이던 취할 수 있으며, 이 값은 다른 가지의 가중치와는 독립적이다. 본 연구에서는 가지의 가중치가 범위로 주어진 DAG를 편의상 인터벌 네트워크(interval network)라고 부른다. 각 가지의 값 c_{ij}가 특정 값을 취한 것을 시나리오 S라 하면, 가능한 모든 시나리오 집합의 개수는 모든 interval의 Cartesian product이므로, $|\Delta| = 2^n$이다.

불확실성이 없는 DAG상에서의 임계경로 문제를 P라 하면, 문제 P의 최대후회 최소화 버전은 다음과 같이 정의된다.

Notation 1

Ω : a set of all feasible solutions

$\mathbf{x}$: a solution

$\mathbf{y}$: the optimal solution under scenario S

c^S : cost realization under scenario S

$R_{\max}(\mathbf{x})$: the maximum regret of a solution $\mathbf{x}$

Min-Max Regret P :

$$\min_{x \in \Omega} R_{\max}(\boldsymbol{x}) = \min_{x \in \Omega} \max_{\substack{y \in \Omega \\ S \in \Delta}} (c^S \boldsymbol{y} - c^S \boldsymbol{x}) \quad (1)$$

식 (1)에서 시나리오 S하에서, 해 $\mathbf{x}$의 경로길이가 $c^S\mathbf{x}$라면 최적해 $\mathbf{y}$의 경로길이는 $c^S\mathbf{y}$이며, 후회(regret)는 해 $\mathbf{x}$의 목적함수의 값과 최적해 $\mathbf{y}$의 목적함수의 값의 차이이고, 경로 $\mathbf{x}$의 최대후회 $R_{\max}(\mathbf{x})$는 가능한 모든 시나리오 하에서의 후회 중 최대값이다.

<그림 1>은 기존 연구[7]의 휴리스틱 알고리듬의 pseudo-code를 나타내는데, 그에 관련된 기호는 다음과 같다.

Notation 2

c^u, c^l(Scenario u and l) : 모든 가지의 가중치가 upper bound 또는 lower bound로 설정된 경우

$c^s(x)$(scenario induced by path $\mathbf{x}$) : 임의의 경로 $\mathbf{x}$가 주어진 경우 c-consistency에 의해 결정되는 가지의 가중치 값, 즉, 경로 x상의 모든 가지의 가중치는

lower bound로, 그렇지 않은 가지의 가중치는 upper bound가 되는 경우,
Φ : 최악대응경로에 제외되어야 하는 가지의 집합
x^j : 마디 0에서 마디 j까지의 최대후회 최소화 최장경로
$x^{ij} = x^j \cup$ 가지(i, j)
y^i, y^{ij} : 경로 x^j와 x^{ij}의 최악대응경로.
r_i, r_{ij} : 경로 x^j와 x^{ij}의 최대 후회.

최대후회를 최소화하는 경로를 찾는 과정에서 주어진 경로의 최대후회는 다음과 같이 구할 수 있다. 두 경로 $\mathbf{x}$와 $\mathbf{y}$가 주어진 경우, 편차를 최대화하는 목적함수의 계수 c^{ij} *는 $\mathbf{y}^{ij}$-$\mathbf{x}^{ij} < 0$이면 l^{ij}이고, 아니면 u^{ij}이 되는데, 이를 c-consistency라 한다[5]. 즉, 주어진 경로 $\mathbf{x}$에 포함된 가지의 길이는 그 lower bound(l^{ij})값으로 설정하고, 나머지 가지의 길이는 그 upper bound(u^{ij})로 설정한다. 이를 $c^s(\mathbf{x})$(scenario induced by path $\mathbf{x}$)라 하며, 이 때 최적해($\mathbf{y}$)는 경로 $\mathbf{x}$의 최악대응경로라 하고, 해 $\mathbf{x}$와 $\mathbf{y}$의 목적값의 차이(= 경로의 길이의 차이)는 주어진 경로 $\mathbf{x}$의 최대후회이다.

기존의 알고리듬[7]은 Dijkstra's 알고리듬과 기본적으로 유사한데, 출발마디(마디 0)에서 마디 i까지의 최대후회 최소화 경로 $\mathbf{x}^i$를 알 때, 마디 j까지의 최대후회 최소화 경로 $\mathbf{x}^j$는 다음과 같다.

$$ij = \underset{\text{arc } ij}{\arg\min}\, r^{ij} \qquad (2)$$
$$x^j = x^{ij}$$

그러므로 기존 알고리듬에서 주어진 경로 $\mathbf{x}$의 최악대응경로와 최대후회를 시나리오 $\mathbf{c}^s(\mathrm{x})$하에서의 최장경로문제를 풀어 쉽게 찾을 수 있고, 시나리오 $\mathbf{c}^s(\mathrm{x})$하에서의 목적함수의 계수는 가 제시한 c-consistency에 의해 결정된다[5].

기존의 알고리듬을 간단히 설명하면 다음과 같다. <그림 1>에서 2행과 7행의 $\mathrm{Max}_y[\mathbf{c}^u\mathbf{y}]$와 $\mathrm{Max}_y[\mathbf{c}^s(\mathbf{x}^{ij})\mathbf{y}]$는 주어진 경로 $\mathbf{x}^0$와 $\mathbf{x}^{ij}$의 최악대응경로를 찾기 위하여, DAG상에서 Dijkstra's 알고리듬을 변형하여 최장경로를 구하는 과정이다. 이 때 네트워크 상의 각 가지의 가중치 c_{ij}는 c-consistency에 의해 결정되는데, 주어진 경로 $\mathbf{x}^0$와 $\mathbf{x}^{ij}$상의 각 가지의 가중치는 $c_{ij} = l_{ij}$, 그 외의 가지들에 대해서는 $c_{ij} = u_{ij}$로 정한다. 알고리듬은 step 0~1에서 초기화를 하고, step 2에서 시작마디의 최악대응경로(시나리오 u 하에서 최장경로)를 구한다. Step 3에서는 네트워크 상의 마디들을 순회하고, step 4에서 각 마디로 들어오는 마디들을 확인하여 이전 마디로부터 주어진 최대후회최소화 최장경로를 연장한다. 이 때, 가지(i, j)가 $\mathbf{y}(\mathbf{x}^i)$에 포함되지 않으면(step 6) reduction rule에 의해 최대후회가 간단히 구해지고(step 7~8), 포함되면 Dijkstra's 알고리듬을 이용하여 주어진 경로($\mathbf{x}^{ij}$)의 최악대응경로($\mathbf{y}(\mathbf{x}^{ij})$)를 구한다(step 9~11). 이 때 구해진 최대후회가 기존의 최대후회보다 작으면 최대후회가 가장 작은 경로($\mathbf{x}^j$)를 현재의 경로($\mathbf{x}^{ij}$)로 갱신한다(step 12~13).

```
Procedure : Algorithm MinMaxRegret(G = (N, A), c_ij∈[l_ij, u_ij]) for each arc (i, j)∈A, source)
 0    for each node j∈ N                                          // Initializations
 1        r_j : = infinity                                       // all maximum regret are set to infinity
 2    x^0 = NULL, r_0 = Max_y[c^u y], let y^0 be the optimal solution of Max_y[c^u y]   // current node = source node
 3    for each node j∈N                                          // The main loop
 4        for each i∈N such that (i, j)∈A, let                   // for each input arc of nodej
 5            x^ij : = x^i ∪ (i, j)                              // extends path
 6            if arc(i, j)∉y^i
 7                r_ij : = r_j − l_ij
 8                y^ij : = y^i
 9            else
10                r_ij : = Max_y[c^s(x^ij) y] - c^l x^ij
                                  // solve the longest path problem under the scenario induced by x^ij
11                y^ij : = the optimal solution of Max_y[c^s(x^ij) y]   // the worst-case alternative of x^ij
12            if r_ij<r_j
13                r_j : = r_ij, x^j:=x^ij, y^j:=y^ij              // update minmax regret path until node j
```

<그림 1> Pseudo-code of the Kang's algorithm[7]

3. 휴리스틱 알고리듬의 개선

Kang[7]의 알고리듬이 Conde[3]의 알고리듬과 다른 핵심적인 부분은 step 9~11인데, 마디 0에서 마디 j까지의 최대후회 최소화 최장경로의 대안 중 하나인 $\mathbf{x}^{ij}$의 최악대응경로($\mathbf{y}^{ij}$)를 필요한 순간에 재계산함으로써 해의 품질을 향상한다. 그러나 임의의 마디 i까지의 최대후회최소화 최장경로 $\mathbf{x}^{i}$와 그 최악대응경로 $\mathbf{y}(\mathbf{x}^{i})$를 알 때, 가지$(i, j)$가 $\mathbf{y}(\mathbf{x}^{i})$에 포함되지 않으면(즉, step 6이 성립하면) 경로 $\mathbf{x}^{ij}$는 최대후회최소화 경로이나, 그렇지 않으면(step 6이 성립하지 않으면) 경로 $\mathbf{x}^{ij}$의 최악대응경로($\mathbf{y}^{ij}$)를 구해도 경로 $\mathbf{x}^{ij}$가 마디 j까지의 최악대응경로($\mathbf{y}^{ij}$)에 대응하는 최적의 경로인지 알 수 없다. 즉, 경로($\mathbf{y}^{ij}$)가 최악대응경로가 되는 마디 0에서 마디 j까지의 경로는 2개 이상 존재할 수 있으며, 우리는 이 중 최대후회가 최소인 경로를 선택하여야 한다. 그러나 주어진 최악대응경로 $\mathbf{y}$에 대해 이를 최악대응경로로 삼는 경로 $\mathbf{x}$를 찾는 것은 *NP*-complete이므로[2], 최악대응경로 $\mathbf{y}$를 최악대응경로로 삼는 모든 경로 중에서 최대후회가 최소인 경로를 선택하는 문제 역시 *NP*-hard일 것으로 보인다. 그러므로 $\mathbf{x}^{ij}$의 새로운 대안은 아래와 같다.

Notation 3

Ψ : a set of all feasible solutions, from node 0 to node j, of which arcs $\notin \mathbf{y}^{ij}$.

To find an alternative

$$\max_{z \in \Psi} c^{l} z \tag{3}$$

식 (3)의 해 $\mathbf{z}$는 최악대응경로($\mathbf{y}^{ij}$)와 겹치지 않는 마디 0에서 마디 j까지의 모든 경로 중에서 최악의 경우(가지의 길이를 c^{l} 값으로 설정) 그 길이 가장 긴 경로이며, 이는 $\mathbf{y}^{ij}$를 최악대응경로로 삼는 경로들 중에서 $\mathbf{y}^{ij}$와 겹치지 않는 경로들 중에서는 최대후회가 가장 작은 경로이며, 식 (3)에서 구한 경로 $\mathbf{z}$와 기존의 경로 $\mathbf{x}^{ij}$ 중 그 최대후회가 작은 것을 선택하며, 그 최대후회는 다음과 같다.

To select the less-regretable alternative

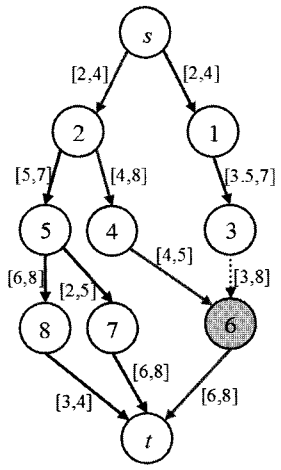
$$r^{ij} = \min(r_{\max}(x^{ij}),\ R_{\max}(z)) \tag{4}$$

위의 식 (3)과 식 (4)에 언급된 과정은 <그림 1>의 step 11과 step 12 사이에 삽입된다. 제안한 알고리듬을 <그림 2>에 나타낸 10개의 마디를 가진 간단한 인터벌 네트워크의 사례로 설명하면 다음과 같다.

마디 3까지의 최적경로($\mathbf{x}^{3}$)는 경로 $\{s\text{-}1\text{-}3\}$이며, 이때 최악대응경로 $\mathbf{y}(\mathbf{x}^{3})$는 경로 $\{s\text{-}2\text{-}4\text{-}6\text{-}t\}$이다. 현재 마디 3에서 마디 6으로 경로를 확장하면, <그림 2>에서 점선으로 표시된 가지 (3, 6)을 포함한 경로 $\mathbf{x}^{36} = \mathbf{x}^{3} \cup (3, 6) = \{s\text{-}1\text{-}3\text{-}6\}$이 되고(<그림 1>의 step 5), step 6의 조건이 만족하므로 $\mathbf{y}(\mathbf{x}^{34}) = \mathbf{y}(\mathbf{x}^{3}) = \{s\text{-}2\text{-}4\text{-}6\text{-}t\}$이다. 이때 최대후회는 $r_{36} = 16.5$이다. 마디 6까지의 다른 경로 $\mathbf{x}^{46} = \{s\text{-}2\text{-}4\text{-}6\}$의 최악대응경로는 $\mathbf{y}(\mathbf{x}^{46}) = \{s\text{-}1\text{-}3\text{-}6\text{-}t\}$이고, 이때 최대후회는 $r_{46} = 17$이므로, 마디 6까지의 최적경로는 $\mathbf{x}^{6} = \mathbf{x}^{36} = \{s\text{-}1\text{-}3\text{-}6\}$이고, 최악대응경로는 $\mathbf{y}(\mathbf{x}^{6}) = \{s\text{-}2\text{-}4\text{-}6\text{-}t\}$, 최대후회는 $r_{6} = r_{36} = 16.5$이다. 다음 단계로 $\mathbf{x}^{6t} = \mathbf{x}^{6} \cup (6, t) = \{s\text{-}1\text{-}3\text{-}6\text{-}t\}$을 구하면, 가지$(6, t)$는 $\mathbf{y}(\mathbf{x}^{6}) = \{s\text{-}2\text{-}4\text{-}6\text{-}t\}$에 포함되므로, step 6의 조건을 만족하지 못한다. 그러므로 알고리듬은 step 10으로 이동하여 경로 $\mathbf{x}^{6t}$의 최악대응경로를 찾는다. 경로 $\mathbf{x}^{6t}$ 상의 모든 가지의 길이는 최소값으로, 나머지 모든 가지의 길이는 최대값으로 설정하여 최장경로를 찾으면 그 경로는 $\mathbf{y}(\mathbf{x}^{6t})$이며, 여기서 $\mathbf{y}(\mathbf{x}^{6t}) = \{s\text{-}2\text{-}5\text{-}7\text{-}t\}$이고, $r_{6t} = 9.5$이다. 기존에 알고리듬에 더하여 식 (3)에 의해 경로 $\mathbf{y}(\mathbf{x}^{6t}) = \{s\text{-}2\text{-}5\text{-}7\text{-}t\}$를 최악대응경로로 삼는 경로는 경로 $\{s\text{-}2\text{-}4\text{-}6\text{-}t\}$와 $\{s\text{-}1\text{-}3\text{-}6\text{-}t\}$이고 이 중 최대후회가 작은 경로는 $\{s\text{-}2\text{-}4\text{-}6\text{-}t\}$로 최대후회는 6이다. 식 (4)에 의해 최대후회는 다음과 같다.

$$r^{6t} = min(\ R_{\max}(\mathbf{x}^{6t}),\ R_{\max}(s\text{-}2\text{-}4\text{-}6\text{-}t)\ )$$
$$= min(\ 9.6,\ 6\ ) = 6$$

그러므로 최대후회최소화 부분경로 $\mathbf{x}^{6t} = \{s\text{-}2\text{-}4\text{-}6\text{-}t\}$로 갱신되고, 이 경로의 최대후회는 9.6에서 6으로 감소된다.



<그림 2> An Example Interval Network[7]

<표 1> Summary of Computational Experiments

				Computation Time(sec.)						GAP				Correct %	
				MIP		Original heuristic		Improved algorithm		Original heuristic		Improved algorithm		Original	Improved
layers	width	c	d	mean	STD	mean	STD	mean	STD	mean	STD	mean	STD	heuristic	algorithm
50	2	10	0.3	0.109	0.156	0.001	0.004	0.027	0.021	0.96%	0.021	0.88%	0.020	69%	71%
			0.9	0.435	0.496	0.002	0.005	0.026	0.018	1.34%	0.018	1.06%	0.016	39%	46%
		20	0.3	0.085	0.048	0.001	0.004	0.026	0.033	1.36%	0.033	1.02%	0.029	72%	77%
			0.9	0.401	0.488	0.001	0.004	0.026	0.024	1.37%	0.024	1.12%	0.022	55%	59%
	4	10	0.3	3.818	2.511	0.003	0.006	0.058	0.026	1.62%	0.026	1.17%	0.021	44%	54%
			0.9	8.678	6.899	0.003	0.006	0.058	0.018	1.18%	0.018	0.84%	0.013	37%	40%
		20	0.3	3.084	1.797	0.003	0.006	0.057	0.023	1.49%	0.023	0.98%	0.019	43%	56%
			0.9	6.793	3.878	0.003	0.006	0.058	0.017	1.24%	0.017	0.99%	0.015	44%	49%
100	2	10	0.3	0.302	0.173	0.001	0.004	0.243	0.018	1.08%	0.018	0.81%	0.017	52%	61%
			0.9	1.860	2.137	0.004	0.007	0.176	0.012	1.15%	0.012	0.99%	0.011	27%	28%
		20	0.3	0.370	0.278	0.005	0.007	0.239	0.022	1.45%	0.022	1.06%	0.019	46%	52%
			0.9	1.334	0.802	0.003	0.006	0.180	0.014	1.40%	0.014	1.03%	0.011	23%	29%
	4	10	0.3	17.212	9.978	0.013	0.006	0.377	0.014	1.31%	0.014	1.08%	0.013	21%	24%
			0.9	58.674	51.408	0.012	0.007	0.391	0.010	1.31%	0.010	1.04%	0.009	6%	10%
		20	0.3	14.930	10.401	0.011	0.007	0.385	0.016	1.69%	0.016	1.18%	0.014	18%	28%
			0.9	58.132	64.654	0.012	0.007	0.397	0.010	1.04%	0.010	0.79%	0.008	15%	21%

GAP = (탐색해-최적해)/최적해 × 100%), mean = arithmetic mean, STD = STandard Deviation

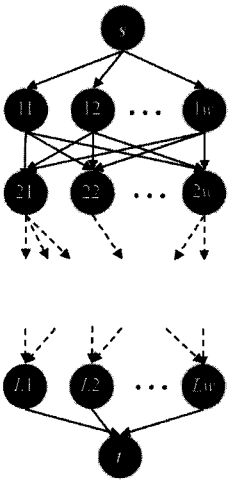
4. 컴퓨터 모의 실험

이 장에서는 무작위로 생성된 인터벌 네트워크 상에서 수행된 일련의 실험 결과를 요약하여 설명한다. 이 실험의 목적은 본 연구에서 제시한 개선된 알고리듬의 성능과 효율을 살펴보기 위하여 Kang[7]이 제시한 알고리듬과 본 연구에서 제안한 알고리듬의 탐색해 간의 편차의 비(GAP = (탐색해-최적해/최적해)를 분석한다. 네트워크 모형에 따른 문제의 난이도 및 대형 최대-최소후회임계경로문제를 풀기 위한 계산수행시간의 증가 경향을 살펴본다.

4.1 네트워크 모형 및 가중치 생성

컴퓨터 모의실험 데이터는 Karaşan[10]에서 사용한 모형과 같은 비순환 계층형 네트워크(*acyclic layered network*) 모형에 기반을 둔다. 계층형이란 네트워크의 마디들이 공통원소를 갖지 않는 노드들의 부분집합으로 분리될 수 있으며, 이 때 각 부분집합의 cardinality는 주어진 폭(width, w)에 해당한다. <그림 3>은 모의실험에서 사용된 L 계층, w폭의 비순환 계층형 네트워크의 형태를 보여준다. 모의실험의 모수는 Karaşan[10] 방법에 기반을 두고 있으며, 모수에 대한 다음의 설명은 Kang[7]의 논문에서 발췌하였다.

각 가지의 interval weight $c_{ij} \in [l_{ij}, u_{ij}]$는 다음과 같이 랜덤하게 생성된다. 균등분포를 따르는 난수 $c_{ij} \in [1, c]$를 생성하고, $l_{ij} \in [(1-d)c_{ij}, (1+d)c_{ij}]$와 $u_{ij} \in [l_{ij}, (1+d)c_{ij}]$를 균등분포에 준하여 랜덤하게 생성한다. 여기서, c와 d는 네트워크의 불확실성의 정도를 조절하는 모수로, c가 클수록 각 가지별 가중치의 중앙값 간의 차이가 크게 되고, d가 클수록 가중치의 간격이 커진다. 모든 테스트는 C++ 언어로 코딩되고 MS Visual C++ ver.6에 의해 컴파일 되었으며, Intel Core2Duo @ 2.33GHz CPU/1GB main memory를 가진 컴퓨터에서 수행되었다. 최적해(MIP solution)은 오픈소스 선형계획법 라이브러리인 Cbc(Coin-or branch and cut, https://projects.coin-or.org/Cbc) ver. 1.1.2에 의해 수행되었는데, 대형 벤치마킹 문제들의 비교결과 상용 소프트웨어인 ILOG CPLEX 12.0에 비해서 5배 정도 느린 것으로 알려져 있으나, 최적해와 휴리스틱 알고리듬에 의한 탐색해 사이의 편차의 비(GAP)를 알기 위해 사용되므로 수행속도는 직접적인 비교 대상이 아니며, 단지 최적해를 구하는데 필요한 시간의 경향을 나타내기 위해 사용되었다.



<그림 3> Illustration of Acyclic Layered Network

4.2 실험 결과

비순환 계층형 네트워크의 크기는 layer L = {50, 100}와, 네트워크의 폭 w = {2, 4}로 각각 2가지 경우로 나누어 4가지 크기의 네트워크를 생성하였다. 모수 c와 d에 대하여 각각 c = {10, 20}, d = {0.3, 0.9}으로 하는 4가지 조건으로 모의 실험하였으며, 각 실험 조건 별로 100회씩 반복하였다.

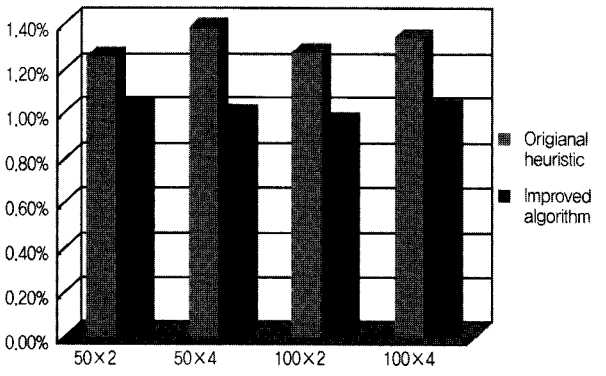
<표 1>은 각 실험 결과를 요약하여 보여준다. <표 1>에서 첫 세열은 네트워크의 형태 및 가중치 조건(L, w, c, d)을 나타내며, MIP 및 기존 휴리스틱(Original heuristic)과 본 연구에서 제안한 개선 알고리듬(Improved algorithm)의 수행도를 나타내는데, 각 열의 의미는 다음과 같다.

- Computation Time : 연산 소요시간(seconds),
- GAP = (탐색해-최적해)/최적해 × 100%),
- Correct % : 정답률, 100번의 반복 중 최적해를 찾은 횟수,
- mean : 산술평균,
- STD(= STandard Deviation) : 표준편차.

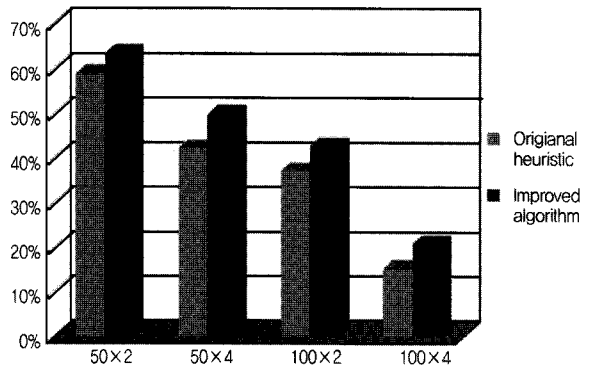
<표 1>에서 네트워크의 크기가 커질수록, 혹은 네트워크의 불확실성이 커질수록(c, d값이 커질수록) 해를 찾는 소요시간이 길어지고, 최적해를 찾는 빈도(Correct %)가 낮아짐을 알 수 있다. <그림 4>는 네트워크의 크기별로 기존 알고리듬(Original heuristic)과 개선된 알고리듬(Improved algorithm)의 평균 GAP을 요약해서 보여준다. 기존의 알고리듬과 개선된 알고리듬을 비교할 때, 계산시간 면에서는 개선된 알고리듬이 기존의 알고리듬에 비해서 많은 시간을 소요하나, 향상율을 아래와 같이 정의할 때 최적해와의 차이(GAP) 면에서는 총평균 24% 향상되었다.

$$\text{향상율}(\%) = \frac{\text{기존 해} - \text{개선 해}}{\text{기존 해}} \times 100$$

<그림 5>는 각 네트워크 크기별로 정리된 최적해를 찾은 비율을 요약하여 나타내는데, 개선된 알고리듬은 평균 44% 정답률(Correct %)을 보여 기존 알고리듬의 평균 정답률(38%) 대비 15.8% 향상되었다. 물론, 네트워크의 크기가 커질수록 문제의 난이도가 높아짐에 따라 정답률은 하락하나, 두 알고리듬 간의 정답률 차이는 유지됨을 볼 수 있다. 결과적으로 개선된 알고리듬은 <그림 5>에서 볼 수 있듯이 더 많은 경우에서 최적해를 찾을 뿐만 아니라, <그림 4>에서 볼 수 있듯이 정답을 찾지 못한 경우에도 해의 향상에 기여한다.



<그림 4> Average GAPs on Network Size



<그림 5> Accuracy to Find the Exact Solution

5. 결론 및 추후연구

본 연구에서는 DAG 상에서 가지의 가중치(혹은 길이)가 범위로 주어진 경우, 후회를 최소화하는 임계경로문제를 풀기위한 기존의 휴리스틱 알고리듬을 개선하는 방법을 제안하였다. 최대후회 최소화 임계경로문제가 전형적인 *NP-hard* 문제이므로 실시간으로 large-scale 문제의 효과적인 근사해를 구하기 위해서 Kang [7]은 Dijkstra's 알고리듬을 기반으로 둔 빠른 휴리스틱 알고리듬을 제안하였는데, 본 연구에서는 해의 성능을 개선하는 알고리듬을 보완하였다. 본 연구결과 개선된 알고리듬의 GAP(= (탐색해-최적해)/최적해)은 기존 알고리듬 대비 약 24% 가량 향상된 결과를 나타냈고, 연산시간은 어느 정도 증가하였으나 실시간 연산에서 수용 가능한 수준을 보였다.

본 연구에서 제안한 알고리듬은 네트워크 모형의 가지의 가중치의 확률 분포를 구하기 어려울 때, 최소한의 가정으로 범위의 값으로 주어진 임계경로문제와 최단경로문제에서 최소후회 기준(Min-max regret criterion)을 적용할 때 실시간 해를 구하는 용도로 사용될 수 있다. 예를 들어, PERT/CPM에서 각 활동의 수행시

간의 추정치를 범위(interval)로 모형화한 경우, 컴퓨터 네트워크의 부하 범위가 알려진 경우, 일반적인 차량 경로 탐색 문제와 같이 최단경로 또는 임계경로 문제로 모형화가 가능한 다양한 의사결정문제에 적용될 수 있을 것으로 기대한다.

현재까지 인터벌 네트워크에서의 최단경로 혹은 임계경로 문제의 현실적인 사례는 많이 개발되지 못했다. 그러므로 이와 같은 형태로 모형화 가능한 현실 문제를 발굴하고 그 결과가 현실에서 어떤 의미를 갖는지에 대한 분석 등이 연구되어야 할 것이다.

참고문헌

[1] Averbakh, I. and Lebedev, V.; "Interval data minmax regret network optimization problems," *Discrete Applied Mathematics*, 138 : 289-301, 2004.

[2] Chanas, S. and Zieliski, P.; "The computational complexity of the criticality problems in a network with interval activity times," *European Journal of Operational Research*, 136(3) : 541-550, 2002.

[3] Conde, E.; "A minmax regret approach to the critical path method with task interval times," *European Journal of Operational Research*, 197(1) : 235-242, 2009.

[4] Eppstein, D.; "Finding the k shortest paths," *SIAM Journal on Computing*, 28(2) : 652-673, 1998.

[5] Inuiguchi, M. and Sakawa, M.; "Minimax Regret Solution To Linear-Programming Problems With An Interval Objective Function," *European Journal of Operational Research*, 86(3) : 526-536, 1995.

[6] Kang, J.-G.; "A Heuristic Algorithm to Find the Minimax Regret Longest Path Problem with Interval Lengths," *Journal of the Korea Management Engineers Society*, 14(3) : 35-46, 2009.

[7] Kang, J.-G.; "Minmax Regret Approach to Disassembly Sequence Planning with Interval Data," *Journal of Society of Korea Industrial and Systems Engineering*, 32(4) : 192-202, 2009.

[8] Kang, J.-G. and Brissaud, D.; "A Product Lifecycle Costing System with Imprecise End-of-Life Data," in the 14th CIRP Conference on Life Cycle Engineering, Waseda University, Japan, 2007.

[9] Kang, J.-G., Lee, D.-H., and Xirouchakis, P.; "Disassembly Sequencing with Imprecise Data : a Case Study," *International Journal of Industrial Engineering-Theory, Applications and Practice*, 10(4) : 407-412, 2003.

[10] Karaşan, O. E., Pinar, M., and Yaman, H.; "The Robust Shortest Path Problem with Interval Data[Online]," Available at : http://www.ie.bilkent.edu.tr/~mustafap/pubs/.

[11] Kasperski, A. and Zielinski, P.; "An approximation algorithm for interval data minmax regret combinatorial optimization problems," *Information Processing Letters*, 97(5) : 177-180, 2006.

[12] Kouvelis, P. and Yu, G.; *Robust Discrete Optimization and Its Application*. Boston. : Kluwer Academic Publishers, 1997.

[13] Montemanni, R. and Gambardella, L. M.; "An Exact Algorithm for the Robust Shortest Path Problem with Interval Data," *Computers and Operations Research*, 31(10) : 1667-1680, 2004.

[14] Montemanni, R. and Gambardella, L. M.; "A branch and bound algorithm for the robust spanning tree problem with interval data," *European Journal of Operational Research*, 161(3) : 771-779, 2005.

[15] Zieliski, P.; "The computational complexity of the relative robust shortest path problem with interval data," *European Journal of Operational Research*, 158(3) : 570-576, 2004.