http://dx.doi.org/10.7236/JIWIT.2012.12.3.51
JIWIT 2012-3-8

방향 그래프의 Prim 최소신장트리 알고리즘

A Prim Minimum Spanning Tree Algorithm for Directed Graph

최명복*, 이상운**

Myeong-Bok Choi, Sang-Un Lee

요 약 본 논문에서는 무방향 그래프의 최소신장트리 (Minimum Spanning Tree, MST) 알고리즘인 Prim MST 알고리즘으로 방향 그래프의 최소신장트리 (DMST)를 구하는 알고리즘을 제안하였다. 먼저, 무방향 그래프와 방향 그래프의 차이점을 반영하여 각 노드에서 유출되는 호들 중 최소 가중치를 가진 호 (Minimum Weight Arc, MWA)를 선택하는 Prim DMST 알고리즘을 제안하였다. 다음으로 Prim DMST 알고리즘과 DMST의 대표적인 Chu-Liu/Edmonds DMST 알고리즘을 실제 3개 그래프에 적용하여 DMST를 찾지 못하는 단점을 보였다. 마지막으로 항상 DMST를 찾을 수 있는 알고리즘으로 Prim DMST를 변형시킨 진보된 Prim DMST 알고리즘을 제안하였다. Prim DMST 알고리즘은 각 노드의 유출 호들 중 MWA를 선택하는 방법이다. 반면에 진보된 Prim DMST 알고리즘은 각 노드의 유출 호들과 유입 호들 중 일치하는 호들을 선택하는 방법을 택하였으며, 만약에 일치하는 호가 없을 경우 각 노드의 유출 호들 중 MWA를 선택하는 방법이다. 제안된 알고리즘을 17개의 다양한 그래프에 적용한 결과, 항상 Chu-Liu/Edmonds DMST 알고리즘과 동일한 DMST를 찾는데 성공하였다. 또한, Chu-Liu/Edmonds DMST 알고리즘과 같이 사이클을 제거하기 위한 복잡한 계산을 하지 않아도 되며, Prim DMST 알고리즘 보다 수행속도를 크게 단축시킬 수 있었다.

Abstract This paper suggests an algorithm that obtains Directed Graph Minimum Spanning Tree (DMST), using Prim MST algorithm which is Minimum Spanning Tree (MST) of undirected graph. At first, I suggested the Prim DMST algorithm that chooses Minimum Weight Arc(MWA) from out-going nodes from each node, considering differences between undirected graph and directed graph. Next, I proved a disadvantage of Prim DMST algorithm and Chu-Liu/Edmonds DMST (typical representative DMST) of not being able to find DMST, applying them to 3 real graphs. Last, as an algorithm that can always find DMST, an advanced Prim DMST is suggested. The Prim DMST algorithm uses a method of choosing MWA among out-going arcs of each node. On the other hand, the advanced Prim DMST algorithm uses a method of choosing a coinciding arc from the out-going and in-going arcs of each node. And if there is no coinciding arc, it chooses MWA from the out-going arcs from each node. Applying the suggested algorithm to 17 different graphs, it succeeded in finding the same DMST as that found by Chu-Liu/Edmonds DMST algorithm. Also, it does not require such a complicated calculation as that of Chu-Liu/Edmonds DMST algorithm to delete the cycle, and it takes less time for process than Prim DMST algorithm.

Key words : Minimum Spanning Tree, MST, Directed Graph Minimum Spanning Tree, DMST, Prim MST Algorithm

*종신회원, 강릉원주대학교 멀티미디어공학과
**정회원, 강릉원주대학교, 멀티미디어공학과
접수일자 2012년 4월 10일, 수정완료 2012년 5월 23일,
게재확정일자 2012년 6월 8일.

Received: 10 April, 2012, Revised: 23 May, 2012,
Accepted: 8 June, 2012
Corresponding Author:cmb5859@gmail.com
Dept. of Multimedia Engineering, Gangnung-Wonju National University Wonju Campus, Korea

Ⅰ. 서 론

그래프 (Graph)는 정점들 (Vertices)과 간선들(Edges)로 구성되어 있으며, 정점들이 간선들로 연결되어 있고 (Connected), 간선들은 방향성 (Directed)과 무방향성 (Undirected)으로 구분된다. 또한 간선들은 가중치가 있는 경우 (Weighted)와 없는 경우 (Unweighted)로 구분된다. 무방향 그래프 (Undirected Graph)는 $G=(V,E)$로 표기하며, 방향 그래프 (Directed Graph)는 $G=(N,A)$는로 표기한다. 노드 (Nodes, N)는 정점 (Vertices, V)이라 하며, 호 (Arcs, A)는 간선 (Edges, E)또는 화살표 (Arrows)라고도 한다. 그래프가 연결되어 있고, 무방향성이며, 가중치를 갖고 있는 경우 신장트리 (Spanning Tree, ST)를 구할 수 있다. ST는 사이클이 발생하지 않는 상태에서 그래프의 모든 정점들을 간선으로 연결한 트리이다. 하나의 그래프에는 ST가 여러 개 존재할 수 있으며, 최소신장트리 (Minimum Spanning Tree, MST)는 그래프가 갖고 있는 다수의 ST들 중 모든 정점들을 연결하는 간선들의 가중치의 합이 최소가 되면서 사이클이 발생하지 않는 ST를 찾는 것으로 전기, 전화, 가스 또는 수도 분야에 활용될 수 있다.[1] 대표적인 MST 알고리즘으로는 Borůvka[2,3], Prim[4]과 Kruskal[5]이 있다. Borůvka MST 알고리즘[2,3]은 모든 간선들의 가중치가 서로 상이한 (Distinct) 그래프에서 MST를 찾는 알고리즘으로 제안되었으며, 현재는 Prim과 Kruskal MST 알고리즘이 널리 사용되고 있다.[1]

MST 알고리즘을 방향 그래프 (Directed Graph)에 적용할 수 있는가? 방향 그래프는 일반적으로 근 (Root) r이 지정되어 있으며, 근 r로부터 방향 경로 (Directed path)를 통해 G의 모든 노드에 도달할 수 있어야 한다. 이 조건을 만족시키기 위해서는 근을 제외한 모든 노드는 유입 차수 (In-degree, 유입되는 호의 수)가 반드시 1개이어야 하며, 유출 차수 (Out-degree, 유출되는 호의 수)는 0 또는 1개 이상이어도 상관없다. 방향그래프의 신장트리 (Directed Spanning Tree, DST)를 찾는 문제는 근 노드 r에서 임의의 모든 모드로 직접 경로를 갖도록 하는 트리를 형성하는 것이며, DMST (Directed Minimum-weight ST)는 방향그래프에 대해 사이클이 없는 신장트리를 형성하면서, 연결된 호들의 가중치 합이 최소가 되도록 하는 것이다.[6-10]

DMST를 찾는 대표적인 알고리즘으로는 Chu-Liu/Edmonds 알고리즘[9,11]을 적용할 수 있다. 그러나 Chu-Liu/Edmonds 알고리즘[9,11]을 적용할 경우 어떤 그래프에서는 DMST를 찾지 못하는 경우도 발생하며, 사이클 발생시 이를 해결하는 과정이 복잡한 단점을 갖고 있다. 또한, 방향성이 없는 간선들과 방향성이 있는 호의 차이로 인해 MST 알고리즘을 DMST를 찾는데 그대로 적용할 수는 없으며, 방향 그래프에 적합하도록 수정하여 적용하여야만 한다. 본 논문에서는 무방향 그래프의 MST를 찾는 Prim MST 알고리즘[4]과 Chu-Liu/Edmonds DMST 알고리즘[9,11]을 적용하여 방향 그래프의 DMST를 구하는 문제점을 고찰해 보고, 간단하면서도 항상 DMST를 찾을 수 있는 새로운 알고리즘을 제안한다.

2장에서는 방향 그래프 MST에 대한 기본 개념을 살펴보고, 실제 그래프에 Prim[4] 과 Chu-Liu/Edmonds 알고리즘[9,11]을 적용한 문제점을 고찰해 본다. 3장에서는 DMST를 보다 쉽게 찾을 수 있는 알고리즘을 제안한다. 4장에서는 15개의 다양한 실제 그래프들에 대해 제안된 DMST 알고리즘을 Prim[4]과 Chu-Liu/Edmonds 알고리즘[9,11]과 비교하여 알고리즘의 적용성을 평가해 본다.

Ⅱ. 관련 연구와 연구 배경

1. 개념 정의

무방향 그래프 $G=(V,E)$의 간선은 $e=\{x,y\}$로, 방향 그래프 $G=(N,A)$의 호는 방향성이므로 순서쌍 $a=(x,y)$로 표기한다. 여기서, x는 꼬리 (Tail), y는 머리 (Head)라 하며, 노드 n은 유출 차수와 유입 차수를 갖는다. 방향 그래프 $G=(N,A)$를 트리로 볼 때, 하나의 노드를 근 (Root, r) 으로 지정한 경우를 “근이 있는 트리 (Rooted Tree)”라 하며, 근을 지정하지 않은 경우를 “자유 트리 (Free Tree)”라 한다.[12]

방향 그래프는 유입 분기 (In-branching)와 유출 분기 (Out-branching)로 구분할 수 있다. 유입 분기는 근 r에 근원을 두고 분기하는 트리로 임의의 노드에서 근 r로 방향 경로를 갖고 있는 경우이며, 유출 분기는 근 r은 방향 경로를 통해 G의 모든 노드에 도달할 수 있다.[13] 방향그래프의 신장트리 (DST)를 찾는 문제는 근 노드 r에서 모든 노드로 유일한 방향 경로를 갖도록 하는 트리를 형성하는 것이며, DMST는 호들의 가중치 합이 최소가

되는 DST이다. DMST는 도로와 전화망 분야에 활용될 수 있다.[14]

유입 분기 DST는 방향그래프 $G(N,A)$와 근 노드 $r \in N$이 주어졌을 때, G의 모든 노드에서 근 노드 r로 도달 가능하다. r을 제외한 각 노드는 2개의 호가 꼬리를 공유하지 않고 정확히 하나의 유출 호 (꼬리)를 가져야 하며, r은 유출 호를 갖지 않는다.[13,14] 즉, 근 r을 제외한 모든 노드는 $In-degree \geq 1$과 $Out-degree = 1$을, 근 r은 $In-degree \geq 1$, $Out-degree = 0$ 조건을 만족시켜야 한다.

반면에, 유출 분기 DMST는 $G(N,A)$와 근 노드 $r \in N$이 주어졌을 때, 근 노드 r로부터 모든 다른 노드로 유일한 방향 경로를 갖고 있으며, 모든 호의 비용 (가중치)의 합 $(\Sigma w(a))$이 최소인 ST로, 근을 제외한 각 노드는 단지 하나의 유입 호만을 가져야 하며 (사이클 미 발생), $n-1$개의 호가 모든 노드를 연결한 그래프이다.[6-10] 즉, 근 r은 $In-degree = 0$과 $Out-degree = 1,2,3,\cdots$을, 근 r 이외의 모든 노드는 $In-degree = 1$과 $Out-degree = 0,1,2,\cdots$ 조건을 만족시켜야 한다. 본 논문에서는 유출 분기 방향 그래프의 최소 신장트리를 찾는 문제에 초점을 맞추며, 이를 "DMST"라 한다.

2. 방향 그래프의 최소신장트리 알고리즘

최소신장트리 (MST)는 일반적으로 무방향 그래프의 모든 정점을 연결하는 간선들의 가중치 합이 최소가 되는 신장트리를 찾는 문제로 알려져 있다. MST를 찾는 대표적인 알고리즘으로는 Borůvka[2,3], Prim[4]과 Kruskal[5]이 있다. Borůvka[2,3]와 Kruskal[5] MST 알고리즘은 그래프의 간선들을 대상으로 최소 가중치를 갖는 간선들을 선택하여 신장시키는 방법이며, Prim MST 알고리즘[4]은 정점들을 최소 가중치를 갖는 간선들로 신장시키는 방법이다. 무방향 그래프에 적용된 MST 알고리즘을 방향 그래프에 적용시키려면 Borůvka[2,3]와 Kruskal[5] MST 알고리즘으로 DMST를 찾는 과정에서 "방향그래프의 한 노드의 $In-degree = 1$이 되어야 하며, 근 r로부터 다른 모든 노드로 방향경로가 설정되어야 한다."는 조건을 만족시키지 못하는 경우가 빈번히 발생할 수 있다. 따라서 Borůvka[2,3]와 Kruskal[5] MST 알고리즘으로는 DMST에 적용할 수 없다. 반면에 Prim MST 알고리즘[4]은 근 노드로부터 방향 경로를 갖도록 노드들을 연결하는 방법으로 방향그래프의 신장트리를 얻는 조건은 만족시킬 수 있다. Prim MST 알고리즘[4]은 "임의의 정점을 선택하고, 이에 연결된 간선들 중에서 MWE를 선택한다. 이 간선에 연결된 정점의 간선들 가중치와 기존에 선택된 정점에서 선택되지 않은 간선들 가중치 중에서 MWE를 선택하는 방법이다. (단, 이 과정에서 사이클이 발생하는 간선은 무시한다.) 이 방법을 모든 정점들이 선택될 때까지 수행한다." Prim MST 알고리즘은 최대 $v-1$번을 수행한다.

방향 그래프의 대표적인 최소신장트리 알고리즘은 Chu-Liu/Edmonds DMST 알고리즘[9,11]이 있으며, 그림 1과 같이 수행된다. 이 알고리즘은 그래프의 $In-degree = 0$인 노드가 존재할 경우 모든 노드의 최소 가중치를 갖는 유입 호 선택시 사이클이 발생하지 않아 대부분은 ③에서 알고리즘이 종료된다. 그러나 사이클이 발생하였을 경우 사이클을 제거하기 위해 최소 가중치를 갖는 유입 호를 선택하는 ④의 계산 방법이 어려운 단점이 있다.

≪Chu-Liu/Edmonds DMST Algorithm ≫

① 만약 근 노드로 유입되는 호가 있다면 삭제한다.
② 근을 제외한 모든 노드에 대해 최소 가중치를 가진 유입 호를 선택하고, 선택된 $n-1$개의 호로 S 집합을 생성한다.
③ S가 사이클이 형성되지 않으면 $G(N,S)$는 MST가 되며, 알고리즘을 종료한다. 그렇지 않으면 ④를 수행한다.
④ 각 사이클에 대해, 사이클에 있는 노드들을 Pseudo-node k로 축소시키고, 사이클 외부의 어떤 노드 i에서 사이클에 있는 노드 j로 유입 호의 가중치를 $c(i,k) = c(i,j) - [c(x_j,j) - min_j[c(x_j,j)]$로 수정한다. 여기서 $c(x_j,j)$는 사이클에 있는 임의의 노드에서 j 노드로 유입되는 호의 가중치이다.
⑤ 각 Pseudo-node에 대해, 수정된 최소 가중치를 가진 유입 호를 선택한다.
⑥ 선택된 사이클 유입 호를 사이클에 존재하는 동일한 실제 노드로 유입되는 호와 교체한다.
⑦ 만약 사이클이 추가로 존재하면 ④로 복귀한다.

그림 1. Chu-Liu/Edmonds DMST Algorithm
Fig. 1. Chu-Liu/Edmonds DMST 알고리즘

Prim MST 알고리즘[4]을 방향 그래프에 적용하기 위해 수정된 형태의 알고리즘은 다음과 같이 수행되며, 그림 2에 제시하였다.

① 근 노드가 지정되지 않은 그래프의 경우 임의의 노드를 근 노드로 설정하며, 근 노드가 지정된 그래프의 경우는 근 노드를 DMST Nodes (n_{mst})에 추가한다.
② n_{mst}에 추가된 노드의 유출 호들을 Candidate Arcs (a_c)에 추가한다.(a_a)
③ a_c에 있는 호들 $(a_r + a_a)$의 y 노드가 n_{mst}에 존재하면 사이클이 발생하는 경우이므로 이들 호를 삭제한다.(a_d)
④ a_c에서 최소 가중치를 가진 호 (MWA)를 선택하여 DMST Arcs (a_{mst})에 추가하고, a_c에서 삭제한다.

(만약 동일한 가중치를 갖는 호가 다수 존재시 모두 선택한다.)

⑤ a_{mst}에 새로 추가된 호의 y 노드를 N에서 삭제하고, DMST Nodes (n_{mst})에 추가한다.

⑥ 만약 $N=\{\varnothing\}$이면 알고리즘을 종료하고, 그렇지 않으면 ②로 복귀한다.

≪Prim's DMST Algorithm ≫

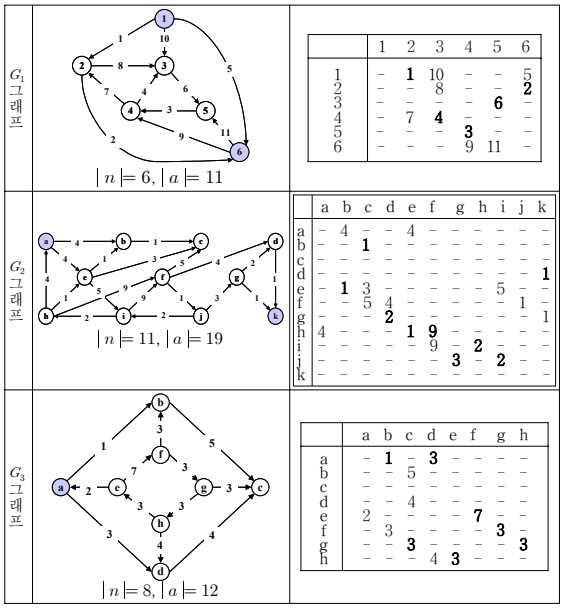
MWA : Minimum-weight Arc
Candidate Arcs : ST를 구성하기 위해 연결에 이용될 후보 Arc들 ($a_c = a_r + a_a - a_d$)
a_a : 새로 추가되는 Node의 Out-degree Arcs, a_r : 삭제되지 않고 남아 있는 Arcs
a_d : DMST Nodes로 이미 선택되었거나 사이클 발생으로 삭제되는 Arcs
MWA : Minimum-Weight Arc (x, y), DMST = DMST Nodes (n_{mst}) + DMST Arcs (a_{mst})

| n |×| n | **Adjacency 행렬 작성**
N에서 근 노드 선택, DMST Nodes (n_{mst})에 저장, N에서 근 노드 삭제
FOR | n | *to* 0 do
  n_{mst}에 추가된 노드의 Out-degree Arcs들을 Candidate Arcs에 추가(a_a)
  IF Candidate Arcs ($a_r + a_a$) 중 $y \in n_{mst}$ THEN Candidate Arcs에서 삭제(a_d) **/* 사이클 발생**
  ENDIF
  Candidate Arcs에 남아 있는 Arcs들($a_r + a_a - a_d$) **중 MWA 선택**
  IF MWA ≥ 2 and Head 노드 동일 THEN 1개만 선택하여 a_{mst}에 추가하고 a_c 에서 삭제, Head 노드 동일 호는 삭제 **【S_1】**
  ELSE IF MWA ≥ 2 and Head 노드 상이 THEN 모두 선택, a_{mst}에 추가, a_c 에서 삭제 **【S_2】**
  ELSE IF MWA =1 THEN 선택하여 a_{mst}에 추가하고 a_c 에서 삭제 **【S_3】**
  ENDIF
  a_{mst}에 추가된 호 (x, y)에서 y를 n_{mst}에 저장, N에서 삭제
END

그림 2. Prim DMST 알고리즘
Fig. 2. Prim DMST Algorithm

3. 알고리즘 적용 문제점과 연구 배경

그림 3의 3개 그래프에 대해 Prim DMST 알고리즘과 Chu-Liu/Edmonds DMST 알고리즘을 적용하여 보자.



	1	2	3	4	5	6
1	–	**1**	10	–	–	5
2	–	–	8	–	–	**2**
3	–	–	–	–	**6**	–
4	–	7	**4**	–	–	–
5	–	–	–	**3**	–	–
6	–	–	–	9	11	–

	a	b	c	d	e	f	g	h	i	j	k
a	–	4	–	–	4	–	–	–	–	–	–
b	–	–	**1**	–	–	–	–	–	–	–	–
c	–	–	–	–	–	–	–	–	–	–	–
d	–	–	–	–	–	–	–	–	–	–	**1**
e	–	**1**	3	–	–	–	–	–	5	–	–
f	–	–	5	4	–	–	–	–	–	1	–
g	–	–	–	**2**	–	–	–	–	–	–	1
h	4	–	–	–	**1**	**9**	–	–	–	–	–
i	–	–	–	–	–	9	–	**2**	–	–	–
j	–	–	–	–	–	–	**3**	–	**2**	–	–
k	–	–	–	–	–	–	–	–	–	–	–

	a	b	c	d	e	f	g	h
a	–	**1**	–	**3**	–	–	–	–
b	–	–	5	–	–	–	–	–
c	–	–	–	–	–	–	–	–
d	–	–	4	–	–	–	–	–
e	2	–	–	–	–	**7**	–	–
f	–	3	–	–	–	–	**3**	–
g	–	–	**3**	–	–	–	–	**3**
h	–	–	–	4	**3**	–	–	–

그림 3. 알고리즘에 사용된 그래프
Fig. 3. Graphs for Algorigthm

G_1 그래프는 Yang[9]에서 인용되었으며, $In-degree=0$인 근 노드가 ①임을 알 수 있다. G_2 그래프는 Llewelly[15]에서, G_3 그래프는 Ikeda[16]에서 인용되었다. G_2와 G_3 그래프는 근 노드가 없어 DMST를 구하기 위해서는 임의로 근 노드를 설정하여야 한다. 본 논문에서는 편의상 노드 ⓐ를 근 노드로 설정하고 알고리즘을 적용하였다.

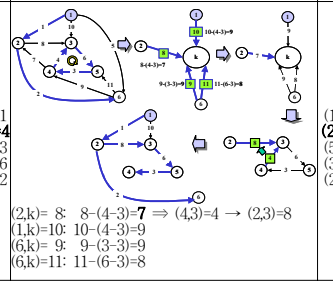
Prim DMST 알고리즘과 Chu-Liu/Edmonds DMST 알고리즘을 적용한 결과는 G_1 그래프는 표 1에, G_2 그래프는 표 2에, G_3 그래프는 표 3에 각각 제시되어 있다.

표 1. G_1 그래프의 DMST 알고리즘 적용
Table 1. Application of DMST Algorithm to G_1 Graph

(a) Prim DMST 알고리즘

N	MST Nodes	Candidate Arcs				선택 방법	MST Arcs
		a_a	a_r	a_d	$a_r + a_a - a_d$		
{1,2,3,4,5,6}	{∅}	{∅}	{∅}	{∅}	{∅}	–	{∅}
{2,3,4,5,6}	{1}	(1,2)=1,(1,3)=10 (1,6)=5	{∅}	{∅}	**(1,2)=1**,(1,3)=10 (1,6)=5	S_3	**(1,2)=1**
{3,4,5,6}	{1,2}	(2,3)=8,(2,6)=2	(1,3)=10,(1,6)=5	{∅}	(1,3)=10,(1,6)=5 (2,3)=8,**(2,6)=2**	S_3	**(2,6)=2**
{3,4,5}	{1,2,6}	(6,4)=9,(6,5)=11	(1,3)=10,**(1,6)=5** (2,3)=8	(1,6)=5	(1,3)=10,**(2,3)=8** (6,4)=9,(6,5)=11	S_3	**(2,3)=8**
{4,5}	{1,2,6,3}	(3,5)=6	**(1,3)=10**,(6,4)=9 (6,5)=11	(1,3)=10	(6,4)=9,(6,5)=11 **(3,5)=6**	S_3	**(3,5)=6**
{4}	{1,2,6,3,5}	(5,4)=3	(6,4)=9,**(6,5)=11**	(6,5)=11	(6,4)=9,**(5,4)=3**	S_3	**(5,4)=3**
{∅}	{1,2,6,3,5,4}	알고리즘 종료				–	–

(b) Chu-Liu/Edmonds DMST 알고리즘

Node	노드 유입 MWA		MSF Arcs	MSF 유입 MWA	DMST Arcs
	ALL	선택			
1	–	–	–		–
2	(1,2)=1	(1,2)=1	(1,2)=1		(1,2)=1
3	(4,3)=4	(4,3)=4	**(4,3)=4**		**(2,3)=8**
4	(5,4)=3	(5,4)=3	(5,4)=3		(5,4)=3
5	(3,6)=6	(3,6)=6	(3,5)=6		(3,5)=6
6	(2,6)=2	(2,6)=2	(2,6)=2		(2,6)=2

(2,k)= 8: 8-(4-3)=**7** ⇒ (4,3)=4 → (2,3)=8
(1,k)=10: 10-(4-3)=9
(6,k)= 9: 9-(3-3)=9
(6,k)=11: 11-(6-3)=8

G_1 그래프에서 Prim DMST 알고리즘은 근 노드 ①로부터 모든 노드로 유일한 방향 경로가 설정되었고, 근 노드를 제외한 모든 노드의 $In-degree=1$이며, $\Sigma w(a)=20$인 MST를 얻었다. Chu-Liu/Edmonds DMST 알고리즘은 ③-④-⑤에서 사이클이 발생하여 이 사이클로 유입되는 최소 가중치 호인 $(2,3)=8$이 선택되고 ③ 노드로 유

입되는 $(4,3)=4$가 삭제되었다. 따라서 근 노드 ①로부터 모든 노드로 유일한 방향 경로가 설정되고, 근 노드를 제외한 모든 노드의 $In-degree=1$이며, $\Sigma w(a)=20$인 MST를 얻을 수 있었다.

G_2 그래프에 대해, Prim DMST 알고리즘은 근 노드 ⓐ로부터 모든 노드로 방향경로가 설정되었으며, $\Sigma w(a)=32$를 얻었다. Chu-Liu/ Edmonds DMST 알고리즘은 최소 가중치를 가진 유입 호가 다수 발생하였을 경우 어떤 호를 선택하는지에 대해 명확히 제시하지 않고 있다. 그러나 DMST는 근을 제외한 모든 노드의 유입 호가 1개만이 존재해야 하므로 본 논문에서는 1개만을 선택하였다. Chu-Liu/Edmonds DMST 알고리즘은 근 노드를 ⓐ라 할 때, ⓗ-ⓕ-ⓘ-ⓙ에 사이클이 발생하였다. 사이클로 유입되는 호는 $(e,i)=5$만 존재하며, 이를 연결하는 경우 ⓘ 노드로 유입되는 $(j,i)=2$ 호가 삭제된다. 이 경우 다시 ⓗ-ⓔ-ⓘ가 사이클이 발생하였으며, 이 사이클로 유입되는 호는 $(a,e)=4$만 존재하여 $(h,e)=1$이 $(a,e)=4$로 대체되었다. 결국, $\Sigma w(a)=29$인 MST를 얻었다. 결국 Chu-Liu/Edmonds DMST 알고리즘이 Prim DMST 알고리즘보다 호들의 가중치 합이 작은 DMST를 얻을 수 있었다.

표 2. G_2 그래프의 DMST 알고리즘 적용
Table 2. Application of DMST Algorithm to G_2 Graph

(a) Prim DMST 알고리즘

N	MST Nodes	Candidate Arcs $(a_r+a_u-a_d)$	선택방법	MST Arcs
{a,b,c,d,e,f,g,h,i,j,k}	{ ∅ }	{ ∅ }	-	{ ∅ }
{b,c,d,e,f,g,h,i,j,k}	{a}	**(a,b)=4**,(a,e)=4	S_2	**(a,b)=4**,(a,e)=4
{c,d,f,g,h,i,j, k}	{a,b,e}	**(b,c)=1**,(e,c)=3,(e,i)=5	S_3	**(b,c)=1**
{d,f,g,h,i,j,k}	{a,b,e,c}	**(e,i)=5**	S_3	**(e,i)=5**
{d,f,g,h,j,k}	{a,b,e,c,i}	(i,f)=9,**(i,h)=2**	S_3	**(i,h)=2**
{d,f,g,j,k}	{a,b,e,c,i,h}	**(i,f)=9**,(h,f)=9	S_1	**(i,f)=9**
{d,g,j,k}	{a,b,e,c,i,h, f}	(f,d)=4,**(f,j)=1**	S_3	**(f,j)=1**
{d,g,k}	{a,b,e,c,i,h,f,j}	(f,d)=4,**(j,g)=3**	S_3	**(j,g)=3**
{d,k}	{a,b,e,c,i,h,f,j,g}	(f,d)=4,(g,d)=2,**(g,k)=1**	S_3	**(g,k)=1**
{d}	{a,b,e,c,i,h,f,j,g,k}	(f,d)=4,**(g,d)=2**	S_3	**(g,d)=2**
{ ∅ }	{a,b,e,c,i,h,f,j,g,k,d}	{ ∅ }	-	-

(b) Chu-Liu/Edmonds DMST 알고리즘

Node	노드 유입 MWA		MSF Arcs	MSF 유입 MWA	DMST Arcs
	ALL	선택			
a	(h,a)=4	-	-		-
b	(e,b)=1	(e,b)=1	(e,b)=1		(e,b)=1
c	(b,c)=1	(b,c)=1	(b,c)=1		(b,c)=1
d	(g,d)=2	(g,d)=2	(g,d)=2		(g,d)=2
e	(h,e)=1	(h,e)=1	**(h,e)=1**		**(a,e)=4**
f	(h,f)=9,(i,f)=9	(h,f)=9	(h,f)=9		(h,f)=9
g	(j,g)=3	(j,g)=3	(j,g)=3		(j,g)=3
h	(i,h)=2	(i,h)=2	(i,h)=2		(i,h)=2
i	(j,i)=2	(j,i)=2	**(j,i)=2**		**(e,i)=5**
j	(f,j)=1	(f,j)=1	(f,j)=1		(f,j)=1
k	(d,k)=1,(g,k)=1	(d,k)=1	(d,k)=1		(d,k)=1

(e,k)=5: 5-(2-1)=4 ⇒ (j,i)=2 → (e,i)=5
(a,k)=4: 4-(1-1)=4 ⇒ (h,e)=1 → (a,e)=4

표 3. G_3 그래프의 DMST 알고리즘 적용
Table 3. Application of DMST Algorithm to G_3 Graph

(a) Prim DMST 알고리즘

N	MST Nodes	Candidate Arcs $(a_r+a_u-a_d)$	선택방법	MST Arcs
{a,b,c,d,e,f,g,h}	{ ∅ }	{ ∅ }	-	{ ∅ }
{b,c,d,e,f,g,h}	{a}	**(a,b)=1**,(a,d)=3	S_3	**(a,b)=1**
{c,d,e,f,g,h}	{a,b}	**(a,d)=3**,(b,c)=5	S_3	**(a,d)=3**
{c,e,f,g,h}	{a,b,d}	(b,c)=5,**(d,c)=4**	S_3	**(d,c)=4**
{e,f,g,h}	{a,b,d,c}	{ ∅ }		-
		알고리즘 수행 불가		

(b) Chu-Liu/Edmonds DMST 알고리즘

Node	노드 유입 MWA		MSF Arcs	MSF 유입 MWA	DMST Arcs
	ALL	선택			
a	(e,a)=2	-	-		-
b	(a,b)=1	(a,b)=1	(a,b)=1		(a,b)=1
c	(g,c)=3	(g,c)=3	(g,c)=3		(g,c)=3
d	(a,d)=3	(a,d)=3	(a,d)=3		(a,d)=3
e	(h,e)=3	(h,e)=3	(h,e)=3		(h,e)=3
f	(e,f)=7	(e,f)=7	(e,f)=7		(e,f)=7
g	(f,g)=3	(f,g)=3	(f,g)=3		(f,g)=3
h	(g,h)=3	(g,h)=3	(g,h)=3		(g,h)=3

ⓔ-ⓕ-ⓖ-ⓗ 사이클로 유입되는 호가 없어 MST를 구하지 못함

실패

G_3 그래프에 대해, Prim DMST 알고리즘은 ⓒ 노드에서 더 이상의 유출 호가 존재하지 않아 알고리즘이 종료되었으며, ST를 얻는데 실패하였다. 또한, Chu-Liu/ Edmonds DMST 알고리즘도 ⓔ-ⓕ-ⓖ-ⓗ에서 사이클이 발생하였으나 이 사이클로 유입되는 호가 없어 알고리즘이 수행되지 못하였으며, ST를 얻는데 실패하였다.

즉, G_1 그래프는 동일한 MST를 얻은 반면, G_2 그래프는 가중치 합이 다른 결과를 얻었으며, G_3 그래프는 MST를 얻는데 실패하였다. 이와 같이 다양한 결과를 나타내는 알고리즘을 일반적으로 신뢰할 수 있는가? 또한, G_2 그래프의 Chu-Liu/Edmonds DMST 알고리즘으로 얻은 $\Sigma w(a)$가 최소인 최적의 MST라고 할 수 있는가? Chu-Liu/Edmonds DMST 알고리즘의 사이클 제거 과정을 단순화시킬 수는 없는가? Prim DMST 알고리즘의 수행 횟수를 단축시키면서 항상 MST를 얻을 수 는 있는가? 이러한 의문점을 해결하기 위해 3장에서는 Prim DMST 알고리즘을 수정하여 “진보된 Prim DMST 알고리즘 (Advanced Prim DMST Algorithm)”을 제안한다.

Ⅲ. 진보된 Prim DMST 알고리즘

1. 알고리즘 제안

방향그래프의 MST를 간단히 얻기 위해 Prim DMST

알고리즘을 수정한 다음의 알고리즘을 제안한다. 이 알고리즘은 그림 4에 제시하였다.

① 근 노드를 포함한 모든 노드에 대해 최소 가중치를 갖는 유입 호 (a_{mwa})를 선택한다. (만약 하나의 노드에서 동일한 가중치를 가진 유입 호가 다수 존재시 모두 선택한다.)

② 근 노드가 지정되지 않은 그래프의 경우 임의의 노드를 근 노드로 설정하며, 근 노드가 지정된 그래프의 경우는 근 노드를 DMST Nodes (n_{mst})에 추가한다.

③ n_{mst}에 추가된 노드의 유출 호들을 Candidate Arcs (a_c)에 추가한다.(a_a)

④ a_c에 있는 호들 ($a_r + a_a$)의 y 노드가 n_{mst}에 존재하면 사이클이 발생하는 경우이므로 이들 호를 삭제한다.(a_d)

⑤ a_c의 모든 호들에 대해, a_{mwa}와 비교한다. 만약 동일한 호가 존재하면 DMST Arcs (a_{mst})에 추가하고, a_c에서 삭제한다. a_c에서 a_{mst}에 추가된 호와 Head가 동일한 호를 삭제 한다. (왜냐하면, 방향그래프의 한 노드의 유입 호의 개수는 1이 되어야만 하기 때문이다.)

만약, a_c와 a_{mst}가 동일한 호가 존재하지 않으면 a_c에서 MWA를 선택하여 a_{mst}에 추가하고, a_c에서 삭제한다. 이 결과 $a_c = a_r$만 존재한다.

⑥ a_{mst}에 새로 추가된 호의 y 노드를 N에서 삭제하고, DMST Nodes (n_{mst})에 추가한다.

⑦ 만약 $N = \{\varnothing\}$이면 알고리즘을 종료하고, 그렇지 않으면 ③ 으로 복귀한다.

≪Advanced Prim DMST Algorithm ≫

MWA : Minimum-weight Arc
Candidate Arcs : ST를 구성하기 위해 연결에 이용될 후보 Arc들 ($a_c = a_r + a_a - a_d$)
a_a : 새로 추가되는 Node의 Out-degree Arcs, a_r : 삭제되지 않고 남아 있는 Arcs
a_d : MSF Nodes로 이미 선택되었거나 사이클 발생으로 삭제되는 Arcs
MWA : Minimum-Weight Arc (x, y), DMST = DMST Nodes (n_{mst}) + DMST Arcs (a_{mst})

$|n|\times|n|$ 인접행렬 작성
근 노드를 포함한 모든 노드의 유입 호들 중 최소 가중치를 갖는 호 선택, a_{mwa}에 추가
N에서 근 노드 선택, DMST Nodes (n_{mst})에 저장, N에서 근 노드 삭제
FOR $|n|$ *to* 0 do
  n_{mst}에 추가된 노드의 Out-degree Arcs들을 Candidate Arcs에 추가(a_a)
  IF Candidate Arcs ($a_r + a_a$) 중 $y \in n_{mst}$ THEN Candidate Arcs에서 삭제(a_d) **/* 사이클 발생**
  ENDIF
  Candidate Arcs ($a_c = a_r + a_a - a_d$) **와 a_{mwa} 비교**
  IF $a_c = a_{mwa}$호 존재 THEN
    모두 선택하여 a_{mst}에 추가하고 a_c에서 삭제 **【S_1】**
    a_{mst}에 추가된 호와 동일한 Head를 가진 호를 a_c에서 삭제 **【S_2】**
  ELSE IF $a_c \neq a_{mwa}$ THEN
    IF $w(x,y) = w(x,z)$ and $w(y,z) < w(x,y)$ THEN $w(x,z)$ 선택 a_{mst}에 추가, a_c에서 삭제 **【S_3】**
    ELSE IF 동일 가중치 호가 없을 경우 THEN MWA 선택 a_{mst}에 추가, a_c에서 삭제 **【S_4】**
    ENDIF
  ENDIF
  a_{mst}에 추가된 호 (x, y)에서 y를 n_{mst}에 저장, N에서 삭제
END

그림 4. 진보된 Prim DMST 알고리즘
Fig. 4. Advanced Prim DMST Algorithm

2. 알고리즘 적용 방법

그림 3의 3개 그래프에 진보된 Prim DMST 알고리즘을 적용한 과정은 표 4에, 적용 결과 얻은 MST는 그림 5에 제시되어 있다. 알고리즘 적용 결과, 진보된 Prim DMST 알고리즘은 3개 그래프 모두에서 Chu-Liu/Edmonds DMST 알고리즘[9,11]과 동일한 $\Sigma w(a)$의 MST를 얻는데 성공하였다. 또한, Chu-Liu/Edmonds DMST 알고리즘[9,11]과 같이 "사이클을 제거하기 위해 Pseudo-node를 적용하고 유입 호의 가중치를 수정하여 최소 가중치를 갖는 유입 호를 선택하는 과정"이 없어 알고리즘이 단순함을 알 수 있다.

표 4. 진보된 Prim DMST 알고리즘 적용
Table 4. Advanced Prim DMST Algorithm Application

(a) G_1 그래프 (근 노드 = 1)

N	n_{mst}	Candidate Arcs ($a_r + a_a - a_d$)	선택방법	a_{mwa}	a_{mst}
{1,2,3,4,5,6}	{ ∅ }	{ ∅ }	-	(1,2)=1 (4,3)=4 (5,4)=3 (3,5)=6 (2,6)=2	{ ∅ }
{2,3,4,5,6}	{1}	**(1,2)=1**,(1,3)=10,(1,6)=5	S_1		**(1,2)=1**
{3,4,5,6}	{1,2}	(1,3)=10,(**1,6**)=5,(2,3)=8,**(2,6)=2**	S_1,S_2		**(2,6)=2**
{3,4,5}	{1,2,6}	(1,3)=10,**(2,3)=8**,(6,4)=9,(6,5)=11	S_4		**(2,3)=8**
{4,5}	[1,2,6,3}	(6,4)=9,(6,**5**)=11,**(3,5)=6**	S_1,S_2		**(3,5)=6**
{4}	{1,2,6,3,5}	(6,**4**)=9,**(5,4)=3**	S_1,S_2		**(5,4)=3**
{ ∅ }	{1,2,6,3,5,4}	알고리즘 종료			-

(b) G_2 그래프 (근 노드 = a)

N	n_{mst}	Candidate Arcs ($a_r + a_a - a_d$)	선택 방법	a_{mwa}	a_{mst}
{a,b,c,d,e,f,g,h,i,j,k}	{ ∅ }	{ ∅ }	-	(h,a)=4 (e,b)=1 (b,c)=1 (g,d)=2 (h,e)=1 (h,f)=9,(i,f)=9 (j,g)=3 (j,h)=2 (j,j)=2 (f,j)=1 (d,k)=1,(g,k)=1	-
{b,c,d,e,f,g,h,i,j,k}	{a}	(a,**b**)=4,**(a,e)=4**	S_3		**(a,e)=4**
{b,c,d,f,g,h,i,j,k}	{a,e}	(a,**b**)=4,**(e,b)=1**,(e,c)=3 (e,i)=5	S_1,S_2		**(e,b)=1**
{c,d,f,g,h,i,j,k}	{a,e,b}	(e,**c**)=3,(e,i)=5,**(b,c)=1**	S_1,S_2		**(b,c)=1**
{d,f,g,h,i,j, k}	{a,e,b,c}	**(e,i)=5**	S_4		**(e,i)=5**
{d,f,g,h,j,k}	{a,e,b,c,i}	**(j,f)=9,(i,h)=2**	S_1		**(i,f)=9,(i,h)=2**
{d,g,j,k}	{a,e,b,c,i,f, h}	(f,d)=4,**(f,j)=1**	S_1		**(f,j)=1**
{d,g,k}	{a,e,b,c,i,f,h,j}	(f,d)=4,**(j,g)=3**	S_1		**(j,g)=3**
{d,k}	{a,e,b,c,i,f,h,j,g}	(f,**d**)=4,**(g,d)=2,(g,k)=1**	S_1,S_2		**(g,d)=2,(g,k)=1**
{ ∅ }	{a,e,b,c,i,f,h,j,g,d,k}	알고리즘 종료			-

(b) G_2 그래프 (근 노드 = f)

N	n_{mst}	Candidate Arcs ($a_r + a_a - a_d$)	선택 방법	a_{mwa}	a_{mst}
{a,b,c,d,e,f,g,h,i,j,k}	{ ∅ }	{ ∅ }	-	(h,a)=4 (e,b)=1 (b,c)=1 (g,d)=2 (h,e)=1 (h,f)=9,(i,f)=9 (j,g)=3 (j,h)=2 (j,j)=2 (f,j)=1 (d,k)=1,(g,k)=1	-
{a,b,c,d,e,g,h,i,j,k}	{f}	(f,c)=5,(f,d)=4,**(f,j)=1**	S_1		**(f,j)=1**
{a,b,c,d,e,g,h,i,k}	{f,j}	(f,c)=5,(f,d)=4,**(j,g)=3** **(j,i)=2**	S_1		**(j,g)=3,(j,i)=2**
{a,b,c,d,e,h,k}	{f,j,g,i}	(f,c)=5,(f,**d**)=4,**(g,d)=2** **(g,k)=1,(i,h)=2**	S_1,S_2		**(g,d)=2,(g,k)=1** **(i,h)=2**
{a,b,c,e}	{f,j,g,i,d,k ,h}	(f,c)=5,**(h,a)=4,(h,e)=1**	S_1		**(h,a)=4,(h,e)=1**
{b,c}	{f,j,g,i,d,k,h,a,e}	(f,c)=5,(a,**b**)=4,**(e,b)=1** (e,c)=3	S_1,S_2		**(e,b)=1**
{c}	{f,j,g,i,d,k,h,a,e,b}	(f,**c**)=5,(e,**c**)=3,**(b,c)=1**	S_1,S_2		**(b,c)=1**
{ ∅ }	{f,j,g,i,d,k,h,a,e,b,c}	알고리즘 종료			-

(d) G_3 그래프 (근 노드 = a)

N	n_{mst}	Candidate Arcs $(a_r + a_a - a_d)$	선택방법	a_{mwa}	a_{mst}
{a,b,c,d,e,f,g,h}	{ ∅ }	{ ∅ }	-	(e,a)=2 (a,b)=1 (g,c)=3 (a,d)=3 (h,e)=3 (e,f)=7 (f,g)=3 (g,h)=3	{ ∅ }
{b,c,d,e,f,g,h}	{a}	**(a,b)=1,(a,d)=3**	S_1		**(a,b)=1,(a,d)=3**
{c,e,f,g,h}	{a,b,d}	(b,c)=5,**(d,c)=4**	S_1		**(d,c)=4**
{e,f,g,h}	{a,b,d,c}	{ ∅ }	-		-
알고리즘 종료					

(e) G_3 그래프 (근 노드 = f)

N	n_{mst}	Candidate Arcs $(a_r + a_a - a_d)$	선택방법	a_{mwa}	a_{mst}
{a,b,c,d,e,f,g,h}	{ ∅ }	{ ∅ }	-	(e,a)=2 (a,b)=1 (g,c)=3 (a,d)=3 (h,e)=3 (e,f)=7 (f,g)=3 (g,h)=3	-
{a,b,c,d,e,h}	{f,g}	(f,b)=3,**(g,c)=3,(g,h)=3**	S_1		**(f,g)=3**
{a,b,d,e}	{f,g,c,h}	(f,b)=3,(h,d)=4,**(h,e)=3**	S_1		**(g,c)=3,(g,h)=3**
{a,b,d}	{f,g,c,h,e}	(f,b)=3,(h,d)=4,**(e,a)=2**	S_1		**(h,e)=3**
{b,d}	{f,g,c,h,e,a}	((f,**b**)=3,(h,**d**)=4,**(a,b)=1,(a,d)=3**	S_1,S_2		**(e,a)=2**
{ ∅ }	{f,g,c,h,e,a,b,d}	알고리즘 종료			**(a,b)=1,(a,d)=3**
					-

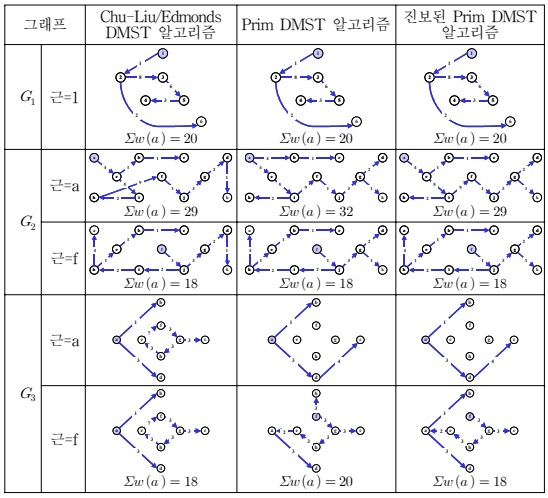


그림 5. 3개 그래프의 DMST
Fig. 5. DMST of 3 Graphs

Ⅳ. 진보된 Prim DMST 알고리즘 적용성 평가

1. $In-degree=0$ 노드가 존재하는 그래프 적용

본 절에서는 그림 6과 같이 그래프 상으로 근 노드를 명확히 알 수 있는 11개 그래프를 대상으로 진보된 Prim DMST 알고리즘의 적용성을 평가해 본다. G_4와 G_5 그래프는 Chen[17]에서, G_6 그래프는 Wenger[18], G_7 그래프는 Llewelly[15]에서, G_8 그래프는 Forbes[19]에서, G_9, G_{10}, G_{11}, G_{12} 그래프는 Ikeda[16]에서, G_{13}, G_{14} 그래프는 List[20]에서 인용되었다.

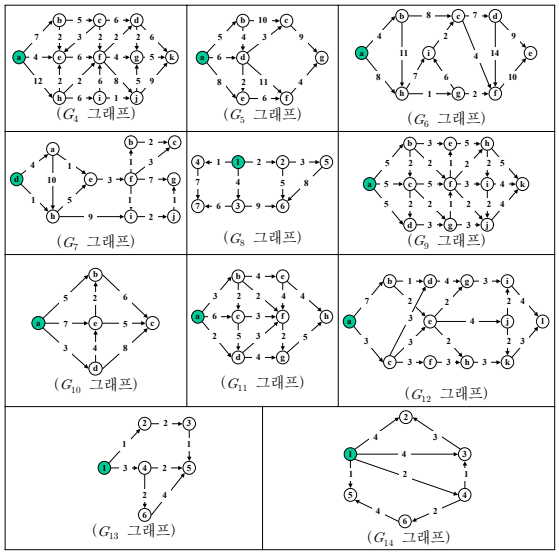


그림 6. $In-degree=0$ 노드 존재 그래프
Fig. 6. Graph with $In-degree=0$Node

근 노드가 지정된 그래프는 인접행렬에서 $In-degree=0$인 열 노드가 반드시 하나는 존재하는 경우이며, 예로 G_4 그래프에서는 ⓐ가 이에 해당된다. 노드 ⓐ는 $In-degree=0$ $Out-degree=3$인 경우이다. 11개의 그래프 모두 근 노드가 지정된 경우이며, 각 열 노드의 MWA를 선택(단, 동일한 가중치를 갖는 In-degree가 다수 존재하면 DST의 조건을 만족시키지 못하므로 하나만 선택한다.)하면 사이클이 발생하지 않는 DMST를 얻을 수 있다. 즉, Chu-Liu/Edmonds와 진보된 Prim DMST 알고리즘이 동일한 MST를 얻는다. 이렇게 얻은 DMST를 그림 7에 제시하였다. 알고리즘 적용성 평가에 사용된 $In-degree=0$인 노드가 존재하는 12개 그래프에 대한 알고리즘 수행 결과를 종합한 데이터는 표 5에 제시하였다. 표에서, Chu-Liu/Edmonds의 수행 횟수는 각 노드의 유입 MWA 수에 대한 사이클 발생 여부를 확인하는 과정과 사이클이 발생하였을 경우 선택되는 사이클 유입 MWA의 수를 의미하며, Prim과 진보된 Prim DMST 알고리즘의 수행속도는 n_{mst}에 추가되는 노드의 횟수를 의미한다. 이들 횟수가 알고리즘 수행속도를 결정한다. 수행속도는 Prim MST 알고리즘이 최대 $n-1$번 수행하는 횟수를 기준으로 수행속도 비율을 계산하였다. $In-degree=0$인 노드를 근 노드로 지정할 수 있는 12개

그래프는 Chu-Liu/Edmonds와 진보된 Prim DMST 알고리즘 모두 동일한 결과를 얻음을 알 수 있다. 반면에 Prim MST 알고리즘은 12개 그래프 중 7개 그래프만이 동일한 MST를 얻었으며, 5개 그래프에서는 MST를 얻지 못하였음을 알 수 있다. 또한, 진보된 Prim DMST 알고리즘은 G_9 그래프를 제외한 11개 그래프에서 Prim DMST 알고리즘보다 수행횟수를 단축시킬 수 있어 알고리즘 수행속도를 향상시킬 수 있다.

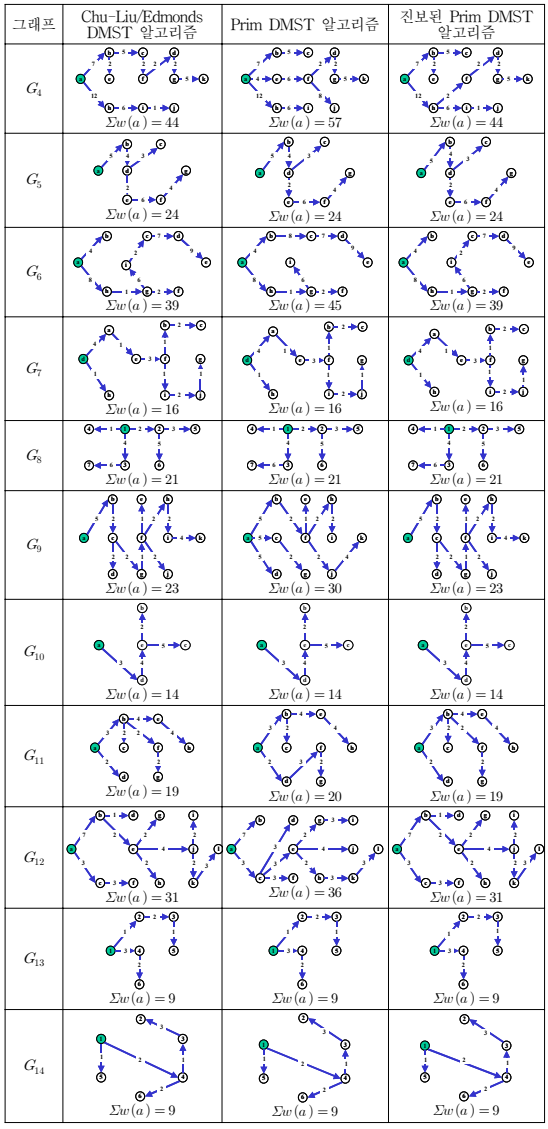


그림 7. $In-degree=0$ **노드 존재 그래프의 DMST**
Fig. 7. DMST of Graph with $In-degree=0$ Node

표 5. $In-degree=0$ **노드 존재 그래프의 MST 알고리즘 비교**
Table 5. Algorithm Comparison of Graph with $In-degree=0$ Node

그래프	$\|n\|$	$\|a\|$	Chu-Liu/Edmonds DMST 알고리즘		Prim DMST 알고리즘			진보된 Prim DMST 알고리즘		
			$\Sigma w(e)$	수행횟수	$\Sigma w(e)$	수행횟수	수행속도	$\Sigma w(e)$	수행횟수	수행속도
G_1	6	11	**20**	9 (5+4)	**20**	5	100%	**20**	**5**	**100%**
G_4	11	22	**44**	10	57	10	100%	**44**	**5**	**50%**
G_5	7	11	**24**	6	**24**	6	100%	**24**	**5**	**83%**
G_6	9	14	**39**	8	45	7	88%	**39**	**6**	**75%**
G_7	10	14	**16**	9	**16**	7	78%	**16**	**6**	**67%**
G_8	7	9	**21**	6	**21**	6	100%	**21**	**2**	**33%**
G_9	11	22	**23**	10	30	**6**	**60%**	**23**	**7**	**70%**
G_{10}	5	8	**14**	4	**14**	4	100%	**14**	**3**	**75%**
G_{11}	8	14	**19**	7	20	5	71%	**19**	**3**	**43%**
G_{12}	12	18	**31**	11	36	7	64%	**31**	**5**	**45%**
G_{13}	6	7	**9**	5	**9**	5	100%	**9**	**3**	**60%**
G_{14}	6	8	**9**	5	**9**	5	100%	**9**	**3**	**60%**

2. $In-degree=0$ 노드가 존재하지 않는 그래프 적용

본 절에서는 그림 8와 같이 그래프 상으로 근 노드를 명확히 알 수 없는 3개 그래프를 대상으로 진보된 Prim DMST 알고리즘의 적용성을 평가해 본다. G_{15}, G_{16} 그래프는 Boyd[7]에서, G_{17} 그래프는 List[20]에서 인용되었다.

근 노드가 지정되지 않은 그래프 (즉, 인접행렬에서 $In-degree=0$인 열 노드가 없을 경우)에 대해 Chu-Liu/Edmonds 알고리즘은 먼저 임의의 노드를 근으로 가정하고 알고리즘을 수행하였으며, 진보된 Prim DMST 알고리즘은 근 노드를 가정하지 않고 알고리즘을 수행하였다. 마지막으로 진보된 Prim DMST 알고리즘으로 얻은 근에 대해 Chu-Liu/Edmonds 알고리즘을 수행한 결과를 제시하였다. 알고리즘 수행 결과는 그림 9와 같다.

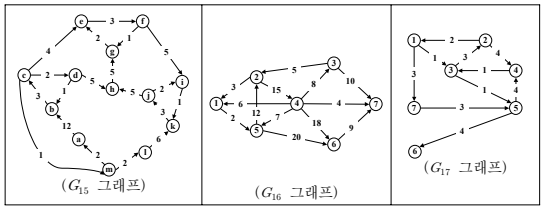


그림 8. $In-degree=0$ **노드 미 존재 그래프**
Fig. 8. Graph without $In-degree=0$ Node

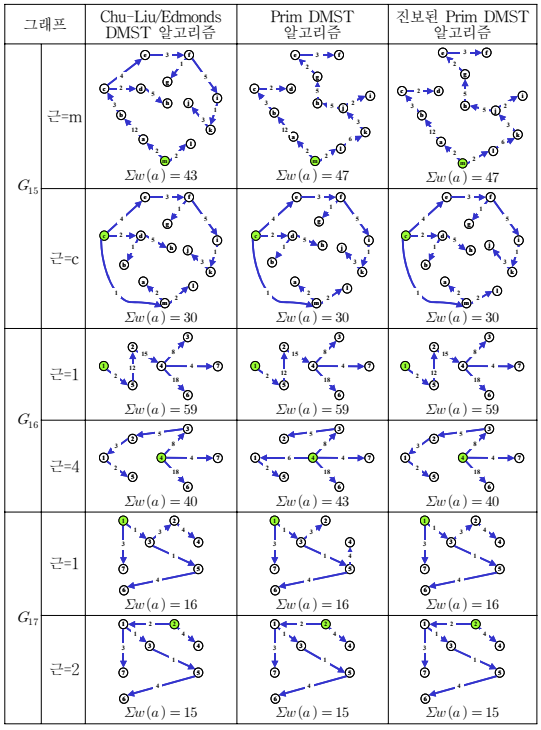


그림 9. $In-degree=0$ **노드 미 존재 그래프의 DMST**
Fig. 9. DMST of Graph without $In-degree=0$**Node**

알고리즘 적용성 평가에 사용된 $In-degree=0$인 노드가 존재하지 않는 5개 그래프에 대한 알고리즘 수행 결과를 종합한 데이터는 표 6에 제시하였다.

표 6. $In-degree=0$ **노드 미 존재 그래프의 MST 알고리즘 비교**
Table 6. Algorithm Comparison of Graph without $In-degree=0$**Node**

그래프	|n|	|a|	근 노드	Chu-Liu/Edmonds DMST 알고리즘		Prim DMST 알고리즘			진보된 Prim DMST 알고리즘		
				$\Sigma w(e)$	수행 횟수	$\Sigma w(e)$	수행 횟수	수행 속도	$\Sigma w(e$	수행 횟수	수행 속도
G_2	11	19	ⓐ	**29**	12 (10+2)	32	9	90%	**29**	**8**	**80%**
			ⓕ	**18**	10	**18**	10	100%	**18**	**6**	**60%**
G_3	8	12	ⓐ	**실패**	–	**실패**	–	–	**실패**	**–**	**–**
			ⓕ	**18**	7	20	5	71%	**18**	**5**	**71%**
G_{15}	13	18	ⓜ	**43**	15 (12+3)	47	11	91%	**47**	**10**	**83%**
			ⓒ	**30**	14 (12+2)	**30**	9	75%	**30**	**8**	**67%**
G_{16}	7	13	①	**59**	6	**59**	6	100%	**59**	**4**	**67%**
			④	**40**	6	43	6	100%	**40**	**4**	**67%**
G_{17}	7	10	①	**16**	6	**16**	4	67%	**16**	**3**	**50%**
			②	**15**	6	**15**	5	83%	**15**	**4**	**67%**

$In-degree=0$인 노드가 없는 5개의 그래프에서 임의의 노드를 근 노드로 지정할 경우, G_3 그래프의 근 노드 ⓐ를 제외한 모든 경우에 대해 진보된 Prim DMST 알고리즘은 Chu-Liu/Edmonds DMST 알고리즘과 동일한 MST를 얻는데 성공하였다. 또한, Prim DMST 알고리즘은 5가지 경우에 대해서만 MST를 얻는데 성공하였다. 알고리즘 수행속도 측면에서도 9가지 경우 모두 진보된 Prim DMST 알고리즘이 가장 좋은 수행속도를 나타내었다.

V. 결론 및 향후 연구과제

본 논문에서는 무방향 그래프의 최소신장트리를 얻는 Prim MST 알고리즘을 변형시켜 방향그래프의 최소신장트리를 얻는 알고리즘을 제안하였다. 먼저 Prim MST 알고리즘을 방향 그래프에 적용하기 위해 수정된 알고리즘을 제안하였다. 다음으로, $In-degree=0$을 갖는 노드가 존재하는 1개 그래프와 $In-degree=0$을 갖는 노드가 없는 2개 그래프에 대해 Prim DMST 알고리즘과 Chu-Liu/Edmonds DMST 알고리즘을 적용하여 문제점을 고찰하였다. 이러한 문제점을 해결하기 위해 Prim DMST 알고리즘을 수정한 진보된 Prim DMST 알고리즘을 제안하였다. 진보된 Prim DMST 알고리즘은 Prim DMST 알고리즘과 동일하게 노드를 신장시키는 방법을 사용하고 있지만 노드를 신장시키는 과정에서 해당 노드의 유입 MWA를 선택하지 않고 그래프의 각 노드의 최소 가중치를 갖는 호와 비교하여 이들 호를 선택하는 방법을 적용하였다.

진보된 Prim DMST 알고리즘은 $In-degree=0$을 갖는 노드가 존재하는 그래프와 $In-degree=0$을 갖는 노드가 없는 그래프 등 17개 그래프에 적용한 결과, 항상 Chu-Liu/Edmonds DMST 알고리즘과 동일한 DMST를 얻을 수 있었으며, Prim MST 알고리즘에 비해 알고리즘 수행시간도 크게 단축시킬 수 있었다.

진보된 Prim DMST 알고리즘은 방향 그래프의 근 노드로부터 모든 노드로 유일한 방향 경로를 가진 DMST를 구할 수 있었다. 이를 기반으로 하여 추후 임의의 노드로부터 시작하여 종점 노드로의 최단거리 경로(Shortest Path)를 찾는 알고리즘으로 적용할 수 있는지를 연구하고자 한다.

참 고 문 헌

[1] Wikipedia, http://en.wikipedia.org/wiki/Minimum_spanning_tree, Wikimedia Foundation Inc., 2007.

[2] O. Borůvka, "O Jistem Problemu Minimalnim," Prace Mor. Prrodved. Spol. V Brne (Acta Societ. Natur. Moravicae), Vol. III, No. 3, pp. 37-58, 1926.

[3] J. Nešetřil, E. Milková, and H. Nešetřilová, "Otakar Borůvka on Minimum Spanning Tree Problem (Translation of the both 1926 Papers, Comments, History)," DMATH: Discrete Mathematics, Vol. 233, 2001.

[4] R. C. Prim, "Shortest Connection Networks and Some Generalisations," Bell System Technical Journal, Vol. 36, pp. 1389-1401, 1957.

[5] J. B. Kruskal, "On the Shortest Spanning Subtree and The Traveling Salesman Problem," Proceedings of the American Mathematical Society, Vol. 7, pp. 48-50, 1956.

[6] P. A. Jensen, "Operations Research Models and Methods," John Wiley and Sons, 2003.

[7] S. Boyd, "Applications of Combinational Optimization: Optimal Paths and Trees," http://www.site.uottawa.ca/~sylvia/csi5166 web /5166tespg26to61.pdf, School of Information Technology and Engineering (SITE), University of Ottawa, Canada, 2005.

[8] C. Duhamel, L. Gouveia, P. Moura, and M. C. Souza, "Models and Heuristics for a Minimum Arborescence Problem," Research Report LIMOS/RR-04-13, http://www.isima.fr/limos/publi/RR-04-13.pdf, 2004.

[9] S. J. Yang, "The Directed Minimum Spanning Tree Problem," http://www.ce.rit.edu/~sjyeec/dmst.html, 2000.

[10] A. Schrijver, "Advanced Graph Theory and Combinatorial Optimization," Dept. of Mathematics, University of Amsterdam, Netherlands, http://citeseer.ist.psu.edu/schrijver01advanced.html, 2001.

[11] N. Bezroukov, "Minimum Spanning Trees," Softpanorama, http://www.softpanorama.org/Algorithms/Digraphs/mst.shtml, 2006.

[12] Wikipedia, "Tree(Graph Theory)," http://en.wikipedia.org/wiki/Tree_graph_Theory/, 2007.

[13] R. Krishnam, B and Raghavachari, "The Directed Minimum-Degree Spanning Tree Problem," FSTTCS 2001, LNCS 2245, pp. 232-243, 2001.

[14] T. UNO, "An Algorithm for Enumerating all Directed Spanning Trees in a Directed Graph," International Symposium on Algorithm and Computation(ISAAC96), pp. 166-173, 1996.

[15] M. Llewellyn, "COP 3503: Computer Science II - Introduction to Graphs," http://www.cs.ucf.edu/courses/cop3503/summer04, 2004.

[16] K. Ikeda, "Mathematical Programming," Dept. Information Science and Intelligent Systems, The University of Tokushima, http:// www-b2.is.tokushima-u.ac.jp/~ikeda/suuri/`maxflow /MaxflowApp. shtml. 2005.

[17] WWL. Chen, "Discrete Mathematics," Department of Mathematics, Division of ICS, Macquarie University, Australia, http://www.maths.mq.edu.au/~wchen/lndmfolder/lndm.html, 2003.

[18] R. Wenger, "CIS 780: Analysis of Algorithms," http://www.cse.ohio-state.edu/~wenger/cis780/shortest_path.pdf, 2004.

[19] J. R. N. Forbes, "CPS196.2 Robotics: Graphs and Matrices," http://www.cs.duke.edu/courses/cps196.2/fall03/pdf/graphs_and_ matrices.pdf

[20] C. List, "MT303: Operations Research Graphs & Networks - Handout 3," Department of Mathematics, National University of Ireland, http://www,maths.may.ie/clist/teching/netw_handout_3 .pdf, 2003.

[21] H. K. Park, C. H. Lee, "Embedded System Implementation of Tree Routing Structure for Ubiquitous Sensor Network," Journal of the Korea Academia-Industrial cooperation Society, v.11 no.10, pp.4531-4535, October 2011.

[22] W. J. Wang, C. S. Han, "Bond Graph Modeling, Analysis and Control of Dual Stage System,"

Journal of the Korea Academia-Industrial cooperation Society, v.13, no.4, pp.1453-1459, April 2012.

저자 소개

최 명 복(종신회원)

- 1992년 : 호서대학교 전자계산학과 (학사)
- 1994년 : 아주대학교 컴퓨터공학과 (석사)
- 2001년 : 아주대학교 컴퓨터공학과 (박사)
- 1997~현재 : 강릉원주대학교 멀티미디어공학과 교수
- 2004. 1~현재 : 한국인터넷방송통신학회 이사

<주관심분야 : 지능형 정보검색, 소프트웨어 공학, 알고리즘 등>

이 상 운(정회원)

- 1983년~1987년 : 한국항공대학교 항공전자공학과 (학사)
- 1995년 ~ 1997년 : 경상대학교 컴퓨터과학과 (석사)
- 1998년 ~ 2001년 : 경상대학교 컴퓨터과학과 (박사)
- 2003년 : 강원도립대학 컴퓨터응용과 전임강사
- 2004년 ~ 2007.2 : 국립 원주대학 여성교양과 조교수
- 2007.3 ~ 현재 : 강릉원주대학교 멀티미디어공학과 부교수

<주관심분야 : 소프트웨어 프로젝트 관리, 소프트웨어 신뢰성, 신경망, 뉴로-퍼지, 그래프 알고리즘 등>