http://dx.doi.org/10.7236/JIWIT.2012.12.6.165
JIWIT 2012-6-21

일반화된 Borůvka 최소신장트리 알고리즘

Generalized Borůvka's Minimum Spanning Tree Algorithm

최명복*, 이상운**

Myeong-Bok Choi, Sang-Un Lee

요 약 무방향성, 가중치를 가진 그래프에서 최소신장트리(Minimum Spanning Tree, MST)는 사이클이 발생하지 않으면서 모든 정점들을 간선들로 연결한 그래프로 간선들의 가중치 합이 최소가 되어야 한다. 최소신장트리(MST)를 구하는 알고리즘으로 Borůvka 알고리즘이 가장 먼저 제안되었지만 일반적으로 사용되지 않고, Prim과 Kruskal 알고리즘이 일반적으로 널리 알려져 왔다. Borůvka 알고리즘은 각 정점에서 최소 가중치를 갖는 간선(Minimum Weight Edge, MWE)을 선택하고 사이클을 제거하는 1st Stage와 MSF(Minimum Spanning Fores)의 MWE를 선택하는 2nd Stage를 수행한다. 이 과정은 시각적으로는 쉽게 MWE를 구하지만 프로그램으로 구현하는데 어려움이 있다. 본 논문은 일반화된 Borůvka 알고리즘을 제안한다. 1st Stage에서 각 정점에서 MWE들을 모두 선택하고, Kruskal 방법을 도입하여 오름차순으로 정렬된 MWE들에 대해 사이클의 최대 가중치 간선을 제거하면서 MSF를 형성시킨다. 만약, MSF가 1개 이상 발생하면 2nd Stage에서 MSF 간선들을 오름차순으로 정렬시켜 MWE를 선택하였다. 제안된 알고리즘을 7개의 여러 간선들 가중치가 동일하거나 상이한 그래프에 적용하여 알고리즘 적합성을 검증하였다. 검증 결과, 일반화된 Borůvka 알고리즘은 사이클 검증에 요구되는 간선 수가 Kruskal 알고리즘보다 적어 보다 빠르게 MST를 구할 수 있었다.

Abstract Given a connected, weighted, and undirected graph, the Minimum Spanning Tree (MST) should have minimum sum of weights, connected all vertices, and without any cycle taking place. Borůvka Algorithm is firstly suggested as an algorithm to evaluate the MST, but it is not widely used rather than Prim and Kruskal algorithms. Borůvka algorithm selects the Minimum Weight Edge (MWE) from each vertex with distinct weights in 1st stage, and selects the MWE from each MSF (Minimum Spanning Forest) in 2nd stage. But the cycle check and the number of MSF in 1st stage and 2nd stage are difficult to implication by computer program even if it is easy to verify visually. This paper suggests the generalized Borůvka Algorithm, This algorithm selects all of the same MWEs for each vertex, then checks the cycle and constructs MSF for ascending sorted MWEs. Kruskal method bring into this process. if the number of MSF greats then 1, this algorithm selects MWE from ascending sorted inter-MSF edges. The generalized Borůvka algorithm is verified its application by being applied to the 7 graphs with the many minimum weights or distinct weight edges for any vertex. As a result, the generalized Borůvka algorithm is less required for cycle verification then the Kruskal algorithm. Therefore, the generalized Borůvka algorithm is more fast to obtain MST then Kruskal algorithm.

Key Words : Minimum Spanning Tree, Undirected, Distinct Weights, Same Weights, Cycle

*종신회원, 강릉원주대학교, 멀티미디어공학과
**정회원, 강릉원주대학교 멀티미디어공학과
접수일자 : 2012년 8월 9일, 수정완료 : 2012년 10월 25일
게재확정일자 : 2012년 12월 14일

Received: 9 August 2012 / Revised: 25 October 2012 / Accepted: 14 September 2012
**Corresponding Author: cmb5859@gmail.com
Dept. of Multimedia Engineering, Gangnung-Wonju National University Wonju Campus, Korea

Ⅰ. 서 론

그래프 $G=(V,E)$는 정점들 (Vertices, V)간에 간선들(Edges, E)로 연결되어 (Connected)있다. 그래프는 방향성 (Directed)과 무방향성 (Undirected)으로 구분된다. 또한 간선이 가중치를 가지고 있는 경우 (Weighted)와 없는 경우 (Unweighted)로 구분된다. 그래프가 연결되어 있고 무방향성이며, 가중치를 갖고 있는 경우 신장트리 (Spanning Tree, ST)를 구할 수 있다. ST는 사이클이 발생하지 않으면서 모든 정점들을 간선으로 연결한 트리로 $|E|=|V|-1$ 이다. 하나의 그래프에는 다수의 ST가 존재할 수 있으며, 최소신장트리(Minimum Spanning]Tree, MST)는 ST에 관여된 간선들의 가중치 합 $(\Sigma w(e))$이 최소가 되는 경우로 전기, 전화, 가스 또는 수도 분야에 활용될 수 있다.[1] 본 논문은 MST 알고리즘에 초점을 맞춘다.

대표적인 MST 알고리즘으로는 Borůvka[2,3], Prim[4], Kruskal[5]과 역-삭제(Reverse-Delete) 알고리즘[6]이 있다. Borůvka 알고리즘[2,3]은 모든 간선들의 가중치가 서로 상이한 (Distinct) 그래프에서 MST를 찾는 알고리즘으로 1926년에 처음 제안되었으며, 현재는 1950년대에 제안된 Prim과 Kruskal 알고리즘이 널리 사용되고 있다.[1]

MST를 구할 경우, 주어진 그래프에서 임의의 한 정점에 부속된 간선들의 가중치 (길이, 또는 비용)가 모두 다른 값을 갖는 경우, 모두 동일한 값을 갖는 경우와 일부분만이 동일한 값을 갖는 경우로 분류될 수 있다.

하나의 정점에 부속된 모든 간선들이 동일한 가중치를 갖는 경우 하나의 정점에서 임의로 하나의 간선을 선택하면 MST를 얻을 수 있기 때문에 굳이 MST 알고리즘을 적용할 필요가 없다. 또한, 모든 간선들의 가중치가 서로 상이하다면 그래프의 MST는 유일하게 얻을 수 있다.[7] 그러나 여러 간선들이 동일한 가중치를 갖는 경우에 어떤 간선을 선택할 것인가에 대한 문제가 제기될 수 있다. 즉, 임의로 하나만 선택할 것인가? 아니면 모두 선택할 것인가?. 만약, 모든 간선들을 선택할 경우 사이클이 발생하며, 사이클을 제거하기 위해 해당 사이클에서 최대 가중치 간선을 제거해야만 한다.

Erickson[7]은 "만약 모든 간선들의 가중치가 상이한 그래프에 적용 가능한 MST 알고리즘이 문제를 해결하는 일관된 방법을 갖고 있다면, 여러 간선들이 동일한 가중치를 갖는 그래프에도 적용될 수 있다."고 하였다.

Borůvka[2,3] 알고리즘은 하나의 정점에서 동일한 최소 가중치 간선을 모두 선택할 경우 사이클을 제거하는 방법은 제안하고 있지 않다. 본 논문은 하나의 정점에 부속된 최소 가중치 간선들이 다수 존재하는 경우에도 Borůvka 알고리즘을 적용할 수 있도록 일반화한 "일반화된 Borůvka 알고리즘" (Generalized Borůvka Algorithm)을 제안한다.

2장에서는 대표적인 Borůvka, Prim, Kruskal과 역-삭제 알고리즘을 고찰한다. 또한, 하나의 정점에 부속된 최소 가중치 간선들이 1개만 존재하는 (Distinct) 경우와 다수가 존재하는 사례에 Borůvka 알고리즘을 적용하고 문제점을 고찰해 본다. 3장에서는 하나의 정점에 부속된 최소 가중치 간선이 다수 존재하는 경우에도 적용할 수 있는 "일반화된 Borůvka 알고리즘"을 제안한다. 4장에서는 다양한 간선들 가중치를 갖는 그래프에 실제 적용하여 제안된 알고리즘의 적용성을 검증해본다.

Ⅱ. 관련연구와 연구 배경

1. MST 알고리즘 고찰

Borůvka 알고리즘에 대해서는 다음과 같이 다양한 방법으로 설명되고 있다.[3]

(1) Nešetřil et al.[3]: 그래프의 모든 간선들의 가중치가 상이하다고 가정하고, 각 정점 u에 부속된 간선들 중 (Minimum- weight Edge, MWE)를 선택한다. MWE로 연결된 정점들로 구성된 트리를 하나의 정점으로 취급한다. 이 과정에서 MWE 이외의 루프 (간선 $\{u,v\}$와 $\{v,u\}$가 동일한 트리에 존재)와 병렬 (동일한 트리 쌍 $\{F_a,F_b\}$간에 연결된 간선들)을 삭제한다. 축소된 그래프의 트리들이 다수 존재시 이들을 하나의 트리로 만드는 신장트리 (Blue Spanning Tree) T를 찾기 위해 알고리즘을 반복적으로 수행한다.

(2) Pe'er[8]: 각 정점 u에 대해 MWE를 선택한다. 선택된 간선들로 연결된 구성품 (Component)을 하나의 정점 (Meta Vertex)으로 축소 교체한다. Meta Vertex 간에 MWE를 제외한 모든 루프를 삭제한다.

(3) Wikipedia[9]:$\{u,v\}$가 이전에 u에서 선택되었든지 상관없이 정점 v에 연결된 MWE가 $\{v,u\}$이면 중복에 상관없이 선택한다. 이들 그룹을 모든 정점을 신장시키는 하나의 트리가 될 때까지 결합시킨다.

여기서 루프와 병렬을 “사이클” 용어로 통일시킨다. 또한 구성된 트리를 단일 정점, Meta Vertex, 구성품, Forest, 또는 MSF (Minimum Spanning Forest, F)로 다양하게 부르고 있다.[3,8,9-13] 본 논문에서는 이를 “MSF”라 한다. MSF들이 MWE들로 하나의 트리로 연결되면 MST가 된다.

Borůvka 알고리즘을 종합하면, 각 정점에서 MWE를 선택 하고 사이클이 발생하는 간선들을 삭제한다. 다음으로 MSF 간 MWE를 선택하면 MST를 얻는다. 이 알고리즘은 그림 1에 제시하였다. 여기서 가중치가 동일한 MWE가 다수 존재시 모두 선택하는지 여부는 거론하고 있지 않다. 따라서 이 경우를 고려한 알고리즘으로 일반화시킬 필요가 있다.

```
[1st Stage]
for  1 to |v| do
     각 정점에서 최소 가중치 간선 (MWE) 1개 선택.
end
만약, 사이클이 존재시 사이클 제거.
|F| : MSF 개수.
if       |F|=1 then 알고리즘 종료
else if  |F|≥2 then 2nd Stage 수행
endif
[2nd Stage]
MSF 각각을 하나의 정점으로 치환, 정점간 MWE 선택.
```

그림 1. Borůvka 알고리즘
Fig 1. Borůvka Algorithm

Prim 알고리즘은 임의의 정점 u를 선택하고, 이 정점에 부속된 간선들 중에서 MWE인 $\{u,v\}$을 선택한다. 다음으로 새로 선택된 정점 v에 부속된 간선들의 가중치와 기존에 u에 부속된 간선들 중 MWE로 선택되지 않은 간선들 중에서 MWE를 선택하는 방법이다. 단, 이 과정에서 사이클이 발생하는 간선은 무시한다.

Kruskal 알고리즘은 그래프의 모든 간선들을 오름차순으로 정렬시키고, 첫 번째 MWE부터 시작하여 마지막까지 사이클이 발생하지 않는 간선들 (e_{mst})을 선택하는 방법이다. Kruskal 알고리즘은 그림 2와 같다. 역-삭제 알고리즘은 Kruskal 알고리즘을 반대로 수행하는 방법으로 간선들을 내림차순으로 정렬하고 최대 가중치 간선을 삭제하여도 정점들이 연결이 되어 있다면 최대 가중치 간선을 삭제하는 방법이다. 이 방법은 $|e|<2|v|$ 인 경우 Kruskal 알고리즘보다 빠를 수 있다. 그러나 정점들이 분리되는지 여부를 검증하는 자동화된 방법을 구현해야 한다.

```
F_i : MSF
for  i = 1 to |e|
     모든 간선들을 오름차순으로 정렬.
     MWE {u,v} 선택.
     if       (u∈F_1)∩(v∈F_1) then Skip /* 사이클
     else if  (u∈F_1)∩(v∈F_2) then F_1 = F_1 + F_2
     else if  (u∈F_1)∩(v∉F_i) then F_1 ← F_1 + {v}
     else if  (u∉F_i)∩(v∉F_i) then F_i ← {u,v} 생성.
     endif
end
```

그림 2. Kruskal 알고리즘
Fig 2. Kruskal Algorithm

그림 3은 Peiper[14]로부터 인용된 3개의 그래프를 예로 제시하고 있다. G_1 그래프는 각 정점에 부속된 간선들의 가중치가 모두 상이한 경우이며, G_2 그래프는 ⓓ와 ⓙ 정점의 MWE가 2개씩 존재한다. G_3 그래프는 16개 정점 중 10개 정점에서 MWE가 동일한 값들이 존재한다. 특히, ⓚ는 MWE=2가 3개, ⓙ는 MWE=1이 4개가 존재한다.

G_1 그래프에 대해 Borůvka와 Kruskal 알고리즘으로 MST를 구한 결과는 그림 4와 같다. Borůvka 알고리즘은 1st Stage에서는 다행히도 사이클이 발생하지 않았지만 2개의 F가 발생하였다. 따라서 2nd Stage에서 $\{F_a, F_b\}$간에 MWE인 $\{a,c\}=7$이 선택되었다.

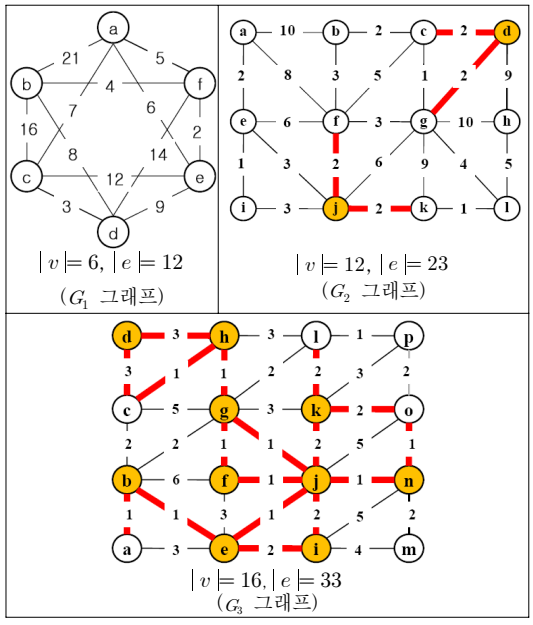
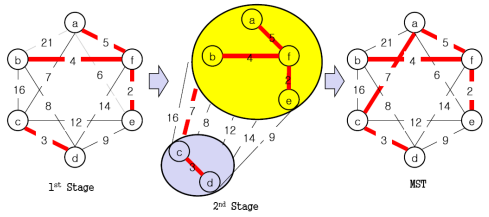


그림 3. 그래프
Fig 3. Graph

Kruskal 알고리즘은 $|e|=12$ 개를 오름차순으로 정렬하고 사이클이 발생하는 $\{a,e\}=6$을 제외하면 6번째인 $\{a,c\}=7$에서 $|e_{mst}|=|v|-1=5$를 얻었으며, 나머지 6개의 간선들은 모두 사이클이 발생하는 최대 가중치 간선들로 제외되었다.

결국, 어떠한 MST 알고리즘을 적용하여도 $|e_{mst}|=5$와 $\Sigma w(e)=21$을 얻을 수 있다. 그러나 얼마나 빨리, 쉽게 구할 수 있느냐의 알고리즘 효율성 문제가 제기된다. 즉, Kruskal 알고리즘은 12개 간선들 중에서 $|e_{mst}|=|v|-1=5$ 를 얻는 6번째에서 알고리즘을 종료시키면 6회로 단축시킬 수 있다. Borůvka 알고리즘은 1st Stage에서 6개 정점에서 4개의 간선을 찾고 2nd Stage에서 1개의 간선을 찾을 수 있다. 즉, Borůvka 알고리즘이 Kruskal 알고리즘에 비해 보다 빠르고 쉽게 MST를 얻을 수도 있다. 그러나 사이클 발생여부와 사이클의 최대 가중치 간선 삭제 방법과 더불어 2개의 F가 발생하였음을 자동적으로 검출해야 한다.



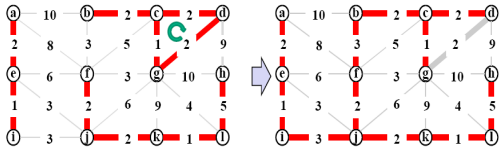
(a) Borůvka 알고리즘

All Edges	오름차순 정렬	사이클 Check	MSF	MST
{a,b}=21	{e,f}=2	-	-	{e,f}=2
{a,c}=7	{c,d}=3	x	{e-f}	{c,d}=3
{a,e}=6	{b,f}=4	x	{e-**f**}, {c-d}	{b,f}=4
{a,f}=5	{a,f}=5	x	{b-e-**f**}, {c-d}	{a,f}=5
{b,c}=16	{a,e}=6	O	**{a**-b-**e**-f}, {c-d}	-
{b,d}=8	{a,c}=7	x	**{a**-b-e-f}, {**c**-d}	{a,c}=7
{b,f}=4	{b,d}=8	O	{a-b-c-d-e-f}	-
{c,d}=3	{d,e}=9	O	{a-b-c-d-e-f}	-
{c,e}=12	{c,e}=12	O	{a-b-c-d-e-f}	-
{d,e}=9	{d,f}=14	O	{a-b-c-d-e-f}	-
{d,f}=14	{b,c}=16	O	{a-b-c-d-e-f}	-
{e,f}=2	{a,b}=21	O	{a-b-c-d-e-f}	-
$\|e_c\|=12$			$\|e_{mst}\|=5$	

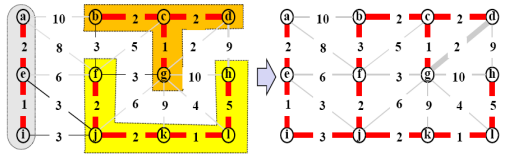
(b) Kruskal 알고리즘

그림 4. G_1 **그래프의 MST**
Fig 4. MST of Graph G_1

Borůvka 알고리즘을 G_2 그래프에 적용시 "동일한 가중치를 갖는 모든 MWE를 선택하는 방법"을 적용하였다. 알고리즘 수행 결과는 그림 5과 같다. 1st Stage를 수행한 결과 1개의 사이클이 발생하였으며, 사이클에서 최대 가중치 간선을 자동적으로 삭제해야 한다. 또한, 3개의 F가 발생하였기 때문에 2nd Stage에서 이들을 연결하는 MWE를 자동적으로 선택해야 한다. 따라서 사이클이 발생하였는지 여부와 사이클을 어떻게 자동적으로 제거할 수 있는가와 다수의 F를 하나의 MST로 자동적으로 연결시키는 문제가 발생한다.



(a) 1st Stage



(b) 2nd Stage

그림 5. G_2 **그래프의 Borůvka 알고리즘 적용 MST**
Fig 5. MST of Graph G_2 applied to Borůvka Algorithm

Borůvka 알고리즘을 G_3 그래프에 적용한 결과는 그림 6에 제시되어 있다. 이 문제에 대해서도 동일한 가중치를 갖는 MWE를 모두 선택하는 방법을 적용하였다. 1st Stage를 수행한 결과 4개의 사이클이 발생하였으며, 사이클에서 최대 가중치 간선을 자동적으로 삭제해야 한다. 그러나 다행스럽게도 1st Stage에서 모든 정점들이 연결되어 MST를 빠르게 얻어 2nd Stage는 수행되지 않는다.

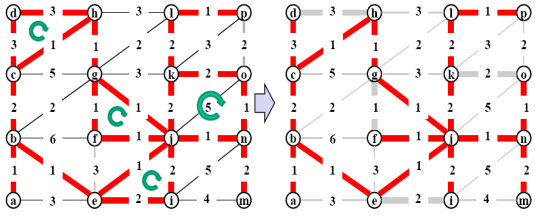


그림 6. G_3 **그래프의 Borůvka 알고리즘 적용 MST**
Fig 6. MST of Graph G_3 applied to Borůvka Algorithm

2. 연구 배경

모든 간선들이 상이한 가중치를 갖는 경우 Borůvka 알고리즘을 적용하기는 어렵지 않다. 그러나 하나의 정점에 부속된 MWE가 다수 존재시 모두 선택하는 경우, 1st Stage를 수행하면 사이클이 다수 존재할 수 있다. 이를 해결하는 방법은 가시적인 수작업 방법으로는 쉽게 해결할 수 있지만 컴퓨터를 활용한 알고리즘으로 자동으로 수행할 수 있는 방법은 제안되지 않고 있다. 이는 1926년에 컴퓨터가 존재하지 않던 시대에 제안된 알고리즘이기 때문이다. 이러한 이유로 인해 Borůvka 알고리즘이 Prim이나 Kruskal 알고리즘에 비해 보다 간단하면서도 빠르게 MST를 구할 수 있음에도 불구하고 일반적으로 적용되지 않고 있다고 생각된다.

따라서 3장에서는 MWE가 다수 존재하는 경우에도 Borůvka 알고리즘을 자동적으로 적용할 수 있는 "일반화된 Borůvka 알고리즘"을 제안한다.

Ⅲ. 일반화된 Borůvka 알고리즘

본 장에서는 하나의 정점에 부속된 MWE가 다수 존재하는 경우에 모두 선택하는 방법으로 Borůvka 알고리즘을 적용하는 "일반화된 Borůvka 알고리즘"을 다음과 같이 제안한다.

(1) 각 정점에서 MWE를 선택하여 e_{mst}에 저장한다. 만약 동일한 가중치 간선들이 다수 존재시 모두 선택한다. 간선 e_F에서 e_{mst}를 삭제한다.

(2) e_{mst}를 대상으로 오름차순으로 정렬시켜 Kruskal 알고리즘을 적용하여 사이클의 최대 가중치 간선을 삭제하고 $|F|$ 를 구한다.

(3) 만약, $|F| \geq 2$ 이면 각 F내의 정점들 간에 e_{mst}가 아닌 간선들을 e_F에서 삭제한다. e_F를 오름차순으로 정렬하여 MWE를 선택하면서 $|F| = 1$ 이 되면 종료한다.

일반화된 Borůvka 알고리즘은 그림 7에 제시되어 있다. 제안된 알고리즘은 모든 간선들 e_c를 대상으로 하는 Kruskal 알고리즘에 비해 1st Stage에서 얻은 e_{mst} 만으로 MST를 얻는 경우에는 Kruskal 알고리즘 보다 빠르게 MST를 얻을 수 있다. 2nd Stage를 수행하는 경우에는 축소된 e_F (MSF들 간의 간선)만을 대상으로 하기 때문에 Kruskal 알고리즘을 $|e_{mst}| = |v| - 1$까지 수행하는 횟수까지 수행할 수도 있다.

[1st Stage]
$e_F \leftarrow e$. /* Inter-MSF 간선.
for $i = 1$ to $|v|$ do
  각 정점에서 최소 가중치 간선 (MWE) 선택, e_{mst}**에 저장,** e_F**에서 삭제**.
end
e_{mst}에서 $\{u,v\}$와 $\{v,u\}$ 존재시 $\{v,u\}$ 삭제.
/* e_{mst}에 대해 Kruskal 알고리즘 수행.
e_{mst}를 오름차순으로 정렬.
for $i = 1$ to $|e_{mst}|$ do
  e_{mst}에서 MWE $\{u,v\}$ 선택.
  if $(u \in F_1) \cap (v \in F_1)$ then skip /* 사이클
  else if $(u \in F_1) \cap (v \in F_2)$ then $F_1 = F_1 + F_2$
  else if $(u \in F_1) \cap (v \notin F_i)$ then $F_1 \leftarrow F_1 + \{v\}$
  else if $(u \notin F_i) \cap (v \notin F_i)$ then $F_i \leftarrow \{u,v\}$ 생성.
  endif
  $e_F = e_F \setminus \{u,v\}$.
end
$|F|$: MSF 개수.
if $|F| = 1$ then 알고리즘 종료
else if $|F| \geq 2$ then 2nd Stage 수행.
endif
[2nd Stage] /* e_F에 대해 Kruskal 알고리즘 수행.
F_i 내부의 e_{mst} 이외 간선을 e_F에서 삭제.
e_F를 오름차순으로 정렬.
while $|F| = 1$
  e_F에서 MWE $\{u,v\}$ 선택.
  if $(u \in F_i) \cap (v \in F_j)$ then $F_i = F_i + F_j$, $|F| = F - 1$,
    $e_{mst} \leftarrow \{u,v\}$, F_i 내부의 e_{mst} 이외 간선을 e_F에서 삭제.
  endif
end

그림 7. 일반화된 Borůvka 알고리즘
Fig 7. Generalized Borůvka Algorithm

일반화된 Borůvka 알고리즘을 G_2 그래프에 적용한 결과는 표 1에, G_3 그래프에 적용한 과정은 표 2에 제시되어 있다.

표 1. G_2 **그래프의 일반화된 Borůvka 알고리즘 적용**
Table 1. Generalized Borůvka Algorithm Applying to Graph G_2

(a) 1st Stage

V	MWE	중복 제거		MSF	사이클	e_{mst}
a	{a,e}=2					
b	{b,c}=2					
c	{c,g}=1					
d	{d,c}=2 {d,g}=2	{a,e}=2 {b,c}=2 {c,g}=1 {d,c}=2 {d,g}=2 {e,i}=1 {f,j}=2 {h,l}=5 {j,k}=2 {k,l}=1	{c,g}=1 {e,i}=1 {k,l}=1 {a,e}=2 {b,c}=2 {d,c}=2 **{d,g}=2** {f,j}=2 {j,k}=2 {h,l}=5	–	x	{c,g}=1
e	{e,i}=1			{c-g}	x	{e,i}=1
f	{f,j}=2			{c-g}, {e-i}	x	{k,l}=1
g	{g,c}=1			{c-g}, {**e**-i}, {k-l}	x	{a,e}=2
h	{h,l}=5			{**c**-g}, {a-e-i}, {k-l}	x	{b,c}=2
i	{i,e}=1			{b-**c**-g}, {a-e-i}, {k-l}	x	{d,c}=2
j	{j,f}=2 {j,k}=2			{b-c-**d**-g}, {a-e-i}, {k-l}	O	–
k	{k,l}=1			{b-c-d-g}, {a-e-i}, {k-l}	x	{f,j}=2
l	{l,k}=1			{b-c-d-g}, {a-e-i}, {**k**-l}, {f-**j**}	x	{j,k}=2
				{b-c-d-g}, {a-e-i}, {f-j-k-**l**}	x	{h,l}=5
				{b-c-d-g}, {a-e-i}, {f-h-j-k-l}		
$\lvert e_{mst} \rvert = 10$				$\lvert F \rvert = 3$		$\lvert e_{mst} \rvert = 9$

(b) 2nd Stage

e_F		MSF	e_{mst}
{a,b}=10	{e,j}=3	{b-c-d-g}, {a-**e**-i}, {f-h-**j**-k-l}	{e,j}=3
{a,f}=8	{i,j}=3	–	–
{e,f}=6	{b,f}=3	{**b**-c-d-g}, {a-e-**f**-h-i-j-k-l}	{b,f}=3
{e,j}=3	{f,g}=3	{a-b-c-d-e-f-g-h-i-j-k-l}	(종료)
{i,j}=3	{g,l}=4		
{b,f}=3	{c,f}=5		
{c,f}=5	{e,f}=6		
{f,g}=3	{g,j}=6		
{g,j}=6	{a,f}=8		
{g,k}=9	{g,k}=9		
{g,l}=4	{d,h}=9		
{g,h}=10	{a,b}=10		
{d,h}=9	{g,h}=10		
	–	$\lvert F \rvert = 1$	$\lvert v \rvert = 12$, $\lvert e_{mst} \rvert = 11$

Ⅳ. 알고리즘 적용성 평가

본 장에서는 간선들 중 일부분이 동일한 가중치를 갖는 그림 8의 4개 그래프에 대해 일반화된 Borůvka 알고리즘을 적용하여 적합성을 검증해본다. G_4 와 G_7 그래프는 Chen[7]에서, G_5와 G_6 그래프는 Peiper[14]에서 인용되었다. 일반화된 Borůvka 알고리즘을 적용한 결과는 표 3 ~ 표 6에 제시되어 있다.

표 2. G_3 **그래프의 일반화된 Borůvka 알고리즘 적용**
Table 2. Generalized Borůvka Algorithm Applying to Graph G_3

(a) 1st Stage

V	MWE	중복 제거		MSF	사이클	e_{mst}
a	{a,b}=1					
b	{b,a}=1					
	{b,e}=1			–		
c	{c,h}=1			{a-**b**}		
d	{d,c}=3	{a,b}=1	{a,b}=1	{a-b-e}	x	{a,b}=1
	{d,h}=3	{b,e}=1	{b,e}=1	{a-b-**e**}, {c-h}	x	{b,e}=1
e	{e,b}=1	{c,h}=1	{c,h}=1	{a-b-e-j}, {c-h}	x	{c,h}=1
	{e,j}=1	{d,c}=3	{e,j}=1	{a-b-e-**j**}, {c-h}, {**f**-g}	x	{e,j}=1
f	{f,g}=1	{d,h}=3	{f,g}=1	{a-b-e-f-**g**-j}, {c-**h**}	x	{f,g}=1
	{f,j}=1	{e,j}=1	{f,j}=1	{a-b-c-e-f-**g**-h-**j**}	x	{f,j}=1
g	{g,f}=1	{f,g}=1	{g,h}=1	{a-b-c-e-f-g-h-**j**}	x	{g,h}=1
	{g,h}=1	{f,j}=1	**{g,j}=1**	{a-b-c-e-f-g-h-j-n}	O	–
h	{h,c}=1	{g,h}=1	{j,n}=1	{a-b-c-e-f-g-h-j-**n**}, {l-p}	x	{j,n}=1
	{h,g}=1	{g,j}=1	{l,p}=1	{a-b-c-**e**-f-g-h-j-n-o), {l-p}	x	{l,p}=1
i	{i,e}=2	{i,e}=2	{n,o}=1	{a-b-c-e-f-g-h-**i**-**j**-n-o), {l-p}	x	{n,o}=1
	{i,j}=2	{i,j}=2	{i,e}=2	{a-b-c-e-f-g-h-i-**j**-n-o), {l-p}	x	{i,e}=2
j	{j,e}=1	{j,n}=1	**{i,j}=2**	{a-b-c-e-f-g-h-i-j-**k**-n-o),	O	–
	{j,f}=1	{k,j}=2	{k,j}=2	{**l**-p)	x	{k,j}=2
	{j,g}=1	{k,l}=2	{k,l}=2	{a-b-c-e-f-g-h-i-j-**k**-l-n-**o**-p)	x	{k,l}=2
	{j,n}=1	{k,o}=2	**{k,o}=2**	{a-b-c-e-f-g-h-i-j-k-l-**n**-o-p)	O	–
k	{k,j}=2	{l,p}=1	{m,n}=2	{a-b-**c**-e-f-g-h-i-j-k-l-m-n-o	x	{m,n}=2
	{k,l}=2	{m,n}=2	{d,c}=3	-p}	x	{d,c}=3
	{k,o}=2	{d,c}=3	**{d,h}=3**	{a-b-c-**d**-e-f-g-**h**-i-j-k-l-m-n-	O	
l	{l,p}=1	{n,o}=1		o-p}		
m	{m,n}=2			{a-b-c-d-e-f-g-h-i-j-k-l-m-n		
n	{n,j}=1			-o-p}		
	{n,o}=1					
o	{o,n}=1					
p	{p,l}=1					
$\lvert v \rvert = 16$		$\lvert e_{mst} \rvert = 19$		$\lvert F \rvert = 1$		$\lvert e_{mst} \rvert = 15$

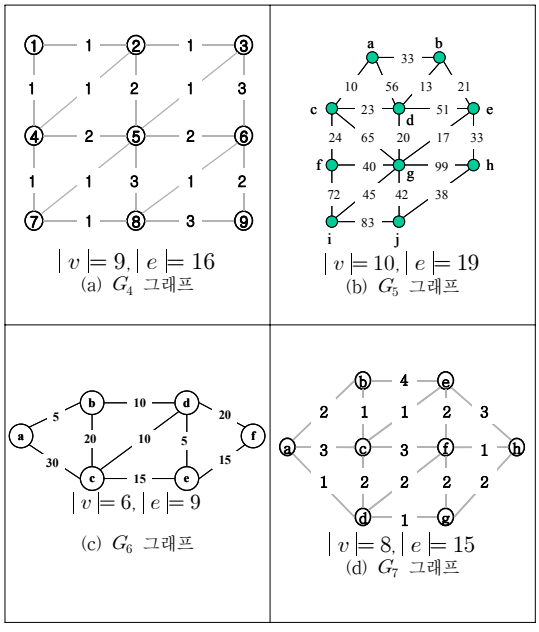


그림 8. 알고리즘 평가에 사용된 그래프
Fig 8. Graphs for Evaluation of Algorithm

표 3. G_4 그래프의 일반화된 Borůvka 알고리즘 적용
Table 3. Generalized Borůvka Algorithm Applying to Graph G_4

(a) 1st Stage

V	MWE	중복 제거		MSF	사이클	e_{mst}
1	{1,2}=1					
	{1,4}=1					
2	{2,1}=1					
	{2,3}=1	{1,2}=1	{1,2}=1	-	x	{1,2}=1
	{2,4}=1	{1,4}=1	{1,4}=1	{**1**-2}	x	{1,4}=1
3	{3,2}=1	{2,3}=1	{2,3}=1	{1-**2**-4}	x	{2,3}=1
	{3,5}=1	{2,4}=1	**{2,4}=1**	{1-**2**-3-**4**}	O	-
4	{4,1}=1	{3,5}=1	{3,5}=1	{1-2-**3**-4}	x	{3,5}=1
	{4,2}=1	{4,7}=1	{4,7}=1	{1-2-3-**4**-5}	x	{4,7}=1
5	{5,3}=1	{5,7}=1	**{5,7}=1**	{1-2-3-4-**5**-**7**}	O	-
	{5,7}=1	{6,8}=1	{6,8}=1	{1-2-3-4-5-7}	x	{6,8}=1
6	{6,8}=1	{7,8}=1	{7,8}=1	{1-2-3-4-5-**7**}, {6-**8**}	x	{7,8}=1
7	{7,4}=1	{9,6}=2	{9,6}=2	{1-2-3-4-5-**6**-7-8}	x	{9,6}=2
	{7,5}=1			{1-2-3-4-5-6-7-8-9}		
	{7,8}=1					
8	{8,6}=1					
	{8,7}=1					
9	{9,6}=2					
$\|v\|=9$		$\|e_{mst}\|=10$		$\|F\|=1$		$\|e_{mst}\|=8$

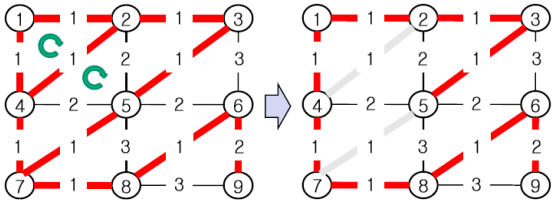
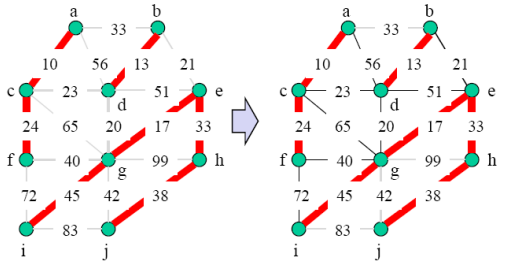


표 4. G_5 그래프의 일반화된 Borůvka 알고리즘 적용
Table 4. Generalized Borůvka Algorithm Applying to Graph G_5

(a) 1st Stage

V	MWE	중복 제거		MSF	사이클	e_{mst}
a	{a,c}=10					
b	{b,d}=13	{a,c}=10	{a,c}=10	-	x	{a,c}=10
c	{c,a}=10	{b,d}=13	{b,d}=13	{a-c}	x	{b,d}=1
d	{d,b}=13	{e,g}=17	{e,g}=17	{a-c}, {b-d}	x	3
e	{e,g}=17	{f,c}=24	{f,c}=24	{a-c}, {b-d}, {e-g}	x	{e,g}=17
f	{f,c}=24	{h,e}=33	{h,e}=33	[a-c-f}, {b-d}, {e-g}	x	{f,c}=24
g	{g,e}=17	{i,g}=45	{j,h}=38	{a-c-f}, {b-d}, {e-g-h}	x	{h,e}=33
h	{h,e}=33	{j,h}=38	{i,g}=45	{a-c-f}, {b-d}, {e-g-h-j}	x	{j,h}=38
i	{i,g}=45			{a-c-f}, {b-d}, {e-g-h-i-j}		{i,g}=45
j	{j,h}=38					
$\|v\|=10$		$\|e_{mst}\|=7$		$\|F\|=3$		$\|e_{mst}\|=7$



(b) 2nd Stage

e_F		MSF	e_{mst}
{a,b}=33	{d,g}=20	{a-c-f}, {b-**d**}, {e-**g**-h-i-j}	{d,g}=20
{a,d}=56	{b,e}=21	-	-
{c,d}=23	{c,d}=23	{a-**c**-f}, {b-**d**-e-g-h-i-j}	{c,d}=23
{c,g}=65	{a,b}=33	{a-b-c-d-e-f-g-h-i-j}	
{f,g}=40	{f,g}=40		
{f,i}=72	{d,e}=51		
{b,e}=21	{a,d}=56		
{d,e}=51	{c,g}=65		
{d,g}=20	{f,i}=72		
-		$\|F\|=1$	$\|v\|=10$, $\|e_{mst}\|=9$

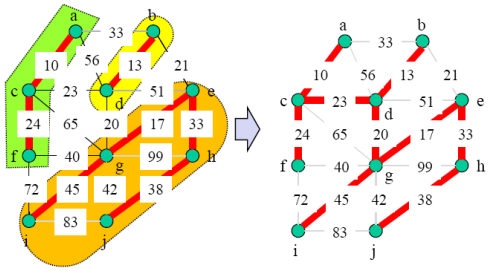
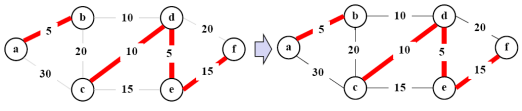


표 5. G_6 그래프의 일반화된 Borůvka 알고리즘 적용
Table 5. Generalized Borůvka Algorithm Applying to Graph G_6

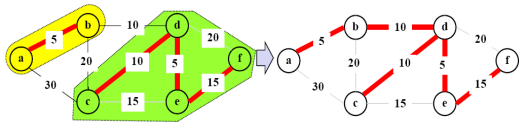
(a) 1st Stage

V	MWE	중복 제거		MSF	사이클	e_{mst}
a	{a,b}=5					
b	{b,a}=5	{a,b}=5	{a,b}=5	-	x	{a,b}=5
c	{c,d}=10	{c,d}=10	{d,e}=5	{a-b}	x	{d,e}=5
d	{d,e}=5	{d,e}=5	{c,d}=10	{a-b}, {**d**-e}	x	{c,d}=10
e	{e,d}=5	{f,e}=15	{f,e}=15	{a-b}, {**d**-**e**}	x	{f,e}=15
f	{f,e}=15			{a-b}, {c-d-e-f}		
$\|v\|=6$		$\|e_{mst}\|=4$		$\|F\|=2$		$\|e_{mst}\|=4$



(b) 2nd Stage

e_F		MSF	e_{mst}
{a,c}=30	{b,d}=10	{a-**b**}, {c-**d**-e-f}	{b,d}=10
{b,c}=20	{b,c}=20	{a-b-c-d-e-f}	
{b,d}=10	{a,c}=30		
-		$\|F\|=1$	$\|v\|=6$, $\|e_{mst}\|=5$

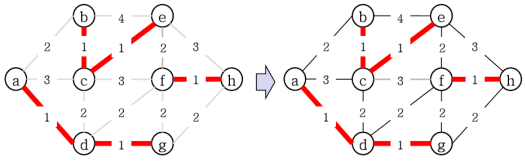


G_4 그래프는 1st Stage만을 수행하여 MST를 얻었다. 반면에 G_5, G_6과 G_7 그래프는 2nd Stage를 수행하여 MST를 얻었다.

표 6. G_7 그래프의 일반화된 Borůvka 알고리즘 적용
Table 6. Generalized Borůvka Algorithm Applying to Graph G_7

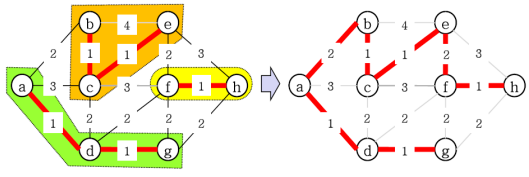
(a) 1st Stage

V	MWE	중복 제거		MSF	사이클	e_{mst}
a	{a,d}=1					
b	{b,c}=1					
c	{c,b}=1	{a,d}=1	{a,d}=1	-	x	{a,d}=1
	{c,e}=1	{b,c}=1	{b,c}=1	[a-d}	x	{b,c}=1
d	{d,a}=1	{c,e}=1	{c,e}=1	{a-d}, {b-**c**}	x	{c,e}=1
	{d,g}=1	{d,g}=1	{d,g}=1	{a-**d**}, {b-c-e}	x	{d,g}=1
e	{e,c}=1	{f,h}=1	{f,h}=1	{a-d-g}, {b-c-e}	x	{f,h}=1
f	{f,h}=1			{a-d-g}, {b-c-e}, {f-h}		
g	{g,d}=1					
h	{h,f}=1					
$\|v\|=8$		$\|e_{mst}\|=5$		$\|F\|=3$		$\|e_{mst}\|=5$



(b) 2nd Stage

e_F		MSF	e_{mst}
{a,b}=2	{a,b}=2	{**a**-d-g}, {**b**-c-e}, {f-h}	{a,b}=2
{a,c}=3	{c,d}=2		-
{c,d}=2	{d,f}=2	{a-b-c-**d**-e-g}, {**f**-h}	{d,f}=2
{c,f}=3	{e,f}=2	{a-b-c-d-e-f-g-h}	
{d,f}=2	{f,g}=2		
{e,f}=2	{g,h}=2		
{e,h}=3	{a,c}=3		
{f,g}=2	{c,f}=3		
{g,h}=2	{e,h}=3		
-		$\|F\|=1$	$\|v\|=8$, $\|e_{mst}\|=7$



본 논문에 적용된 7개 그래프에 대한 알고리즘 수행 횟수를 Kruskal 알고리즘과 비교한 결과는 표 7에 제시하였다. 참고로 Kruskal 알고리즘은 $|e|$ 회 수행하는 경우와 $|e_{mst}| = |v|-1$ 에서 종료하는 기준을 적용하였다. 표에서 $|v|-1$ 의 경우 동일한 가중치 간선이 다수 존재하여 오름차순으로 정렬할 경우 동일한 값은 순서와 상관없기 때문에 알고리즘 수행 횟수는 범위로 결정된다.

표 7. MST 알고리즘 수행횟수 비교
Table 7. MST Algorithm Performance Comparison

그래프	$\|v\|$	$\|e\|$	MST 가중치 합	일반화된 Borůvka 알고리즘			Kruskal 알고리즘	
				1st Stage	2nd Stage	계	$\|e\|$ 회 수행	$\|v\|-1$ 회 수행
G_1	6	12	21	4	1	**5**	12	6
G_2	12	23	24	10	2	**12**	23	[15,16]
G_3	16	33	21	19	-	**19**	33	[22,28]
G_4	9	16	9	10	-	**10**	16	[10,13]
G_5	10	19	224	7	2	**9**	19	13
G_6	6	9	45	4	1	**5**	9	[5,6]
G_7	8	15	9	5	2	**7**	15	[7,11]

비록 Kruskal 알고리즘을 $|e_{mst}| = |v|-1$ 에서 종료하는 기준을 적용하더라도 7개 데이터 모두에서 일반화된 Borůvka 알고리즘의 수행횟수가 보다 적음을 알 수 있다. 결국, 제안된 일반화된 Borůvka 알고리즘을 일반화된 MST 알고리즘으로 적용할 수 있을 것이다.

V. 결 론

본 논문은 MST를 찾는 알고리즘으로 처음으로 제안된 Borůvka 알고리즘을 고찰하고 일반적으로 적용할 수 있는 일반화된 Borůvka 알고리즘을 제안하였다. 이 알고리즘은 기존의 알고리즘[15-16] 등에 확장하여 적용할 수 있을 것이다.

Borůvka 알고리즘은 모든 간선들이 상이한 가중치를 갖는 그래프를 가정하고 제안된 알고리즘이다. 그러나 임의의 하나의 정점에 부속된 간선들 중에서도 최소 가중치 간선이 다수 존재하는 경우도 발생한다. 하나의 정점에 부속된 동일한 값을 가진 최소 가중치 간선을 모두 선택하였을 경우, 사이클이 발생할 수 있다. 또한, 하나의 연결된 신장트리를 얻지 못하고 다수의 MSF를 얻는다. 따라서 사이클을 제거하는 작업과 이들 MSF를 자동적으로 하나의 신장트리로 해결할 수 있는 방법은 제시되지 않고 있다. 본 논문은 이러한 문제점을 Kruskal 방법을 도입하여 해결하였다. 해결 방법은 1st Stage에서 얻은 MWE인 e_{mst}에 대해 Kruskal 알고리즘을 적용하고, 만약, MSF가 2개 이상 발생하면 2nd Stage에서 MSF간에

연결된 간선들인 e_F를 오름차순으로 정렬시켜 MWE를 선택하는 방법을 적용하였다.

제안된 알고리즘은 하나의 정점에 부속된 간선들의 가중치가 상이하거나 동일한 경우 모두 Borůvka 알고리즘을 적용하여 MST를 쉽고, 빠르게 구할 수 있었다. 따라서 일반화된 Borůvka 알고리즘도 MST를 구하기 위해 현재 널리 사용되는 Prim 또는 Kruskal 알고리즘과 더불어 일반적으로 활용할 수 있을 것이다.

참 고 문 헌

[[1] Wikipedia, “Minimum Spanning Tree,” http://en.wikipedia.org/wiki/Minimum_spanning_tree, Wikipedia Foundation Inc., 2010.

[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.

[7] Wikipedia, Reverse-Delete Algorithm," http://en.wikipedia.org/wiki/Reverse-delete_algorithm, Wikipedia Foundation Inc., 2010.

[7] J. Erickson, "CS 473G - Graduate Algorithms," Dept. of Computer Science, University of Illinois, 2005.

[8] D. Pe'er and A. Wigderson, "On Minimum Spanning Trees," Hebrew University, MSc Thesis, 1998.

[9] Wikipedia, Borůvka Algorithm," "http://en.wikipedia.org/wiki/Borůvka_algorithm,Wikipedia Foundation Inc., 2010.

[10] J. Eisner, "State-of-the-Art Algorithms for Minimum Spanning Trees: A Tutorial Discussion," University of Pensylvania, 1997.

[11] S. Pettie and V. Ramachndran, "An Optimal Minimum Spanning Tree Algorithm," *Journal of the ACM*, vol. 49, no. 1, pp. 16-34, 2002.

[12] Wikipedia, "Reverse-Delete Algorithm," http://en.wikipedia. org/wiki/Reverse_Delete_algorithm, 2008.

[13] J. Kleinberg and E. Tardos, "*Algorithm Design*," New York: Pearson Education, Inc., 2006.

[14] C. Peiper, CS 400 - Data Structures for Non CS-Majors," http://www.cs.uiuc.edu/class/fa05/cs400 /_labs/Lab12/suuri/, 2005.

[15] Yong-Jin Lee, Dong-Woo Lee, "Minimum Cost Spanning Tree Problem in Wireless Sensor Network and Heuristic Algorithm," Journal of Advanced Information Technology and Convergence, pp. 275~282, vol. 7, no. 4, 2009. 8.

[16] Dongyoung Shin, Joonseok Park, "A Performance Improvement for Tree Search using Dynamic Load Balancing between CPU and GPGPU," Journal of Advanced Information Technology and Convergence, vol. 10, no. 2, pp. 132-140, 2012. 2.

저자 소개

최 명 복(종신회원)

- 1994년 : 아주대학교 컴퓨터공학과(석사)
- 2001년 : 아주대학교 컴퓨터공학과(박사)
- 1997년~현재 : 강릉원주대학교 멀티미디어공학과 교수
- 2004년~현재 : 한국인터넷방송통신학회 이사

<주관심분야 : 지능형 정보검색, 소프트웨어 공학, 알고리즘>

이 상 운(정회원)

- 1995년~1997년 : 경상대학교 컴퓨터과학과(석사)
- 1998년~2001년 : 경상대학교 컴퓨터과학과(박사)
- 2007년~현재 : 강릉원주대학교 과학기술대학 멀티미디어공학과 부교수

<주관심분야 : 소프트웨어 공학, 신뢰성, 신경망, 뉴로-퍼지, 그래프 알고리즘>