DOI: 10.3745/KIPSTA.2010.17A.3.159

하이브리드 최소신장트리 알고리즘

이 상 운†

요 약

본 논문에서는 여러 간선들이 동일한 가중치를 갖고 있는 그래프에서 최소신장트리 (Minimum Spanning Tree, MST)를 얻기 위해 Borůvka, Prim과 Kruskal MST 알고리즘을 실제 그래프에 적용한 결과 Borůvka와 Kruskal MST 알고리즘은 MST를 얻었지만 Prim MST 알고리즘은 MST를 얻는데 실패함을 보였다. 또한, Borůvka의 2nd Stage에서 Inter-MSF MWE를 선택하는 알고리즘이 복잡함을 알 수 있었다. Borůvka의 1st Stage는 최소한의 간선들로 최소신장 포레스트 (Minimum Spanning Forest, MSF)를 얻는 장점을 갖고 있으며, Kruskal MST 알고리즘은 모든 간선들을 대상으로 하지만 항상 MST를 얻는 장점을 갖고 있다. 따라서 본 논문은 Borůvka의 1st Stage와 Kruskal MST 알고리즘의 장점을 결합한 하이브리드 MST 알고리즘을 제안하였다. 하이브리드 MST 알고리즘을 추가적으로 6개의 그래프에 적용한 결과 Kruskal MST 알고리즘과 동일하게 항상 MST를 얻음을 검증하였다. 또한, 알고리즘 수행속도와 메모리 용량 측면에서 비교한 결과 하이브리드 MST 알고리즘이 가장 좋은 성능을 보였다. 따라서 제안된 알고리즘을 일반화된 MST 알고리즘으로 채택이 가능할 것이다.

키워드 : 최소신장트리, 무방향성, 상이한 가중치, 동일 가중치, 사이클

Hybrid Minimum Spanning Tree Algorithm

Sang-Un Lee†

ABSTRACT

In this paper, to obtain the Minimum Spanning Tree (MST) from the graph with several nodes having the same weight, I applied both Borůvka and Kruskal MST algorithms. The result came out to such a way that Kruskal MST algorithm succeeded to obtain MST, but not did the Prim MST algorithm. It is also found that an algorithm that chooses Inter-MSF MWE in the 2nd stage of Borůvka is quite complicating. The 1st stage of Borůvka has an advantage of obtaining Minimum Spanning Forest (MSF) with the least number of the edges, and on the other hand, Kruskal MST algorithm has an advantage of always obtaining MST though it deals with all the edges. Therefore, this paper suggests an Hybrid MST algorithm which consists of the merits of both Borůvka's 1st stage and Kruskal MST algorithm. When applied additionally to 6 graphs, Hybrid MST algorithm has a same effect as that of Kruskal MST algorithm. Also, comparing the algorithm performance speed and capacity, Hybrid MST algorithm has shown the greatest performance Therefore, the suggested algorithm can be used as the generalized MST algorithm.

Keywords : Minimum Spanning Tree, Undirected, Distinct Weights, Same Weights, Cycle

1. 서 론

그래프 $G=(V,E)$는 정점들 (Vertices)과 간선들(Edges)로 구성되어 있으며, 정점들이 연결되어 있고 (Connected), 간선들은 방향성 (Directed)과 무방향성 (Undirected)을 갖고 있으며 또한 가중치가 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 않는 상태에서 그래프의 모든 정점들을 간선으로 연결한 트리이다. 하나의 그래프에서는 다수의 ST가 존재할 수 있다. 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] 본 논문은 MST 알고리즘에 초점을 맞춘다.

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]

그래프에서 간선의 가중치 (길이, 또는 비용)는 모두 다른 가중치를 갖는 경우, 모두 동일한 가중치를 갖는 경우와 일

† 정 회 원 : 강릉원주대학교 과학기술대학 멀티미디어공학과 부교수
논문접수 : 2010년 3월 10일
심사완료 : 2010년 5월 4일

부분만이 동일한 가중치를 갖는 경우로 분류될 수 있다. 모든 간선들이 동일한 가중치를 갖는 경우에는 하나의 정점에 연결된 간선들 중에서 하나씩만 선택하면 MST를 얻을 수 있기 때문에 굳이 MST 알고리즘을 적용할 필요가 없다. 또한, 모든 간선들의 가중치가 서로 상이하다면 그래프의 MST는 유일하게 얻을 수 있다.[6] 그러나 여러 간선들이 동일한 가중치를 갖는 경우에 대해 기존의 MST 알고리즘을 적용하는데 문제가 없는지 살펴보자.

Erickson[6]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. 또한, Chen[7]은 "Prim과 Kruskal MST 알고리즘은 항상 작동한다"고 하였다. 이와 같은 연구 결과를 적용하면 기존의 MST 알고리즘으로 동일한 가중치를 갖는 간선들이 다수 존재하는 그래프에도 MST를 얻어야만 한다. 그러나 일부분만이 동일한 가중치를 갖는 경우에 대해 Borůvka와 Prim MST 알고리즘은 사이클 발생을 제거할 수 없어 MST를 얻는데 실패하는 경우가 발생한다. 지금까지는 이러한 경우에 대해 기존의 MST 알고리즘의 적용 문제점을 거론하거나 개선된 알고리즘을 제안하고 있지 않다.

Borůvka MST 알고리즘은 1st Stage에서는 각 정점에 연결된 MWE만을 선택하므로 Candidate Edge수를 줄이는 장점이 있는 반면, 1st Stage에서 얻은 MSF (Minimum Spanning Forest)을 연결하는 MWE를 찾는 2nd Stage에서는 MSF 간에 간선들을 재구성해야 하며, 특정 개수만을 선택해야 하는 알고리즘의 복잡성이 있다. Kruskal MST 알고리즘은 모든 간선들을 대상으로 하기 때문에 불필요한 Candidate Edge수를 줄일 필요가 있다. 따라서 본 논문에서는 Borůvka의 1st Stage와 Kruskal MST 알고리즘의 장점만을 활용한 하이브리드 MST 알고리즘을 제안한다.

2장에서는 기존에 제안된 대표적인 MST 알고리즘인 Borůvka Prim과 Kruskal MST 알고리즘을 간선들 가중치가 모두 다른 경우와 일부분만이 동일한 경우에 대해 실제 그래프에 적용하고 문제점을 고찰해 본다. 3장에서는 먼저, 간선들 가중치가 일부분만이 동일한 경우에도 적용할 수 있는 일반화된 MST 알고리즘으로 하이브리드 MST Algorithm을 제안한다. 4장에서는 10개의 그래프에 실제 적용하여 제안된 알고리즘의 적용성과 성능을 검증해본다.

2. 관련 연구와 연구 배경

2.1 최소신장트리 알고리즘 고찰

그래프 (Graph) $G=(V,E)$에서 Edge는 무방향성이므로 $\{x,y\}$로 표기한다. 왜냐하면 (x,y) 표기는 순서쌍으로 방향성을 가지고 있음을 의미하기 때문이다.

2.1.1 Borůvka MST 알고리즘

Borůvka MST 알고리즘은 첫 번째 단계에서 각 정점에 연결된 간선들 중 최소 가중치를 갖고 있는 간선 (Minimum Weight Edge, MWE)을 선택하여 사이클이 발생하지 않는 최소 신장 포레스트 (Minimum Spanning Forest, MSF)를 구성하고, 다음 단계에서 MSF들을 연결하기 위해 사이클이 발생하지 않는 MWE를 "MSF 수 - 1"개를 찾는 방법이다. Borůvka MST 알고리즘은 (그림 1)과 같다.

≪ Borůvka's MST Algorithm ≫

$G=(V,E)$
$|v|$: Number of Vertices in Graph $G=(V,E)$
$|e|$: Number of Edges in Graph $G=(V,E)$
MWE : Minimum-Weight Edge $\{x,y\}$
MSF Vertices : Minimal Spanning Forest (MSF) Vertices
MSF Edges : Minimal Spanning Forest (MSF) Edges
$|e_c|$: Number of Candidate Edges
$|e_{1st}|$: Number of Minimum Edges in 1st Stage MSF Edges
$|e_{2nd}|$: Number of Minimum Edges in 2nd Stage MSF Edges
$|f|$: Number of MSF

/* **1st Stage** : MSF Vertices 생성
$v \times v$ **정방행렬 작성**
For 1 to $|v|$ do
  정방행렬의 각 행 Vertex에서 MWE 선택**(단, 동일한 값이 다수 존재시 모두 선택)**, Candidate Edges에 저장
END
FOR 1 to $|e_c|$ do
  IF 선택된 MWE의 $\{x,y\} \in$ MSF Vertices에 있는 하나의 Forest의 원소 THEN Skip /* **사이클 발생**
  ELSE IF 선택된 MWE의 $\{x,y\} \in$ MSF Vertices에 있는 두 Forest의 원소 or 선택된 MWE 의 $\{x,y\} \notin$ MSF Vertices THEN
    선택된 MWE를 MSF Edges (e_{1st})에 추가
    선택된 MWE의 $\{x,y\}$를 MSF Vertices의 해당 Forest에 연결 또는 생성
  ENDIF
  IF $(|e_{1st}| = |v|-1)$ and $(|f|=1)$ THEN 알고리즘 종료
  ELSE LOOP
END
IF $|f|=1$ THEN 알고리즘 종료
ELSE IF $|f| \geq 2$ THEN 2nd Stage 수행
ENDIF

/* **2nd Stage** : MSF를 상호 연결시키는 MWE를 찾음
하나의 MSF를 행으로, 다른 MSF를 열로 하는 Edges 행렬 작성
Inter-MSF MWE(e_f) 선택 **(단, 동일한 값이 다수 존재시 하나씩만 선택)**
IF $|e_f| > |f|-1$
  Inter-MSF MWE를 오름차순으로 정렬
  FOR 1 to $|f|-1$ do
    선택된 MWE를 MSF Edges (e_{2nd})에 추가
    선택된 MWE의 $\{x,y\}$ MSF 연결
  END
ELSE IF 선택된 MWE를 MSF Edges (e_{2nd})에 추가, 선택된 MWE의 $\{x,y\}$ MSF 연결
END

(그림 1) Borůvka MST 알고리즘

2.1.2 Prim MST 알고리즘

Prim MST 알고리즘은 임의의 정점을 선택하고, 이에 연결된 간선들 중에서 MWE를 선택한다. 다음으로 새로 선택된 정점에 연결된 간선들의 가중치와 기존에 선택된 정점에서 선택되지 않은 간선의 가중치들 중에서 MWE를 선택하는 방법이다. (단, 이 과정에서 사이클이 발생하는 간선은 무시한다.) 이 방법을 모든 정점들이 선택될 때까지 수행한다. Prim MST 알고리즘은 (그림 2)에 제시되어 있다.

≪Prim's MST Algorithm ≫

MST는 MSF Vertices와 MSF Edges로 구성
Candidate Edges : ST를 구성하기 위해 연결에 이용될 후보 Edge들
MWE : Minimum-Weight Edge $\{x,y\}$
MSF Vertices : Minimal Spanning Forest (MSF) Vertices
MSF Edges : Minimal Spanning Forest (MSF) Edges
e_{mst} : MSF Edges
$|e_{mst}|$: Number of MSF Edges $=|v|-1$
Candidate Edges $= e_r+e_a-e_d$
e_a : MSF Vertices에 새로 추가되는 Vertex에 부속된 Edges
e_r : Candidate Edges에서 삭제되지 않고 남아 있는 Edges
e_d : Candidate Edges에서 MSF Edges로 이미 선택되었거나 사이클 발생으로 삭제되는 Edges

$|v|\times|v|$ 정방행렬 작성
Step 1. V에서 임의의 Vertex 선택, MSF Vertices에 저장, V에서 선택된 Vertex 삭제
Step 2. For $|v|$ *to* 0 do
  MSF Edges에 새로 추가된 Vertex (y)에 연결된 모든 Edges들을 Candidate Edges에 추가(e_a)
  IF Candidate Edges (e_r+e_a) 중 $y \in MSF\,Vertices$ THEN Candidate Edges에서 삭제(e_d) **/* 사이클 발생 또는 중복 선택**
  END
  Candidate Edges에 남아 있는 Edges들$(e_r+e_a-e_d)$ 중 MWE 선택**(단, 동일한 값이 다수 존재시 모두 선택)**
  선택된 MWE를 MSF Edges에 추가, Candidate Edges에서 삭제
  선택된 MWE의 $\{x,y\}$에서 y를 MSF Vertices에 저장, V에서 삭제
 END

(그림 2) Prim MST 알고리즘

2.1.3 Kruskal MST 알고리즘

Kruskal MST 알고리즘은 그래프의 모든 간선들을 대상으로 오름차순으로 정렬시키고, 첫 번째 MWE부터 시작하여 사이클이 발생하지 않는 한 간선들 (e_{mst})을 $|v|-1$이 될 때까지 선택하는 방법이다. Kruskal MST 알고리즘은 (그림 3)과 같다.

≪Kruskal's MST Algorithm≫

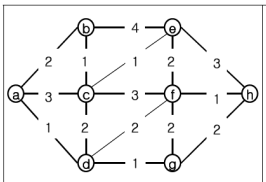
MWE : Minimum-Weight Edge $\{x,y\}$
MSF Vertices : Minimal Spanning Forest (MSF) Vertices
MSF Edges : Minimal Spanning Forest (MSF) Edges
$|f|$: Number of MSF
e_{mst} : MSF Edges
$|e_{mst}|$: Number of MSF Edges $=|v|-1$

$|v|\times|v|$ 정방행렬 작성
FOR 1 to 2$|e|$ *** 상삼각행렬의 모든 Edge들 선택**
  정방행렬의 상삼각행렬에 대한 모든 Edges를 Candidate Edges에 저장
END
Candidate Edges를 오름차순으로 정렬
For $|e|$ = 1 to $|V|-1$
  Candidate Edges에서 MWE 선택, Candidate Edges에서 삭제
  IF 선택된 MWE $\{x,y\} \in$ MSF의 하나의 Forest THEN Skip /* 사이클 발생
  ELSE IF 선택된 MWE $\{x,y\} \in$ MSF의 두 Forest 원소 or $\{x,y\} \notin$ MSF THEN
    선택된 MWE를 MSF Edges에 추가
    선택된 MWE의 x,y Vertex를 MSF의 해당 Forest에 연결 또는 새로운 Forest 생성
  ENDIF
END

(그림 3) Kruskal MST 알고리즘

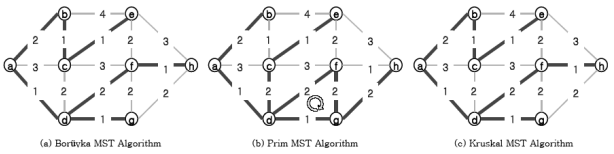
2.2 알고리즘 적용 문제점과 연구 배경

모든 간선들이 동일한 가중치를 가진다면 각 정점에서 하나씩 $|v|-1$개의 다른 간선들을 선택해도 모두 동일한 결과



	a	b	c	d	e	f	g	h
a		2	3	1				
b	2		1		4			
c	3	1		2	1	3		
d	1		2			2	1	
e		4	1			2		3
f			3	2	2		2	1
g				1		2		2
h					3	1	2	

(그림 4) G_1그래프



(그림 5) G_1 그래프의 MST 알고리즘 적용 결과

를 얻는 MST를 얻을 수 있다. 또한, 모든 간선들의 가중치가 상이한 경우 반드시 1개의 MST를 얻는데 문제가 발생하지 않는다. 그러나 다수의 간선들이 동일한 가중치를 갖는 경우에도 MST 알고리즘을 적용할 수 있다면 기존에 제안된 MST 알고리즘들은 모두 일반화된 알고리즘으로 적용될 수 있을 것이다. 본 절에서는 이 경우에 대해 (그림 4)의 G_1 그래프를 대상으로 적용에 문제점이 없는지 고찰해 본다. G_1 그래프는 Chen[7]에서 인용되었다.

대표적인 MST 알고리즘인 Borůvka, Prim과 Kruskal MST 알고리즘을 (그림 4)의 G_1 그래프에 적용하여 MST를 구하는 과정은 <표 1>에, ST를 구한 결과는 (그림 5)에 제시되어 있다.

Borůvka MST 알고리즘은 $|e_{mst}|=|v|-1=7$개를 찾기 위해 1st Stage에서 $|e_c|=5$개를 모두 비교하였지만 MST를 얻지 못하고 $a-d-g$, $b-c-e$와 $f-h$의 MSF를 얻었다. 이들 3개 MSF간을 연결하는 MWE (Inter-MSF MWE)를 찾기 위해 2nd Stage에서 $|e_c|=9$개를 모두 비교하였으며, Inter-MSF MWE 6개중 2개를 선택하여 $|e_{mst}|=|e_{1st}|+|e_{2nd}|=7$, $\Sigma w(e)=9$을 얻어 MST를 얻을 수 있었다.

Prim MST 알고리즘은 $|v|-1=7$개의 정점 연결을 수행하는 과정에서 새로 추가되거나, 기존에 후보 간선 들(Candidate Edges)에 남아 있는 $|e_c|=33$개에 대해 모두 비교 ($|e_s|=33$)한 결과 $|e_{mst}|=8$, $\Sigma w(e)=13$을 얻었으며, 사이클이 발생하여 MST를 얻는데 실패하였다. 반면에 Kruskal MST 알고리즘은 그래프의 모든 간선들을 대상으로 후보 간선들로 하기 때문에 $|e_c|=15$개에 대해 오름차순으로 정렬하여 순차적으로 비교한 결과 $|e_s|=8$번째인 $\{d,f\}=2$에서 $|e_{mst}|=7$, $\Sigma w(e)=9$를 얻고 알고리즘이 종료되었다. 또한 사이클이 발생하지 않아 MST를 얻는데 성공하였다. Chen[7]은 "Prim과 Kruskal MST 알고리즘은 항상 MST를 얻는다"고 하였다. 그러나 본 사례에서 알 수 있듯이, Prim MST 알고리즘은 항상 MST를 얻지 못하는 경우도 발생할 수 있다.

MST 알고리즘들이 요구하는 후보 간선들 수 ($|e_c|$)와 후

〈표 1〉 G_1 그래프의 MST 알고리즘 적용

(a) Borůvka MST 알고리즘

1st Stage				2nd Stage	
Candidate Edges	Cycle	MSF Vertices	MSF Edges	Inter-MSF MWE	MSF Edges
{a,d}=1	-	-	{a,d}=1	b c e | f h a **2** 3 - | - - d - **2** - | **2** - g - - - | **2 2** f h b - - c 3 - e **2** 3	{a,b}=2 {d,f}=2
{b,c}=1	x	a-d	{b,c}=1		
{c,b}=1	O	a-d, b-c	-	a-d-g와 b-c-e MSF : {a,b}=2, {d,c}=2, 중 {a,b}=2 선택	
{c,e}=1	x	a-d, b-c	{c,e}=1		
{d,a}=1	O	a-d, b-c-e	-	a-d-g와 f-h MSF : {d,f}=2, {g,f}=2, {g,h}=2 중 {d,f}=2 선택	
{d,g}=1	x	a-d, b-c-e	{d,g}=1		
{e,c}=1	O	a-d-g, b-c-e	-	b-c-e와 f-h MSF : {e,f}=2는 무시	
{f,h}=1	x	a-d-g, b-c-e	{f,h}=1		
{g,d}=1	O	a-d-g, b-c-e, f-h	-		
{h,f}=1	O	a-d-g, b-c-e, f-h	-		
$\|e_c\|=10$	$\|e_s\|=$		$\|e_{1st}\|=5$	$\|f\|=3$, $\|e_c\|=9$, $\|e_s\|=9$	$\|e_{2nd}\|=$

(b) Prim MST 알고리즘

V	MSF Vertices	Candidate Edges				MSF Edges
		e_a	e_r	Cycle (e_d)	$e_r+e_a-e_d$	
{a,b,c,d,e,f,g,h}	{ ∅ }	{ ∅ }	{ ∅ }	{ ∅ }	{ ∅ }	{ ∅ }
{b,c,d,e,f,g,h}	{a}	{a,b}=2,{a,c}=3 {a,d}=1	{ ∅ }	{ ∅ }	{a,b}=2,{a,c}=3 **{a,d}=1**	{a,d}=1
{b,c,e,f,g,h}	{a,d}	**{d,a}=1**,{d,c}=2 {d,f}=2,{d,g}=1	{a,b}=2,{a,c}=3	{d,a}=1	{a,b}=2,{a,c}=3 {d,c}=2,{d,f}=2 **{d,g}=1**	{d,g}=1
{b,c,e,f,h}	{a,d,g}	**{g,d}=1**,{g,f}=2 {g,h}=2	{a,b}=2,{a,c}=3 {d,c}=2,{d,f}=2	{g,d}=1	**{a,b}=2**,{a,c}=3 **{d,c}=2,{d,f}=2** **{g,f}=2,{g,h}=2**	{a,b}=2 {d,c}=2 {d,f}=2 {g,f}=2 {g,h}=2
{e}	{a,d,g,b,c,f,h}	**{b,a}=2,{b,c}=1** {b,e}=4,**{c,a}=3** **{c,b}=1,{c,d}=2** {c,e}=1,**{c,f}=3** **{f,c}=3,{f,d}=2** {f,e}=2,**{f,g}=2** **{f,h}=1**,{h,e}=3 **{h,f}=1,{h,g}=2**	**{a,c}=3**	{b,a}=2,{b,c}=1 {c,a}=3,{c,b}=1 {c,d}=2,{c,f}=3 {f,c}=3,{f,d}=2 {f,g}=2,{f,h}=1 {h,f}=1,{h,g}=2 {a,c}=3	{b,e}=4,**{c,e}=1** {f,e}=2,{h,e}=3	{c,e}=1
{ ∅ }	{a,d,g,b,c,f,he}	-	-	-	-	(종료)
$\|v\|=8$		$\|e_c\|=33$, $\|e_s\|=33$			$\|e_{mst}\|=8$	

(c) Kruskal MST 알고리즘

Candidate Edges			MSF Vertices	MSF Edges
All Edges	Ascent Sorted Edges	Cycle Check		
{a,b}=2 {a,c}=3 {a,d}=1 {b,c}=1 {b,e}=4 {c,d}=2 {c,e}=1 {c,f}=3 {d,f}=2 {d,g}=1 {e,f}=2 {e,h}=3 {f,g}=2 {g,h}=2	{a,d}=1 {b,c}=1 {c,e}=1 {d,g}=1 {f,h}=1 {a,b}=2 {c,d}=2 {d,f}=2 {e,f}=2 {f,g}=2 {g,h}=2 {a,c}=3 {c,f}=3 {e,h}=3 {b,e}=4	x x x x x x O x - - - - - - -	- a-d a-d, b-**c** a-**d**, b-c-e a-d-g, b-c-e **a**-d-g, **b**-c-e, f-h a-b-**c**-**d**-e-g, f-h a-b-c-**d**-e-g, **f**-h a-b-c-d-e-f-g-h - - - - - -	{a,d}=1 {b,c}=1 {c,e}=1 {d,g}=1 {f,h}=1 {a,b}=2 - {d,f}=2 (종료)
	$\|e_c\|=15$	$\|e_s\|=8$		$\|e_{mst}\|=7$

보 간선들 중 비교되는 Edges 수 ($|e_s|$)는 <표 2>에 제시되어 있다.

Borůvka의 2nd Stage는 사이클이 발생하지 않는 Inter-MSF MWE를 선택하기 위한 알고리즘이 복잡하다. 또한, Prim MST 알고리즘은 MST를 얻을 수 없는 경우가 발생한다. 그러나 Borůvka의 1st Stage와 Kruskal MST 알고리즘은 적용에 문제가 없다. 따라서 Kruskal MST 알고리즘 적용의 단점인 후보 간선들 수를 줄일 수 있는 방법으로 Borůvka MST 알고리즘의 1st Stage에서 MWE를 선택하는 방법을 적용하면 Kruskal MST 알고리즘보다 수행 속도를 향상시킬 수 있을 것이다. 3장에서는 이러한 개념의 하이브리드 MST 알고리즘을 제안한다.

〈표 2〉 MST 알고리즘의 $|e_c|$와 $|e_s|$

Graph	$\|v\|$	$\|e\|$	$\|e_c\|/\|e_s\|$				
			Borůvka MST 알고리즘			Prim MST 알고리즘	Kruskal MST 알고리즘
			1st Stage	2nd Stage	Total		
G_1	8	15	10/10	9/9	19/19	33/33	15/8

3. 하이브리드 MST 알고리즘

3.1 알고리즘

Kruskal MST 알고리즘은 MST를 얻기 위한 후보 간선들을 그래프의 모든 간선들($|e|$)을 선택하지만 사이클은 발생하지 않는 장점을 갖고 있다. 반면에 Borůvka MST 알고리즘은 1st Stage에서 각 정점에 연결된 MWE만을 선택하여 MSF를 형성함으로서 Candidate Edges 수를 줄일 수 있는 장점을 갖고 있다. 그러나 이 알고리즘은 2nd Stage에서 MSF 간에 연결될 MWE를 찾는데 다수의 후보 간선들을 재구성하는 불편함이 있으며, 사이클이 발생하여 MST를 얻지 못하는 단점을 갖고 있다. 본 절에서는 Kruskal과 Borůvka MST 알고리즘에서 장점만을 선택하여 최적의 MST를 찾는 일반화된 알고리즘을 제안한다. 제안된 알고리즘의 개략적인 내용은 다음과 같다.

(1) 각 정점에 연결된 간선들 중 MSF가 1개가 될 때까지 순차적으로 1st, 2nd와 3rd MiWE들을 선택하여 Candidate Edges에 저장한다.(단, 동일한 가중치를 가진 간선들이 다수 존재시 모두 선택한다.) [Borůvka MST 알고리즘의 변형 적용]
(2) 1st, 2nd와 3rd Candidate Edges를 오름차순으로 정렬한다. [Kruskal MST 알고리즘 적용]
(3) Candidate Edges의 1st, 2nd와 3rd 순으로 사이클 검증을 거치면서 MSF를 구성한다. [Borůvka와 Kruskal MST 알고리즘 적용]

위에 제시한 알고리즘을 하이브리드 MST 알고리즘이라 하자. 하이브리드 MST 알고리즘의 상세한 내용은 (그림 6)과 같다. 하이브리드 MST 알고리즘이 Borůvka와 Kruskal MST 알고리즘과의 차이점은 <표 3>과 같다.

〈표 3〉 MST 알고리즘 차이점

구분		Borůvka MST 알고리즘	Kruskal MST 알고리즘	하이브리드 MST 알고리즘
행렬		정방행렬	상삼각행렬	정방행렬
Candidate Edges	대상	Vertex MWE → Inter-MSF MWE	모든 Edges	Vertex 1st MWE → Vertex 2nd MWE → Vertex 3rd MWE
	정렬	없음	오름차순	오름차순
	사이클 검증방법	동일	동일	동일

≪하이브리드 MST Algorithm ≫

정방행렬$\{i,j\}, i,j=1,2,\cdots n$ 작성
/* 1st Stage : 각 행 Vertex에 대해 1st MWE 선택, MSF 생성
n^{th} : Candidate MWE Group ($n=1,2,3$)

FOR $|e_{mst}|$ = 1 to $|v|-1$ /MSF 생성
  For 1 to $|v|$ do /* n^{th} Candidate MWE 생성
    각 행 Vertex에 대해 n^{th} MWE 선택 **(단, 동일한 값이 다수 존재시 모두 선택)**
    IF x와 $y \in n^{th}$ ($n=1,2,3$) Candidate Edges THEN **Skip /중복 선택**
    ELSE 선택된 MWE를 n^{th} Candidate Edges에 저장
  END
  n^{th} Candidate Edges를 오름차순 정렬
  FOR 1 to $|e_c|$ do /* Candidate Edges로부터 첫 번째 Edge부터 MSF Edges 선택
    IF $\{x,y\} \in$ MSF Vertices의 한 MSF 원소 THEN Skip /* Cycle 발생
    ELSE IF $\{x,y\} \in$ MSF Vertices의 두 MSF 원소 or $\{x,y\} \notin$ MSF Vertices THEN
      선택된 Edge를 MSF Edges에 추가 ($e_{mst}=e_{mst}+1$)
      선택된 Edge를 MSF Vertices의 해당 MSF에 연결 또는 새로운 MSF 생성
    ENDIF
    IF $|e_{mst}|=|v|-1$ THEN 알고리즘 종료
    ELSE LOOP
    ENDIF
  END
  IF $|f|=1$ THEN 알고리즘 종료
  ELSE $n=n+1$ THEN LOOP
END

(그림 6) 하이브리드 MST 알고리즘

3.2 적용 방법

G_1 그래프에 대해 하이브리드 MST 알고리즘을 적용한 결과는 <표 4>와 같으며, Kruskal MST 알고리즘과 동일한 MST를 얻을 수 있었다. 하이브리드 MST 알고리즘 1st MWE에서 $|e_c|=5$에 대해 $|e_s|=5$를 수행하여 $|f|=3$을, $|e_{mst}|=5$을 얻었다. $|f|=1$ 조건을 만족시키지 못해 다시 2nd MWE를 구성하여 $|e_c|=6$중 $|e_s|=3$에서 $|f|=1$ 조건을 충족시키고, $|e_{mst}|=2$를 얻어 사이클이 발생하지 않는 MST를 구하고 알고리즘이 종료되었다.

G_1 그래프에 대해 MST 알고리즘 수행 결과 $|e_c|/|e_s|$는

〈표 4〉 G_1 그래프의 하이브리드 MST 알고리즘 적용

Candidate Edges			중복 제거 Candidate Edges			Ascending Sorted Candidate Edges
1st MWE	2nd MWE	3rd MWE	1st MWE	2nd MWE	3rd MWE	
{a,d}=1 {b,c}=1 **{c,b}=1**,{c,e}=1 **{d,a}=1**,{d,g}=1 **{e,c}=1** {f,h}=1 **{g,d}=1** **{h,f}=1**	{a,b}=2 **{b,a}=2** {c,d}=2 **{d,c}=2**{d,f}=2 {e,f}=2 **{f,d}=2 {f,e}=2**{f,g}=2 **{g,f}=2**{g,h}=2 **{h,g}=2**	{a,c}=3 {b,e}=4 **{c,a}=3**{c,f}=3 – {e,h}=3 **{f,c}=3** – **{h,e}=3**	{a,d}=1 {b,c}=1 {c,e}=1 {d,g}=1 – {f,h}=1 – –	{a,b}=2 – {c,d}=2 {d,f}=2 {e,f}=2 {f,g}=2 {g,h}=2 –	{a,c}=3 {b,e}=4 {c,f}=3 – {e,h}=3 – – –	1st MWE : {a,d}=1,{b,c}=1 {c,e}=1,{d,g}=1 {f,h}=1 2nd MWE : {a,b}=2,{c,d}=2 {d,f}=2,{e,f}=2 {f,g}=2,{g,h}=2 3rd MWE : {a,c}=3,{c,f}=3 {e,h}=3,{b,e}=4

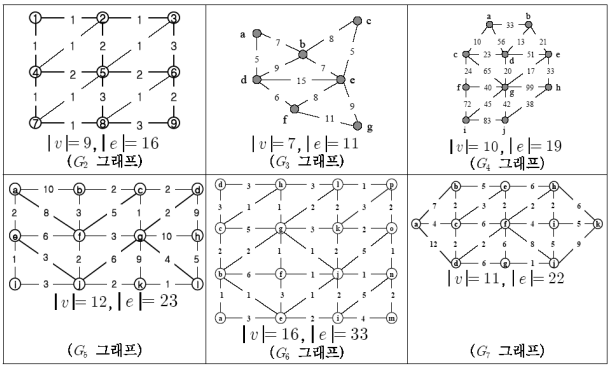
Candidate Edges		Cycle	MSF Vertices	MSF Edges
1st MWE	{a,d}=1 {b,c}=1 {c,e}=1 {d,g}=1 {f,h}=1	x x x x x	– a–d a–d, b–**c** a–**d**, b–c–e a–d–g, b–c–e a–d–g, b–c–e, f–h	{a,d}=1 {b,c}=1 {c,e}=1 {d,g}=1 {f,h}=1
	$\|e_c\|=5$	$\|e_s\|=5$	$\|f\|=3$	$\|e_{mst}\|=5$
2nd MWE	{a,b}=2 {c,d}=2 {d,f}=2 {e,f}=2 {f,g}=2 {g,h}=2	x O x	**a**–d–g, **b**–c–e, f–h a–b–**c**–**d**–e–g, f–h a–b–c–**d**–e–g, **f**–h a–b–c–d–e–f–g–h	{a,b}=2 – {d,f}=2 (알고리즘 종료)
	$\|e_c\|=6$	$\|e_s\|=3$	$\|f\|=1$	$\|e_{mst}\|=2$
3rd MWE	{a,c}=3 {c,f}=3 {e,h}=3 {b,e}=4			
	$\|e_c\|=4$			
$\|v\|-1=7$	$\|e_c\|=11$	$\|e_s\|=8$		$\|e_{mst}\|=7$

기존의 가장 좋은 성능을 보인 Kruskal MST 알고리즘의 15/8에 비해 하이브리드 MST 알고리즘은 11/8로 $|e_c|$는 4개를 줄일 수 있었으며, $|e_s|$는 동일한 결과를 얻었다. 따라서 제안된 알고리즘은 기존의 MST 알고리즘 보다 후보 간선들 수 감소로 알고리즘 수행에 필요한 메모리 감소와 더불어 수행속도를 향상시키는 효과를 얻었다.

4. 알고리즘 적용성 평가

4.1 실험에 적용된 그래프

본 절에서는 (그림 7)의 10개 그래프에 대해 Borůvka, Prim, Kruskal과 하이브리드 MST 알고리즘을 적용하여 MST를 얻을 수 있는지 살펴본다. G_2와 G_7 그래프는 Chen[7], G_3 그래프는 Wikipedia[9], G_4, G_5와 G_6 그래프는 Peiper[8]에서 인용되었다.



(그림 7) 실험에 적용된 그래프

4.2 적용 결과

4.2.1 G_2 그래프

G_2 그래프에 대해 Borůvka, Prim, Kruskal과 하이브리드 MST 알고리즘을 적용하여 얻은 MST는 (그림 8)에 제시되어 있다. G_2 그래프에 대해 Borůvka, Kruskal과 하이브리드 MST 알고리즘은 $|e_{mst}|=|v|-1=8$, $\Sigma w(e)=9$인 MST를 얻은 반면 Prim MST 알고리즘은 $|e_{mst}|=|v|-1$ 조건을 충족시키지 못한 상태로 사이클이 1개 발생하여 MST를 얻는데 실패하였다.

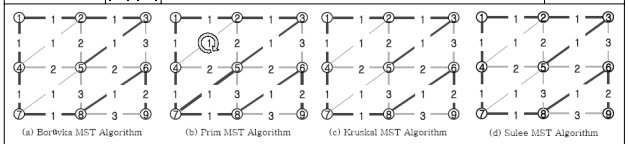
4.2.2 G_3 그래프

G_3 그래프에 대해 Borůvka, Prim, Kruskal과 하이브리드 MST 알고리즘을 적용하여 얻은 MST는 (그림 9)에 제시되어 있다. G_3 그래프에 대해 Borůvka, Prim, Kruskal과 하이브리드 MST 알고리즘 모두 $|e_{mst}|=|v|-1=6$, $\Sigma w(e)=39$인 MST를 얻는데 성공하였다.

4.2.3 G_4 그래프

G_4 그래프에 대해 Borůvka, Prim, Kruskal과 하이브리드

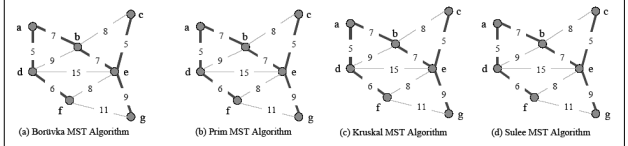
Borůvka MST 알고리즘	Stage	1st Stage						2nd Stage
	MSF	1-2-3-4-5-6-7-8-9						-
	e_{mst}	{1,2}=1,{1,4}=1→{2,3}=1→{3,5}=1→{4,7}=1→{6,8}=1→{7,8}=1→{9,6}=2						-
	$\|e_c\|/\|e_s\|$	19/19						-
Prim MST 알고리즘	Vertex	1	2, 4	3,7	5, 8	6		9
	e_{mst}	{1,2}=1 {1,4}=1	{2,3}=1 {4,7}=1	{3,5}=1 {7,5}=1 {7,8}=1	{8,6}=1	{6,9}=2		-
	$\|e_c\|/\|e_s\|$	2/2	8/8	8/8	13/13	7/7		-
Kruskal MST 알고리즘	e_{mst}	{1,2}=1→{1,4}=1→{2,3}=1→{3,5}=1→{4,7}=1→{6,8}=1→{7,8}=1→{6,9}=2						
	$\|e_c\|/\|e_s\|$	16/13						
하이브리드 MST 알고리즘	Group	1st MWE						2nd MWE
	MSF	1-2-3-4-5-6-7-8-9						-
	e_{mst}	{1,2}=1→{1,4}=1→{2,3}=1→{3,5}=1→{4,7}=1→{6,8}=1→{7,8}=1→{9,6}=2						-
	$\|e_c\|/\|e_s\|$	10/10						-



그래프	$\|v\|$	$\|e\|$	Borůvka MST 알고리즘			Prim MST 알고리즘			Kruskal & 하이브리드 MST 알고리즘		
			$\|e_{mst}\|$	Cycle	$\Sigma w(e)$	$\|e_{mst}\|$	Cycle	$\Sigma w(e)$	$\|e_{mst}\|$	Cycle	$\Sigma w(e)$
G_2	9	16	8	x	9	**9**	**1**	**10**	**8**	x	**9**

(그림 8) G_2 그래프의 MST

Borůvka MST 알고리즘	Stage	1st Stage						2nd Stage
	MSF	a-b-c-d-e-f-g						-
	e_{mst}	{a,d}=5→{b,a}=7→{b,e}=7→{c,e}=5→{f,d}=6→{g,e}=9						-
	$\|e_c\|/\|e_s\|$	8/8						-
Prim MST 알고리즘	Vertex	a	d	f	b	e	c	g
	e_{mst}	{a,d}=5	{d,f}=6	{a,b}=7	{b,e}=7	{e,c}=5	{e,g}=9	-
	$\|e_c\|/\|e_s\|$	2/2	5/5	6/6	8/8	9/9	5/5	-
Kruskal MST 알고리즘	e_{mst}	{a,d}=5→{c,e}=5→{d,f}=6→{a,b}=7→{b,e}=7→{e,g}=9						
	$\|e_c\|/\|e_s\|$	11/9						
하이브리드 MST 알고리즘	Group	1st MWE						2nd MWE
	MSF	a-b-c-d-e-f-g						-
	e_{mst}	{a,d}=5→{c,e}=5→{f,d}=6→{b,a}=7→{b,e}=7→{g,e}=9						-
	$\|e_c\|/\|e_s\|$	6/6						-



그래프	$\|v\|$	$\|e\|$	Borůvka MST 알고리즘			Prim MST 알고리즘			Kruskal & 하이브리드 MST 알고리즘		
			$\|e_{mst}\|$	Cycle	$\Sigma w(e)$	$\|e_{mst}\|$	Cycle	$\Sigma w(e)$	$\|e_{mst}\|$	Cycle	$\Sigma w(e)$
G_3	7	11	6	x	39	6	x	39	**6**	x	**39**

(그림 9) G_3 그래프의 MST

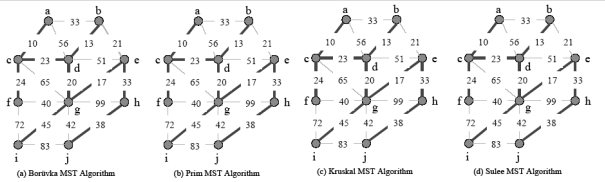
MST 알고리즘을 적용하여 얻은 MST는 (그림 10)에 제시되어 있다. G_4 그래프에 대해 Borůvka, Prim, Kruskal과 하이브리드 MST 알고리즘 모두 $|e_{mst}|=|v|-1=9$, $\Sigma w(e)=225$인 MST를 얻는데 성공하였다.

4.2.4 G_5 그래프

G_5 그래프에 대해 Borůvka, Prim, Kruskal과 하이브리드 MST 알고리즘을 적용하여 얻은 MST는 (그림 11)에 제시되어 있다.

G_5 그래프에 대해 Borůvka, Kruskal과 하이브리드 MST 알고리즘 모두 $|e_{mst}|=|v|-1=11$, $\Sigma w(e)=24$인 MST를 얻는데 성공하였다. 그러나 Prim MST 알고리즘은

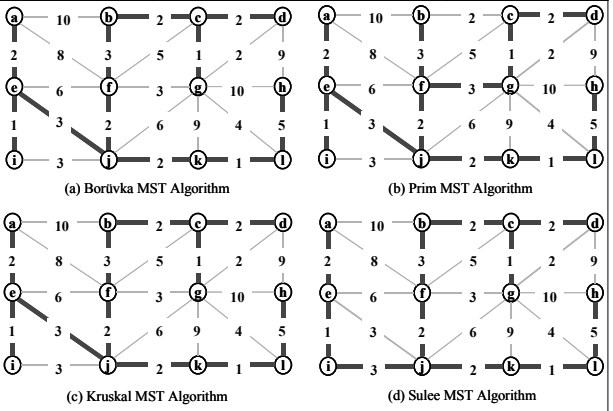
Borůvka MST 알고리즘	Stage	1st Stage						2nd Stage
	MSF	a-c-f, b-d, e-g-h-i-j						a-b-c-d-e-f-g-h-i-j
	e_{mst}	{a,c}=10→{b,d}=13→{e,g}=17→{f,c}=24→{h,e}=35→{i,g}=45→{j,h}=38						{c,d}=23,{f,g}=40,{d,g}=20 ⇒ {c,d}=23,{d,g}=20 선택
	$\|e_c\|/\|e_s\|$	10/10						9/9
Prim MST 알고리즘	Vertex	a	c	d	b	g	e	f	h	j	i
	e_{mst}	{a,c} =10	{c,d} =23	{d,b} =13	{d,g} =20	{g,e} =17	{c,f} =24	{e,h} =35	{h,j} =38	{g,i} =45	-
	$\|e_c\|/\|e_s\|$	3/3	6/6	9/9	8/8	11/11	11/11	8/8	7/7	6/6	-
Kruskal MST 알고리즘	e_{mst}	{a,c}=10→{b,d}=13→{e,g}=17→{d,g}=20→{c,d}=23→{c,f}=24→{e,h}=35→{h,j}=38→{g,i}=45									
	$\|e_c\|/\|e_s\|$	19/13									
하이브리드 MST 알고리즘	Group	1st MWE						2nd MWE			
	MSF	a-c-f, b-d, e-g-h-i-j						a-b-c-d-e-f-g-h-i-j			
	e_{mst}	{a,c}=10→{b,d}=13→{e,g}=17→{f,c}=24→{h,e}=35→{j,h}=38→{i,g}=45						{d,g}=20→{c,d}=23			
	$\|e_c\|/\|e_s\|$	7/7						7/3			



그래프	$\|v\|$	$\|e\|$	Borůvka MST 알고리즘			Prim MST 알고리즘			Kruskal & 하이브리드 MST 알고리즘		
			$\|e_{mst}\|$	Cycle	$\Sigma w(e)$	$\|e_{mst}\|$	Cycle	$\Sigma w(e)$	$\|e_{mst}\|$	Cycle	$\Sigma w(e)$
G_4	10	19	9	x	225	9	x	225	**9**	x	**225**

(그림 10) G_4 그래프의 MST

Borůvka MST 알고리즘	Stage	1st Stage								2nd Stage	
	MSF	a-e-i, b-c-d-g, f-h-j-k-l								a-b-c-d-e-f-g-h-i-j-k-l	
	e_{mst}	{a,e}=2→{b,c}=2→{c,g}=1→{d,c}=2→{e,i}=1→{f,j}=2→{h,l}=5→{j,k}=2→{k,l}=1								{a,b}=10,{b,f}=3,{g,f}=3,{e,j}=3,{i,j}=3 ⇒ {b,f}=3,{e,j}=3 선택	
	$\|e_c\|/\|e_s\|$	14/14								13/13	
Prim MST 알고리즘	Vertex	a	e	i	j	f,k	l		b,g	c	d	h
	e_{mst}	{a,e}= 2	{e,i}= 1	{e,j}= 3	{j,f}=2 {j,k}=2	{k,l}= 1	{f,b}=3 **{f,g}=3**		{g,c} =1	{c,d} =2	{l,h}= 5	-
	$\|e_c\|/\|e_s\|$	3/3	6/6	6/6	8/8	13/13	9/9		16/16	9/9	5/5	-
Kruskal MST 알고리즘	e_{mst}	{c,g}=1→{e,i}=1→{k,l}=1→{a,e}=2→{b,c}=2→{c,d}=2→{f,j}=2→{j,k}=2→{b,f}=3→{e,j}=3→{h,l}=5										
	$\|e_c\|/\|e_s\|$	23/11										
하이브리드 MST 알고리즘	Group	1st MWE								2nd MWE		
	MSF	b-c-d-g, a-e-i, f-h-j-k-l								a-b-c-d-e-f-g-h-i-j-k-l		
	e_{mst}	{c,g}=1→{e,i}=1→{k,l}=1→{a,e}=2→{b,c}=2→{d,c}=2→{f,j}=2→{j,k}=2→{b,f}=3→{e,j}=3→{h,l}=5								{b,f}=3→{i,j}=3		
	$\|e_c\|/\|e_s\|$	10/10								7/3		



그래프	$\|v\|$	$\|e\|$	Borůvka MST 알고리즘			Prim MST 알고리즘			Kruskal & 하이브리드 MST 알고리즘		
			$\|e_{mst}\|$	Cycle	$\Sigma w(e)$	$\|e_{mst}\|$	Cycle	$\Sigma w(e)$	$\|e_{mst}\|$	Cycle	$\Sigma w(e)$
G_5	12	23	11	x	24	11	x	**25**	**11**	x	**24**

(그림 11) G_5 그래프의 MST

$|e_{mst}| = |v| - 1 = 11$ 조건과 사이클이 발생하지 않는 조건은 만족시켰으나 $\Sigma w(e) = 25$로 MST는 얻지 못하였다. 즉, $\{b,c\} = 2$를 선택해야 하나 $\{f,g\} = 3$을 잘못 선택하였다.

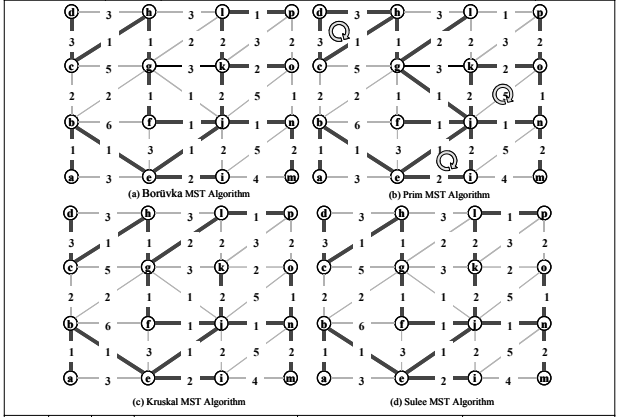
4.2.5 G_6 그래프

G_6 그래프에 대해 Borůvka, Prim, Kruskal과 하이브리드 MST 알고리즘을 적용하여 얻은 MST는 (그림 12)에 제시되어 있다. Kruskal과 하이브리드 MST 알고리즘은 $|e_{mst}| = |v| - 1 = 15$, $\Sigma w(e) = 21$인 MST를 얻는데 성공하였다. 그러나 Borůvka MST 알고리즘은 $|e_{mst}| = |v| - 1 = 15$ 조건은 만족시켰으나 $\Sigma w(e) = 22$로 MST를 얻는데 실패하였다. 즉, 하이브리드 MST 알고리즘에서 알 수 있듯이 $\{n,o\} = 1$을 선택해야 하나 $\{k,o\} = 2$를 잘못 선택하였다. 또한, Prim MST 알고리즘은 $|e_{mst}| = |v| - 1$와 사이클 미발생 조건을 만족시키지 못해 MST를 얻는데 실패하였다.

4.2.6 G_7 그래프

G_7 그래프에 대해 Borůvka, Prim, Kruskal과 하이브리드 MST 알고리즘을 적용하여 얻은 MST는 (그림 13)에 제시

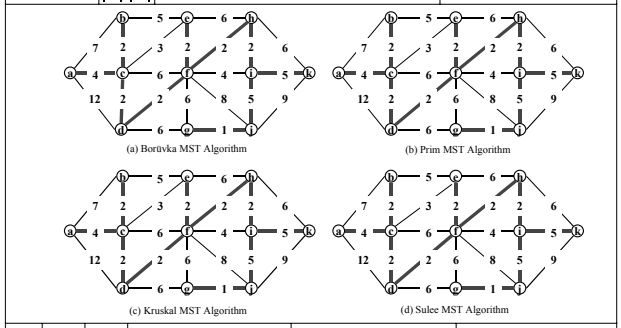
Borůvka MST 알고리즘	Stage	1st Stage								2nd Stage		
	MSF	a-b-c-d-e-f-g-h-i-j-k-l-m-n-o-p								-		
	e_{mst}	{a,b}=1→{b,e}=1→{c,h}=1→{d,c}=3→{e,j}=1→{f,g}=1→{f,j}=1→{g,h}=1→{i,e}=2→{j,n}=1→{k,j}=2→{k,l}=2→{k,o}=2→{l,p}=1→{m,n}=2								-		
	$\|e_{mst}\|/\|e_s\|$	30/26								-		
Prim MST 알고리즘	Vertex	a	b	e	j	f,g,n	h,o	c		i,k,l,m,p		d
	e_{mst}	{a,b}=1	{b,e}=1	{e,j}=1	{j,f}=1 {j,g}=1 {j,n}=1	{g,h}=1 {n,o}=1	{h,c}=1	{e,i}=2 {j,i}=2 {j,k}=2 {o,k}=2 {g,l}=2 {n,m}=2 {o,p}=2		{c,d}=3 {h,d}=3		-
	$\|e_{mst}\|/\|e_s\|$	2/2	6/6	9/9	12/12	23/23	18/18	17/17		23./23		-
Kruskal MST 알고리즘	e_{mst}	{a,b}=1→{b,e}=1→{c,h}=1→{e,j}=1→{f,g}=1→{f,j}=1→{g,h}=1→{j,n}=1 →{l,p}=1→{n,o}=1→{e,i}=2→{g,l}=2→{j,k}=2→{m,n}=2→{c,d}=3										
	$\|e_{mst}\|/\|e_s\|$	33/23										
하이브리드 MST 알고리즘	Group	1st MWE								2nd MWE		
	MSF	a-b-c-d-e-f-g-h-i-j-k-l-m-n-o-p								-		
	e_{mst}	{a,b}=1→{b,e}=1→{c,h}=1→{e,j}=1→{f,g}=1→{f,j}=1→{g,h}=1→{j,n}=1→{l,p}=1→{n,o}=1→{i,e}=2→{g,l}=2→{j,k}=2→{m,n}=2→{d,c}=3								-		
	$\|e_{mst}\|/\|e_s\|$	19/18								-		



그래프	$\|v\|$	$\|e\|$	Borůvka MST 알고리즘			Prim MST 알고리즘			Kruskal & 하이브리드 MST 알고리즘		
			$\|e_{mst}\|$	Cycle	$\Sigma w(e)$	$\|e_{mst}\|$	Cycle	$\Sigma w(e)$	$\|e_{mst}\|$	Cycle	$\Sigma w(e)$
G_6	16	33	15	x	**22**	**18**	**3**	29	**15**	x	**21**

(그림 12) G_6 그래프의 MST

Borůvka MST 알고리즘	Stage	1st Stage								2nd Stage		
	MSF	a-b-c-d-e-f-h-i-k, g-j								a-b-c-d-e-f-g-h-i-j-k		
	e_{mst}	{a,c}=4→{b,c}=2→{c,d}=2→{d,f}=2→{e,f}=2→ {f,h}=2→{g,j}=1→{h,i}=2→{k,i}=5								{i,j}=5		
	$\|e_{mst}\|/\|e_s\|$	16/16								5/5		
Prim MST 알고리즘	Vertex	a	c		b,d	f		e,h		i	j,k	g
	e_{mst}	{a,c}=4	{c,b}=2 {c,d}=2		{d,f}=2	{f,e}=2 {f,h}=2		{h,i}=2		{i,j}=5 {i,k}=5	{j,g}=1	-
	$\|e_{mst}\|/\|e_s\|$	3/3	7/7		11/11	11/11		14/14		9/9	11/11	-
Kruskal MST 알고리즘	e_{mst}	{g,j}=1→{b,c}=2→{c,d}=2→{d,f}=2→{e,f}=2→{f,h}=2→{h,i}=2→{a,c}=4→ {i,j}=5→{i,k}=5										
	$\|e_{mst}\|/\|e_s\|$	22/13										
하이브리드 MST 알고리즘	Group	1st MWE								2nd MWE		
	MSF	g-j, a-b-c-d-e-f-h-i-k								a-b-c-d-e-f-g-h-i-j-k		
	e_{mst}	{g,j}=1→{b,c}=2→{c,d}=2→{d,f}=2→{e,f}=2→ {f,h}=2→{h,i}=2→{a,c}=4→{k,i}=5								{j,i}=5		
	$\|e_{mst}\|/\|e_s\|$	9/9								9/4		



그래프	$\|v\|$	$\|e\|$	Borůvka MST 알고리즘			Prim MST 알고리즘			Kruskal & 하이브리드 MST 알고리즘		
			$\|e_{mst}\|$	Cycle	$\Sigma w(e)$	$\|e_{mst}\|$	Cycle	$\Sigma w(e)$	$\|e_{mst}\|$	Cycle	$\Sigma w(e)$
G_7	11	22	10	x	27	10	x	27	**10**	x	**27**

(그림 13) G_7 그래프의 MST

되어 있다. Borůvka, Prim, Kruskal과 하이브리드 MST 알고리즘 모두 $|e_{mst}| = |v| - 1 = 10$, $\Sigma w(e) = 27$인 MST를 얻는데 성공하였다.

4.3 적용 결과 분석

본 논문에 적용된 7개 그래프에 대한 알고리즘 수행 결과를 종합한 데이터는 <표 4>에 제시하였다.

Borůvka MST 알고리즘은 7개 그래프 중 G_6 그래프에서 MST를 얻는데 실패하였으며, Prim MST 알고리즘은 7개중 4개 그래프 (G_1, G_2, G_5와 G_6)에서 MST를 얻는데 실패하였다. 단지, Kruskal과 하이브리드 MST 알고리즘 만이 7개 그래프 모두에서 MST를 얻는데 성공하였다. 이와 같은 결과를 바탕으로 Kruskal과 하이브리드 MST 알고리즘 만이 간선들의 가중치가 모두 상이한 경우와 다수의 간선들 가중

〈표 4〉 MST 알고리즘의 MST

그래프	$\|v\|$	$\|e\|$	Borůvka MST 알고리즘			Prim MST 알고리즘			Kruskal & 하이브리드 MST 알고리즘		
			$\|e_{mst}\|$	Cycle	$\Sigma w(e)$	$\|e_{mst}\|$	Cycle	$\Sigma w(e)$	$\|e_{mst}\|$	Cycle	$\Sigma w(e)$
G_1	8	15	7	x	9	**8**	**1**	13	**7**	x	**9**
G_2	9	16	8	x	9	**9**	**1**	10	**8**	x	**9**
G_3	7	11	6	x	39	6	x	39	**6**	x	**39**
G_4	10	19	9	x	225	9	x	225	**9**	x	**225**
G_5	12	23	11	x	24	11	x	**25**	**11**	x	**24**
G_6	16	33	15	x	**22**	**18**	**3**	29	**15**	x	**21**
G_7	11	22	10	x	27	10	x	27	**10**	x	**27**

〈표 5〉 MST 알고리즘의 $|e_c|/|e_s|$ 비교

Graph	$\|V\|$	$\|e\|$	$\|e_c\|/\|e_s\|$							
			Borůvka MST 알고리즘			Prim MST 알고리즘	Kruskal MST 알고리즘	하이브리드 MST 알고리즘		
			1st Stage	2nd Stage	Total			1st MWE	2nd MWE	Total
G_1	8	15	10/10	9/ 9	**19/19**	**33/33**	**15/ 8**	5/ 5	6/ 3	**11/ 8**
G_2	9	16	19/19	-	**19/19**	**38/38**	**16/13**	10/10	-	**10/10**
G_3	7	11	8/ 8	-	**8/ 8**	**35/35**	**11/ 9**	6/ 6	-	**6/ 6**
G_4	10	19	10/10	9/ 9	**19/19**	**69/69**	**19/13**	7/ 7	7/ 3	**14/10**
G_5	12	23	14/14	13/13	**27/27**	**75/75**	**23/16**	10/10	7/ 3	**17/13**
G_6	16	33	30/26	-	**30/26**	**110/110**	**33/23**	19/18	-	**19/18**
G_7	11	22	16/16	5/5	**21/21**	**66/66**	**22/13**	9/9	9/4	**18/13**

치가 동일한 경우에도 일반적으로 적용할 수 있는 알고리즘으로 증명되었다.

다음으로 MST 알고리즘 수행에 따른 메모리 용량과 처리속도를 분석한 결과는 <표 5>와 같다. $|e_c|/|e_s|$ 평가 결과, 하이브리드 MST 알고리즘은 메모리 측면인 $|e_c|$에서 7개 그래프 모두에서 가장 적은 메모리를 차지하고 있다. 또한, 수행속도 측면인 $|e_s|$도 가장 빠르게 알고리즘이 종료됨을 알 수 있다. 따라서 제안된 하이브리드 MST 알고리즘을 적용하면 가장 효율성이 좋으면서도 항상 MST를 얻을 수 있을 것이다.

5. 결론 및 향후 연구과제

본 논문에서는 MST를 얻기 위해 Borůvka, Prim과 Kruskal MST 알고리즘을 실제 그래프에 적용하여 문제점을 고찰하고 새로운 MST 알고리즘을 제안하였다.

먼저 하나의 그래프에 적용한 결과 Borůvka와 Kruskal MST 알고리즘은 MST를 얻었지만 Prim MST 알고리즘은 MST를 얻는데 실패함을 보였다. 또한, Borůvka의 2nd Stage에서 Inter-MSF MWE를 선택하는 알고리즘이 복잡함을 알 수 있었다. 이 결과를 토대로, MST를 얻는데 문제가 되지 않는 Borůvka의 1st Stage와 Kruskal MST 알고리즘의 장점만을 채택한 하이브리드 MST 알고리즘을 제안하였다. 하이브리드 MST 알고리즘을 추가적으로 6개의 그래프에 적용한 결과 Kruskal MST 알고리즘과 동일하게 항상 MST를 얻음을 검증하였다. 또한, 알고리즘 수행속도와 메모리 용량 측면에서 비교한 결과 하이브리드 MST 알고리즘이 가장 좋은 성능을 보였다. 따라서 제안된 하이브리드 MST 알고리즘을 일반화된 MST 알고리즘으로 채택이 가능할 것이다.

본 논문에서 제안된 하이브리드 MST 알고리즘은 7개의 그래프에 대해 간선들의 가중치가 모두 다른 경우와 일부분이 동일한 가중치를 갖는 경우 모두 적용할 수 있는 일반화된 MST 알고리즘으로 검증되었다. 그러나 모든 다른 경우의 그래프에 대해 일반성을 검증하기에는 적용된 그래프의 수가 너무 작다고 판단된다. 따라서 추후 다른 그래프를 대상으로 제안된 알고리즘의 일반화된 적용성을 계속적으로 검증하고자 한다. 또한, Prim MST 알고리즘의 사이클 발생 문제점을 해결한 일반화된 Prim MST 알고리즘도 연구할 것이다.

참 고 문 헌

[1] Wikipedia, http://en.wikipedia.org/wiki/Minimum_spanning_tree

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

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

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

[9] Wikipedia, http://en.wikipedia.org/wiki/Kruskal_algorithm

이 상 운

e-mail : sulee@gwnu.ac.kr

1983년~1987년 한국항공대학교 항공전자공학과 (학사)

1995년~1997년 경상대학교 컴퓨터과학과 (석사)

1998년~2001년 경상대학교 컴퓨터과학과 (박사)

2003년 강원도립대학 컴퓨터응용과 전임강사

2004년~2007년 2월 국립 원주대학 여성교양과 조교수

2007년 3월~현 재 강릉원주대학교 과학기술대학 멀티미디어공학과 부교수

관심분야 : 소프트웨어 프로젝트 관리, 소프트웨어 개발 방법론, 소프트웨어 척도 (소프트웨어 규모, 개발노력, 개발기간, 팀 규모), 분석과 설계 방법론, 소프트웨어 시험 및 품질보증, 소프트웨어 신뢰성, 신경망, 뉴로-퍼지, 그래프 알고리즘