http://dx.doi.org/10.7236/JIIBC.2013.13.4.107
JIIBC 2013-4-15

선택-삭제 최소신장트리 알고리즘

Minimum Spanning Tree with Select-and-Delete Algorithm

최명복*, 이상운**

Myeong-Bok Choi, Sang-Un Lee

요 약 본 논문은 알고리즘 수행 횟수를 줄여 최소신장트리를 빨리 얻는 방법을 제안하였다. 제안된 알고리즘은 선택과 삭제 과정을 수행한다. 선택 과정은 먼저, 그래프의 모든 정점들에 대해 Borůvka의 첫 번째 단계를 수행하고, 특정 정점들에 대해 Borůvka의 첫 번째 단계를 재 수행하여 간선들의 모집단을 축소시키는 결과를 얻었다. 삭제 과정은 축소된 모집단 간선들에 대해 3개 정점들 간에 사이클이 발생할 경우 최대 가중치 간선을 삭제한다. 나머지 간선들 중 최대 가중치 간선에 대해 결합가 개념을 적용하여 삭제한다. 마지막으로 결합가가 큰 정점들 간의 사이클이 발생하는 경우 최대 가중치 간선을 삭제하는 기법을 적용하였다. 선택-삭제 알고리즘을 9개의 다양한 그래프에 적용하여 알고리즘 적용성을 평가하였다. 제안된 선택 과정은 MST 알고리즘을 최적으로 수행해야 하는 간선의 수와 비교시 6개는 적은 개수를, 3개 그래프만이 1개 큰 간선을 선택하는 결과를 나타내어 최적으로 간선을 선택하는 방법임을 알 수 있다. 삭제 단계를 Kruskal 알고리즘을 적용할 경우 Kruskal 알고리즘을 최적으로 수행하는 횟수와 비교한 결과 6개의 그래프는 수행 횟수가 적은 반면, 3개 그래프는 1회 많게 수행하는 결과를 얻었다. 또한, 제안된 삭제 단계를 수행할 경우 1개 그래프는 1단계만, 5개 그래프는 2단계까지, 나머지 3개 그래프만이 3단계를 수행하는 결과를 나타내었다. 결국, 선택-삭제 알고리즘이 MST 알고리즘들 중에서 가장 적은 수행 횟수를 나타내었다.

Abstract This algorithm suggests a method in which a minimum spanning tree can be obtained fast by reducing the number of an algorithm execution. The suggested algorithm performs a select-and-delete process. In the select process, firstly, it performs Borůvka's first stage for all the vertices of a graph. Then it re-performs Borůvka's first stage for specific vertices and reduces the population of the edges. In the delete process, it deletes the maximum weight edge if any cycle occurs between the 3 edges of the edges with the reduced population. After, among the remaining edges, applying the valency concept, it gets rid of maximum weight edges. Finally, it eliminates the maximum weight edges if a cycle happens among the vertices with a big valency. The select-and-delete algorithm was applied to 9 various graphs and was evaluated its applicability. The suggested select process is believed to be the vest way to choose the edges, since it showed that it chose less number of big edges from 6 graphs, and only from 3 graphs, comparing to the number of edges that is to be performed when using MST algorithm. When applied the delete process to Kruskal algorithm, the number of performances by Kruskal was less in 6 graphs, but 1 more in each 3 graph. Also, when using the suggested delete process, 1 graph performed only the 1st stage, 5 graphs till 2nd stage, and the remaining till 3rd stage. Finally, the select-and-delete algorithm showed its least number of performances among the MST algorithms.

Key Words : Minimum Spanning Tree, Valency, Select and Delete, Cycle, Minimum Weight Edge, Maximum Weight Edge

*종신회원, 강릉원주대학교 멀티미디어공학과
**정회원, 강릉원주대학교, 멀티미디어공학과
접수일자 : 2013년 2월 20일, 수정완료 : 2013년 6월 28일
게재확정일자 : 2013년 8월 16일

Received: 20 February, 2013 / Revised: 28 June, 2013 / Accepted: 16 August, 2013
*Corresponding Author: cmb5859@gmail.com
Dept. of Multimedia Engineering, Gangnung-Wonju National University, Korea

Ⅰ. 서 론

최소신장트리 (Minimum Spanning Tree, MST)는 그래프 $G=(V,E)$의 모든 정점들 (Vertices, n)을 연결하는 $n-1$개의 간선들 (Edges, m)의 가중치 합 $\Sigma w(m)$이 최소가 되는 신장트리 (Spanning Tree, ST)로 전기, 전화, 가스 또는 수도 분야에 활용되고 있다.[1]

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

Borůvka 알고리즘[2,3]은 첫 번째 단계 (1st Stage)에서 각 정점에 부속된 최소 가중치 간선 (Minimum Weight Edge, MWE)을 선택하고 이들 중 사이클 발생시 최대 가중치 간선을 제거하여 최소 신장 포레스트 (Minimum Spanning Forest, MSF)를 구성한다. 두 번째 단계 (2nd Stage)에서 이들 MSF 간에 MWE를 다시 찾는 방법이다. Prim 알고리즘[4]은 임의의 정점을 선택하고 이에 부속된 모든 간선들 중에서 MWE를 선택하여 다음 정점을 결정하고 한번에 하나씩 기존에 방문한 간선들과 새로운 정점에 부속된 간선들 중에서 다시 MWE를 선택하는 방법으로 모든 정점들을 방문해야 한다. Kruskal 알고리즘[5]은 모든 간선들을 오름차순으로 정렬시키고 최소 가중치 간선부터 시작하여 사이클이 발생하지 않는 간선을 한번에 하나씩 선택하면서 신장트리를 구성하는 방법으로 모든 간선들을 대상으로 한다. 역-삭제 알고리즘[6]은 모든 간선들을 내림차순으로 정렬시키고 최대 가중치 간선부터 시작하여 한번에 하나씩 정점을 분리시키지 않는 간선을 제거하는 알고리즘이다.

Borůvka, Prim, Kruskal과 역-삭제 알고리즘 모두 모집단은 그래프의 모든 간선들이다. Borůvka 알고리즘은 2번에 걸쳐 다수의 간선을 선택하는 방법으로 알고리즘 수행 횟수가 가장 적을 수 있다. 그러나 첫 번째 단계에서 선택되는 간선들은 중복되는 경우가 많아 MST를 찾는데 필요한 간선들을 충분히 선택할 수 없다. 또한, 두 번째 단계에서 MSF 간에 부속된 MWE를 프로그램으로 구현하여 찾기가 쉽지 않은 단점을 갖고 있다. Prim 알고리즘은 모든 정점들을 한번에 하나씩 방문하기 때문에 더 이상의 수행횟수를 줄일 수 있는 방법이 없다. 그러나 동일한 MWE를 가지는 다수의 정점을 동시에 방문한다면 알고리즘 수행횟수를 줄일 수 있을 것이다. Kruskal 알고리즘은 선택된 간선의 수가 $n-1$ 개가 되면 MST를 얻어 알고리즘을 종료시킬 수 있는 방법이 있으나 불필요하게 나머지 간선들을 확인하는 과정을 거친다. 역-삭제 알고리즘도 남아있는 간선의 수가 $n-1$개 일 때 MST를 얻음에도 불구하고 남아있는 모든 간선들을 불필요하게 확인하는 과정을 거친다.

본 논문에서는 그래프의 모든 간선들을 모집단으로 설정하지 않고 MST에 기여하지 않는 간선들을 사전에 가능한 많이 제거하여 수행 횟수를 획기적으로 줄일 수 있는 알고리즘을 제안한다. 제안된 알고리즘은 Borůvka의 첫 번째 단계를 2회 수행하여 간선들을 선택하고, 역-삭제 알고리즘의 변형 형태를 적용하여 최대 가중치 간선을 3회에 걸쳐 제거하는 방법으로 선택-삭제 (Select-Delete, SD) 방법이다.

2장에서는 대표적인 MST 알고리즘인 Borůvka Prim, Kruskal과 역-삭제 알고리즘을 고찰하고 실제 그래프에 적용하고 문제점을 살펴본다. 3장에서는 알고리즘 수행 횟수를 획기적으로 줄일 수 있는 선택-삭제 알고리즘을 제안한다. 4장에서는 8개의 그래프에 실제 적용하여 제안된 알고리즘의 적용성과 성능을 검증해본다.

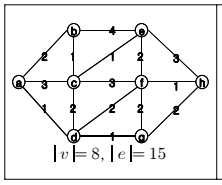
Ⅱ. 관련 연구와 연구 배경

무방향 그래프 (Undirected Graph) $G=(V,E)$의 간선은 집합 $\{x,y\}$로 표기한다. 방향성 그래프 (Digraph)의 간선은 호 (Arc)로 부르며 방향성을 가진 순서쌍 (x,y)로 표기한다. 일반적으로 그래프는 무방향 그래프를 의미하며, MST는 무방향성 그래프를 대상으로 하기 때문에 본 논문에서는 간선을 $\{x,y\}$로 표기한다.

Borůvka 알고리즘은 빅뱅 선택-삭제-빅뱅 선택 방식으로, 첫 번째 단계에서 각 정점에 부속된 간선들 중 최소 가중치 간선 (MWE)을 선택하고 중복 선택된 간선과 사이클이 발생하는 최대 가중치 간선을 제거하여 최소 신장 포레스트 (MSF)를 구성한다. 두 번째 단계에서 MSF들을 상호 연결하는 MWE를 "MSF 수 - 1"개를 찾는 방법이다. Prim 알고리즘은 점진적 구성 방식으로 임의의 정점을 선택하고, 이에 부속된 간선들 중에서 MWE를 선택한다. 다음으로 새로 선택된 정점에 부속된

간선들의 가중치와 기존에 방문한 정점에서 선택되지 않은 간선의 가중치들 중에서 MWE를 선택하는 방법이다. (단, 이 과정에서 사이클이 발생하는 간선은 무시한다.) 이 방법을 모든 정점들이 선택될 때까지 수행한다. Kruskal 알고리즘은 점진적 구성 방식으로 그래프의 모든 간선들을 대상으로 오름차순으로 정렬시키고, 첫 번째 MWE부터 시작하여 사이클이 발생하지 않는 한 간선들을 한번에 하나씩 선택하는 방법이다. 이 과정을 모든 간선들에 대해 수행한다. 역-삭제 알고리즘은 점진적 파괴 방식으로 Kruskal 알고리즘의 반대 개념으로 그래프의 모든 간선들을 대상으로 내림차순으로 정렬시키고, 첫 번째 최대 가중치 간선부터 시작하여 한번에 하나씩 i 또는 j 정점을 그래프에서 분리시키지 않는 간선 $\{i,j\}$를 제거하는 방법이다. 이 과정을 모든 간선들에 대해 수행한다.

그림 1의 G_1 그래프를 대상으로 MST 알고리즘을 적용하여 보자. G_1 그래프는 Peiper[7]에서 인용되었다.



	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 1. G_1 그래프
Fig. 1. G_1 Graph

대표적인 MST 알고리즘인 Borůvka, Prim, Kruskal과 역-삭제 알고리즘을 그림 1의 G_1 그래프에 적용하여 MST를 구하는 과정은 표 1에, 결과는 그림 2에 제시하였다. Borůvka 알고리즘은 첫 번째 단계에서 MST를 얻지 못하고 $a-d-g$, $b-c-e$와 $f-h$의 MSF를 얻었다. 이들 3개 MSF를 연결하는 MWE (Inter-MSF MWE)를 찾기 위해 두 번째 단계에서 9개의 간선들 중 Inter-MSF MWE 2개를 선택하여 $|n|-1=7$개의 간선을 찾고 $\Sigma w(m)=9$인 MST를 얻었다. 이와 같이 Borůvka 알고리즘은 2번째 단계에서 Inter-MSF MWE를 찾는 방법이 어렵다. Prim 알고리즘은 $|n|-1=7$개의 간선을 얻는 과정에서 새로 추가되거나, 기존에 방문하여 선택되지 않고 남아 있는 59개의 간선들을 모두 비교하여 7개의 간선을 선택하고 $\Sigma w(m)=9$인 MST를 얻었다.

표 1 G_1 그래프의 MST 알고리즘 적용
Table 1. MST Algorithm applied to Graph G_1

(a) Borůvka 알고리즘

1st Stage				2nd Stage	
후보 간선들	사이클	MSF 정점	MST 간선	Inter-MSF MWE	MST 간선
{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	– x O x O x O x O O	– a-d a-d, b-c a-d, b-c a-d, b-c-e a-d, b-c-e a-d-g, b-c-e a-d-g, b-c-e a-d-g, b-c-e, f-h a-d-g, b-c-e, f-h	{a,d}=1 {b,c}=1 – {c,e}=1 – {d,g}=1 – {f,h}=1 – –	<table><tr><th></th><th>b c e</th><th>f h</th></tr><tr><td>a</td><td>**2** 3 –</td><td>– –</td></tr><tr><td>d</td><td>– **2** –</td><td>**2** –</td></tr><tr><td>g</td><td>– – –</td><td>**2 2**</td></tr></table> <table><tr><th></th><th>f h</th></tr><tr><td>b</td><td>– –</td></tr><tr><td>c</td><td>3 –</td></tr><tr><td>e</td><td>**2** 3</td></tr></table> a-d-g와 b-c-e MSF : {a,b}=2, {d,c}=2, 중 {a,b}=2 선택 a-d-g와 f-h MSF : {d,f}=2, {g,f}=2, {g,h}=2 중 {d,f}=2 선택 b-c-e와 f-h MSF : {e,f}=2는 무시	{a,b}=2 {d,f}=2

(b) Prim 알고리즘

V	MST 정점	MST 후보 간선들				MST 간선
		추가된 간선	남아있는 간선	삭제된 간선	최종 남은 간선	
{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
{c,e,f,h}	{a,d,g,b}	**{b,a}=2**, {b,c}=1 {b,e}=4	{a,c}=3, {d,c}=2 {d,f}=2, {g,f}=2 {g,h}=2	{b,a}=2	{a,c}=3, {d,c}=2 {d,f}=2, {g,f}=2 {g,h}=2,**{b,c}=1** {b,e}=4	{b,c}=1
{e,f,h}	{a,d,g,b,c}	**{c,a}=3** **{c,b}=1,{c,d}=2** {c,e}=1, {c,f}=3	**{a,c}=3,{d,c}=2** {d,f}=2, {g,f}=2 {g,h}=2,{b,e}=4	{a,c}=3, {d,c}=2 {c,a}=3,{c,b}=1 {c,d}=2	{d,f}=2, {g,f}=2 {g,h}=2, {b,e}=4 **{c,e}=1**, {c,f}=3	{c,e}=1
{f,h}	{a,d,g,b,c,e}	**{e,b}=4,{e,c}=1** {e,f}=2, {e,h}=3	{d,f}=2, {g,f}=2 {g,h}=2, **{b,e}=4** {c,f}=3	{b,e}=4, {e,b}=4 {e,c}=1	**{d,f}=2**, {g,f}=2 {g,h}=2, {c,f}=3 {e,f}=2, {e,h}=3	{d,f}=2
{h}	{a,d,g,b,c,e,f}	**{f,c}=3,{f,d}=2** **{f,e}=2,{f,g}=2** {f,h}=1	**{g,f}=2**, {g,h}=2 **{c,f}=3,{e,f}=2** {e,h}=3	{g,f}=2, {c,f}=3 {e,f}=2, {f,c}=3 {f,d}=2, {f,e}=2 {f,g}=2	{g,h}=2, {e,h}=3 **{f,h}=1**	{f,h}=1
{ ∅ }	{a,d,g,h,c,e,f,h}	**{h,e}=3,{h,f}=1** **{h,g}=2**	**{g,h}=2,{e,h}=3**	{g,h}=2,{e,h}=3 {h,e}=3,{h,f}=1 {h,g}=2	{ ∅ }	–

(c) Kruskal 알고리즘

MST 후보 간선들			MST 정점	MST 간선
모든 정점	오름차순 정렬	사이클 발생		
{a,b}=2	{a,d}=1	x	–	{a,d}=1
{a,c}=3	{b,c}=1	x	{a,d}	{b,c}=1
{a,d}=1	{c,e}=1	x	{a,d}, {b,**c**}	{c,e}=1
{b,c}=1	{d,g}=1	x	{a,**d**}, {b,c,e}	{d,g}=1
{b,e}=4	{f,h}=1	x	{a,d,g}, {b,c,e}	{f,h}=1
{c,d}=2	{a,b}=2	x	{**a**,d,g}, {**b**,c,e}, {f,h}	{a,b}=2
{c,e}=1	{c,d}=2	O	{a,b,**c**,**d**,e,g}, {f,h}	–
{c,f}=3	{d,f}=2	x	{a,b,c,**d**,e,g}, {**f**,h}	{d,f}=2
{d,f}=2	{e,f}=2	O	{a,b,c,d,e,f,g,h}	–
{d,g}=1	{f,g}=2	O	{a,b,c,d,e,f,g,h}	–
{e,f}=2	{g,h}=2	O	{a,b,c,d,e,f,g,h}	–
{e,h}=3	{a,c}=3	O	{a,b,c,d,e,f,g,h}	–
{f,g}=2	{c,f}=3	O	{a,b,c,d,e,f,g,h}	–
{f,h}=1	{e,h}=3	O	{a,b,c,d,e,f,g,h}	–
{g,h}=2	{b,e}=4	O	{a,b,c,d,e,f,g,h}	–

(d) 역-삭제 알고리즘

MST 후보 간선들			삭제	MST 간선
모든 정점	내림차순 정렬	정점 분리		
{a,b}=2	{b,e}=4	x	{b,e}=4	–
{a,c}=3	{e,h}=3	x	{e,h}=3	–
{a,d}=1	{c,f}=3	x	{c,f}=3	–
{b,c}=1	{a,c}=3	x	{a,c}=3	–
{b,e}=4	{g,h}=2	x	{g,h}=2	–
{c,d}=2	{f,g}=2	x	{f,g}=2	–
{c,e}=1	{e,f}=2	x	{e,f}=2	–
{c,f}=3	{d,f}=2	O	–	{d,f}=2
{d,f}=2	{c,d}=2	x	{c,d}=2	–
{d,g}=1	{a,b}=2	O	–	{a,b}=2
{e,f}=2	{f,h}=1	O	–	{f,h}=1
{e,h}=3	{d,g}=1	O	–	{d,g}=1
{f,g}=2	{c,e}=1	O	–	{c,e}=1
{f,h}=1	{b,c}=1	O	–	{b,c}=1
{g,h}=2	{a,d}=1	O	–	{a,d}=1

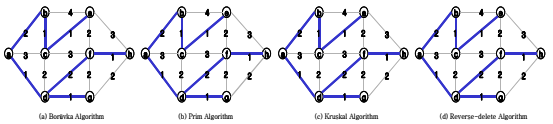


그림 2. G_1 그래프의 MST 알고리즘 적용 결과
Fig. 2. Result of MST Algorithm applied to Graph G_1

Kruskal 알고리즘은 그래프의 모든 간선들인 15개를 대상으로 오름차순으로 정렬하여 순차적으로 비교한 결과 8번째인 $\{d,f\}=2$에서 $|n|-1=7$개의 간선과 $\Sigma w(m)=9$인 MST를 얻어 알고리즘을 종료시켜도 되나 나머지 7개의 간선을 추가적으로 비교한다. 역-삭제 알고리즘은 그래프의 모든 간선들인 15개를 대상으로 내림차순으로 정렬하여 순차적으로 비교한 결과 9번째인 $\{c,d\}=2$를 삭제하면 $|n|-1=7$개의 간선과 $\Sigma w(m)=9$인 MST를 어 알고리즘을 종료시켜도 된다. 그러나 나머지 6개의 간선을 추가적으로 비교한다. 또한, 그래프를 시각적으로 확인하지 않는 상황에서 정점이 분리되는지에 대한 판단을 하기 어려운 단점이 있다.

그래프의 간선들을 오름차순으로 정렬시키면 그림 3으로 표현할 수 있다. 여기서 α와 γ는 사이클이 발생하지 않는 가중치가 작은 간선들로 MST의 간선이며, β는 가중치는 γ보다 적지만 사이클에서의 최대 가중치 간선이다. γ는 β보다 큰 가중치를 갖고 있지만 특정 사이클 내에서는 최대 값이 되지 않아 MST로 선택되는 간선이다. δ는 MST에 기여를 하지 못하는 사이클에서의 최대 가중치 간선이다. 따라서 MST를 얻기 위해서는 β와 δ를 삭제해야만 한다. 실제적으로 β와 γ 관계는 $\gamma>\beta$인 경우와 β와 γ가 혼재되어 존재할 수 있다. 그림 3에서는 편의상 $\gamma>\beta$로 표현하였다.

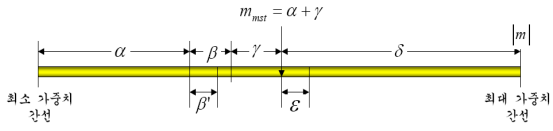


그림 3. 그래프의 간선 오름차순 표현
Fig. 3. Ascending Representation of Graph Edges

결국, 그래프의 총 간선의 수 $|m|$에서 우리가 찾고자 하는 MST 간선은 $|m_{mst}|=\alpha+\gamma=|n|-1$가 된다. 이를 찾기 위해 Kruskal 알고리즘은 $|m|=\alpha+\beta+\gamma+\delta$을 대상으로 $\alpha+\beta+\gamma$개가 되는 시점까지 사이클이 발생하는 경우의 최대 가중치 간선 β를 제거하면 $|n|-1$개의 간선으로 알고리즘을 빨리 종료시킬 수 있다. 역-삭제 알고리즘은 $\delta+\gamma+\beta$를 대상으로 $\delta+\beta$를 삭제하면 남아 있는 간선 $\alpha+\gamma$가 MST가 되어 알고리즘을 빨리 종료시킬 수 있다. Borůvka 알고리즘은 첫 번째 단계에서 $\alpha+\beta$를 선택 한 후 β를 제거하고 두 번째 단계에서 γ를 선택한다. 그러나 실제적으로는 첫 번째 단계에서 β까지 선택하지 못할 수도 있으며, α도 충분히 선택하지 못할 경우도 발생한다. Prim 알고리즘은 $|n|$에 연결된 $|m|$을 대상으로 $\alpha+\gamma$를 선택하면서 $\beta+\delta$를 제거한다. Borůvka와 Prim 알고리즘은 더 이상 빨리 알고리즘을 종료시킬 수 있는 방법이 없다.

그래프에서 MST에 기여를 하지 못하는 $\delta+\beta$ 간선을 사전에 모두 제거할 수 있는 방법은 없는가? 현실적으로 그래프의 정점들이 복잡하게 연결되어 있고 각 정점에 부속된 간선들 가중치들이 다른 정점에 부속된 간선의 가중치 보다 적음에도 불구하고 MST에 기여하지 못하는 복잡성이 존재한다. 즉, 이상적인 경우로 $\alpha<\gamma<\beta<\delta$ 관계가 성립한다면 $\beta+\delta$를 쉽게 구별할 수 있지만 실제로는 β와 γ가 혼재되어 있고 어떤 간선이 MST에 속하는지 알 수 없는 경우가 대부분이다. 따라서 $\delta+\beta$ 간선들을 사전에 모두 제거할 수 있는 방법은 지금까지 제안되지 않고 있다. 결국, 이러한 문제점을 해결하기 위해서는 $\delta+\beta$를 사전에 가능한 많이 제거하는 방법과 더불어 이와 같이 축소된 모집단을 대상으로 알고리즘을 빨리 종료시키는 방법이 요구된다.

Ⅲ. 선택-삭제 MST 알고리즘

1. 알고리즘

본 장에서는 먼저 $\delta+\beta$를 사전에 가능한 많이 제거하는 선택 방법과 축소된 모집단을 대상으로 알고리즘을 빨리 종료시키는 제거 방법인 선택-제거 알고리즘을 제안한다. 간선 선택 방법은 Borůvka의 첫 번째 단계를 1회 수행하고 다시 특정 정점들에 대해 Borůvka의 첫 번째 단계를 재 수행하여 $\alpha+\beta+\gamma+\epsilon$ 또는 $\alpha+\beta'+\gamma$만을 선택하여 모집단의 수를 줄이는 방법이다. 다음으로 축소된 간선 모집단을 대상으로 MST를 빨리 찾아 수행 횟수를 단축시키는 문제를 해결해야 한다. 만약에, 축소된 간선 모집단을 대상으로 Kruskal 알고리즘을 적용할

경우 $\alpha+\beta+\gamma$개가 되는 시점까지 사이클이 발생에서의 최대 가중치 간선 β를 제거하고 알고리즘을 종료시키는 방법과 동일한 수행 횟수를 얻는다. 예로, G_1 그래프의 경우 8번째인 $\{d,f\}=2$까지만 수행하면 된다. 그러나 항상 $(\alpha+\beta+\gamma)>(\beta+\epsilon)$이 성립하기 때문에 최대 가중치 간선 $\beta+\epsilon$를 제거하는 것이 알고리즘 수행 횟수를 획기적으로 줄일 수 있다. 따라서 본 논문에서는 이 방법을 적용한다. 그러나 역-삭제 알고리즘은 하나의 간선을 제거하였을 경우 이에 연결된 정점이 그래프에서 분리되는지를 판단하는 자동화된 방법을 찾지 못하고 있다. 만약, 이에 대해 알고리즘으로 쉽게 구현할 수 있는 방법이 존재한다면 본 논문에서 제안하는 방법을 개선하는 결과를 얻을 것이다. 이 문제는 추후 연구과제로 남겨두고 본 장에서는 축소된 간선 모집단을 대상으로 3회의 삭제 과정을 거쳐 MST를 얻는 방법으로 이 문제를 해결한다. 즉, 첫 번째로 i,j,k 정점의 간선 $\{i,k\}$, $\{j,k\}$, $\{i,j\}$가 존재하여 사이클이 발생할 경우 이들 중 최대 가중치 간선을 삭제한다. 두 번째로 남아있는 간선 들 중 최대 가중치 간선 $\{i,j\}$에 대해 두 정점 i와 j의 결합가 (Valency, v)가 2 이상이면 최대 가중치 간선 $\{i,j\}$를 삭제한다. 세 번째로, 결합가가 2 이상인 두 정점 i와 j사이에 사이클이 발생하는 경우 최대 가중치 간선을 삭제하는 방법을 적용한다.

선택-삭제 알고리즘을 설명하기 위해 그래프의 정점들을 5개 집합으로 분류한다. 시작 정점을 S, 마지막 종료 정점을 D, 시작 정점에 인접한 정점들을 P_S (시작 정점 행의 간선들), 종료 정점에 인접한 정점들을 P_D (종료 정점 열의 간선들), 시작과 종료 정점에 인접하지 않은 정점들을 P로 분류하자. 그림 1 G_1 그래프의 정방행렬에서 이들 집합을 구하면 $S=\{a\}$, $D=\{h\}$, $P_S=\{b,c,d\}$, $P_D=\{e,f,g\}$, $P=\{\varnothing\}$가 된다. 이로부터 $P+P_S+P_D=\{b,c,d,e,f,g\}$를 얻을 수 있다. 또한 행 (Row) 정점에서 선택된 간선의 수를 행 결합가 (v_r), 열 (Column)정점에서 선택된 간선의 수를 열 결합가 (v_c)라 한다. 결국, 정점 i의 결합가 $v(i)=v_r(i)+v_c(i)$이다. 선택-삭제 알고리즘은 그림 4와 같이 수행된다.

초기 조건 : 가중치 간선 정방행렬 작성
[선택] : 정방행렬
S_1 : 최소 가중치 간선 (MWE) 1차 선택
1. 행 정점들 : MWE $\{i,j\}$ 1개 선택 (단, 동일 값 모두 선택)
2. 선택된 $\{i,j\}$에 대해 $\{j,i\}$ 선택
3. 하삼각행렬 (Lower Triangle Matrix)에서 선택된 MWE 삭제

S_2 : MWE 2차 선택
1. P_S+P 행 정점 : 상삼각행렬에서 $v_r=0$인 경우, MWE 1개 선택 (단, 동일한 값은 모두 선택)
2. P_S+P+P_D 열 정점 : 상삼각행렬에서 $v_c=0$이 경우, MWE 1개 선택 (단, 동일한 값은 모두 선택)
3. 선택된 $\{i,j\}$에 대해 $\{j,i\}$ 선택, 상삼각행렬의 선택되지 않은 간선들과 하삼각행렬 모두 삭제

만약 선택된 MWE 개수가 $n-1$개이면 알고리즘 종료
[삭제] : 상삼각행렬
D_1 : $v_c\geq 2$인 열 정점
$\{i,k\}$, $\{j,k\}$에 대해 $\{i,j\}$가 존재시 이들 중 최대 가중치 간선 삭제 (만약, 최대 가중치가 동일하면 임의로 하나 삭제)
만약 남아있는 MWE 개수가 $n-1$개이면 알고리즘 종료
D_2 : P_S+P+P_D 행과 P_S+P+P_D열 : 최대 가중치 간선 $\{i,j\}$에 대해
만약 $v(i)\geq 2$와 $v(j)\geq 2$이면 최대 가중치 간선 삭제 (만약 동일한 가중치에 대해 모두 조건을 만족하면 임의로 하나 삭제)
만약 남아있는 MWE 개수가 $n-1$개이면 알고리즘 종료
D_3 : $v_r(i)\geq 2$인 정점에서 $v_c(j)\geq 2$인 정점까지의 경로 선택, 최대 가중치 간선 삭제 (만약, 모두 동일한 값이면 $v_c(j)\geq 2$인 정점의 가중치 값 1개 삭제)

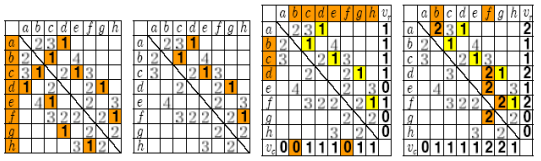
그림 4. 선택-삭제 알고리즘
Fig. 4. Select-Delete Algorithm

2. 선택-삭제 알고리즘 적용 방법

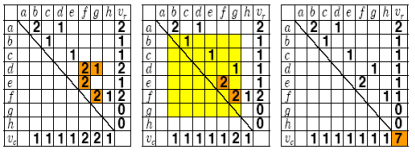
G_1 그래프에 대해 선택-삭제 알고리즘을 적용한 결과는 표 2와 같으며, 기존의 MST 알고리즘들과 동일한 MST를 얻음을 알 수 있다. G_1 그래프는 $|n|=8$이므로 $\alpha+\gamma=|n|-1=7$이 된다. Kruskal과 역-삭제 알고리즘에 따르면 $\alpha+\beta+\gamma=8$, $\delta=7$로 $\beta=1$이며, 선택-삭제 알고리즘에 따르면 $\epsilon=1$이 된다. G_1 그래프에 대한 MST 알고리즘의 수행 횟수를 비교한 결과는 표 3에 제시되어 있다.

표 2. G_1 **그래프의 선택-삭제 알고리즘 적용**
Table 2. Select-Delete Algorithm applied to Graph G_1

(a) S_1 단계 (b) S_2 단계



(c) D_1 단계 (d) D_2 단계



(c) $\{d,f\}=2,\ \{e,f\}=2,\ \{d,e\}=\varnothing$
$\{d,g\}=1,\ \{f,g\}=2,\ \{d,f\}=2 \rightarrow \{d,f\}=2$ 삭제

(d) $\{e,f\}=2 \rightarrow v(e)=2, v(f)=3,\ \{f,g\}=2 \rightarrow v(f)=3, v(g)=2$이므로 임의로 하나 삭제
MWE의 개수가 $n-1=7$이므로 알고리즘 종료

표 3. G_1 **그래프의 MST 알고리즘 수행 횟수 비교**
Table 3. Running Time Comparsion of MST Algorithm of Graph G_1

알고리즘	수행 방법									수행 횟수
Borůvka 알고리즘	1단계 (선택-삭제)					2단계 (선택)				선택-삭제-선택 (3회)
	선택 (α)			삭제		α		γ		
	10 (5개 중복)			5		1		1		
Kruskal 알고리즘	$\|m\|$	α		β		γ		δ		15회
	15	6		1		1		7		
역-삭제 알고리즘	$\|m\|$	δ		γ		β		α		15회
	15	7		1		1		6		
선택-삭제 알고리즘	1단계 (선택-선택)					2단계 (삭제-삭제-삭제)				선택-선택-삭제-삭제 (4회)
	선택	S_1	S_2			D_1	D_2	D_3		
	9	α	α	β	γ ϵ	β	ϵ	-		
		5	1	1	1 1	1	1	-		
Prim 알고리즘	수행		대상			선택				8회
	8		59			7				

Ⅳ. 알고리즘 적용성 평가

본 장에서는 그림 5의 8개 그래프에 대해 Borůvka, Prim, Kruskal과 선택-삭제 알고리즘을 적용하여 MST를 얻을 수 있는지 살펴본다. G_2, G_7, G_8 그래프는 Chen[8]에서, G_3 그래프는 Wikipedia[9-12], G_4, G_5, G_6, G_9 그래프는 Peiper[7]에서 인용되었다.

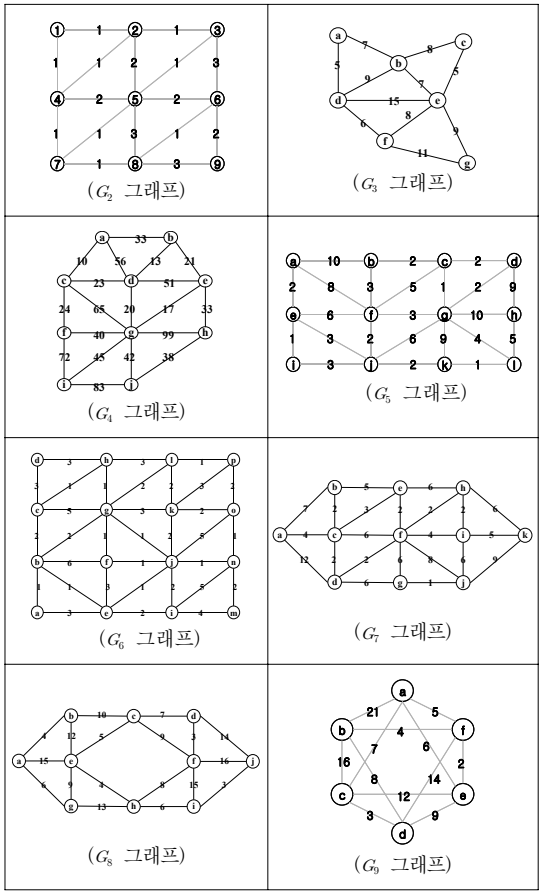


그림 5. 실험에 적용된 그래프
Fig. 5. Graphs Applied to Experiment

1. 적용 결과

모든 그래프의 Borůvka, Prim, Kruskal과 역-삭제 알고리즘의 수행 과정은 지면 관계상 기술하지 않는다. 또한, 역-삭제 알고리즘은 Kruskal과 동일한 결과를 나타내므로 생략한다. 8개의 그래프에 대해 제안된 선택-삭제 알고리즘의 선택과 삭제 후 얻은 MST 간선들과 더불어 알고리즘을 수행한 결과 얻은 MST를 그림 6 ~ 그림 13에 표현하였다. 선택-삭제 알고리즘도 8개 그래프 모두에서 MST를 얻음을 알 수 있다.

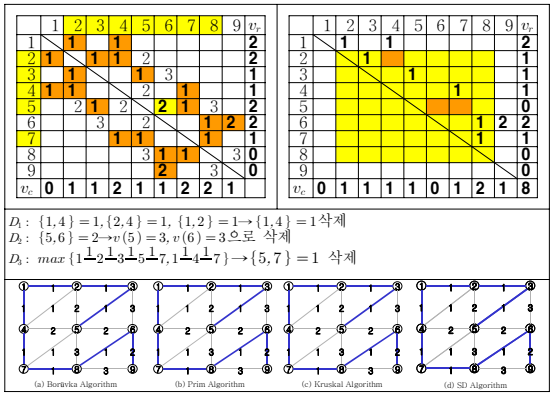


그림 6. G_2 그래프의 MST
Fig. 6. MST of Graph G_2

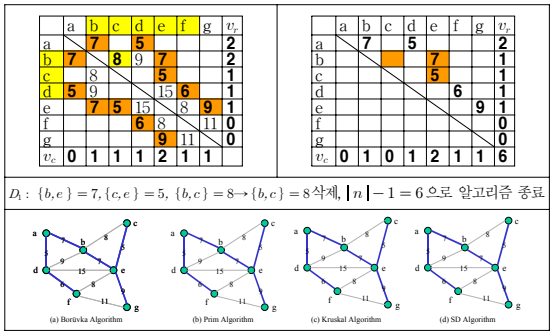


그림 7. G_3 그래프의 MST
Fig. 7. MST of Graph G_3

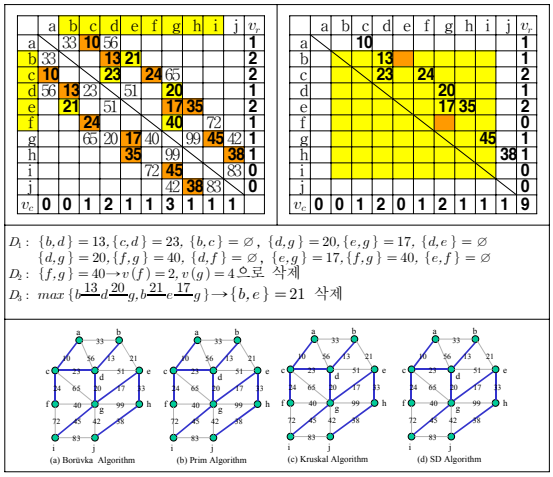


그림 8. G_4 그래프의 MST
Fig. 8. MST of Graph G_4

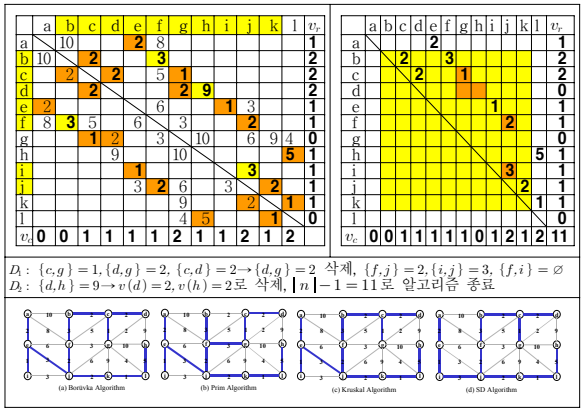


그림 9. G_5 그래프의 MST
Fig. 9. MST of Graph G_5

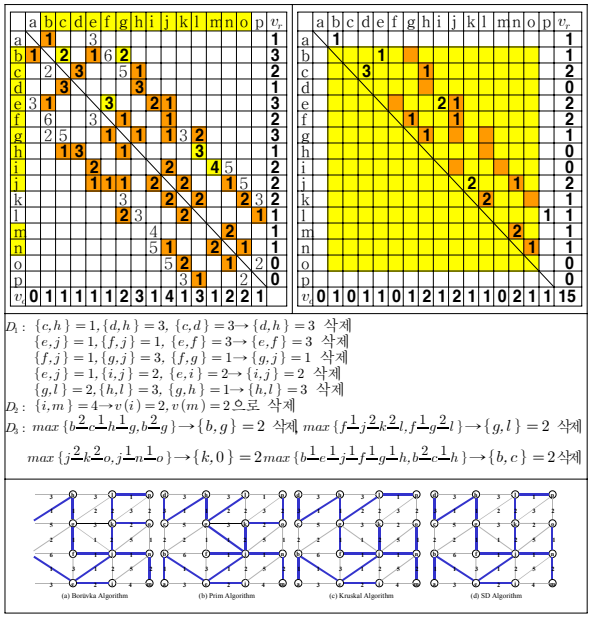


그림 10. G_6 그래프의 MST
Fig. 10. MST of Graph G_6

2. 적용 결과 분석

본 논문에 적용된 9개 그래프에 대한 알고리즘 수행 결과를 종합한 데이터는 표 4와 표 5에 제시하였다.

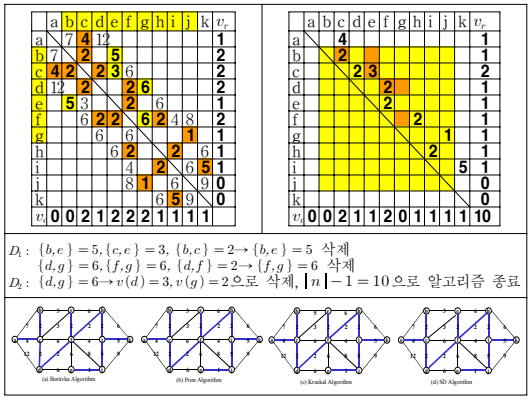


그림 11. G_7 그래프의 MST
Fig. 11. MST of Graph G_7

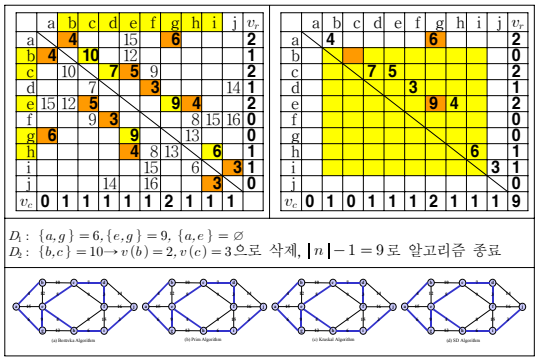


그림 12. G_8 그래프의 MST
Fig. 12. MST of Graph G_8

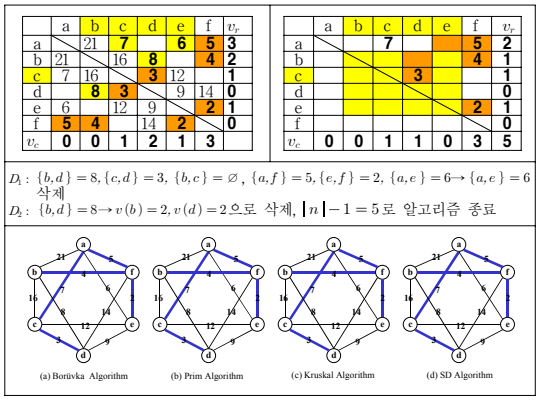


그림 13. G_9 그래프의 MST
Fig. 13. MST of Graph G_9

표 4. MST 알고리즘 비교
Table 4. Comparsion of MST Algorithms

그래프	$\|n\|$	$\|m\|$	Borůvka 알고리즘		Kruskal 알고리즘				선택-삭제 알고리즘			
			선택	삭제	선택	α	β	γ	δ	S_1+S_2	D_1	D_2	D_3
G_1	8	15	**10**	**5**	**2**	**6**	**1**	**1**	**7**	**9**	**1**	**1**	-
G_2	9	16	**19**	**11**	**0**	**3**	**5**	**5**	**3**	**11**	**1**	**1**	**1**
G_3	7	11	**8**	**2**	**0**	**5**	**3**	**1**	**2**	**7**	**1**	-	-
G_4	10	19	**10**	**4**	**3**	**4**	**4**	**5**	**6**	**12**	**0**	**1**	**1**
G_5	12	23	**14**	**5**	**2**	**6**	**5**	**5**	**7**	**12**	**1**	**1**	-
G_6	16	33	**29**	**17**	**3**	**7**	**8**	**8**	**10**	**24**	**5**	**1**	**3**
G_7	11	22	**16**	**7**	**1**	**7**	**4**	**4**	**7**	**13**	**2**	**1**	-
G_8	10	18	**10**	**4**	**3**	**8**	**2**	**1**	**7**	**10**	**0**	**1**	-
G_9	6	12	**6**	**2**	**1**	**4**	**1**	**1**	**6**	**7**	**1**	**1**	-

표 5. MST 알고리즘 수행 횟수 비교
Table 5. Running Time Comparsion of MST Algorithms

그래프	α	β	γ	δ	Kruskal 알고리즘		Borůvka 알고리즘		선택-삭제 알고리즘			
					최적 수행 횟수	실제 수행 횟수	선택	삭제	선택			삭제
									계	δ 감소	β 감소	
G_1	**6**	**1**	**1**	**7**	**8**	15	**12**	**5**	**9**	**6**	**0**	**2**
G_2	**3**	**5**	**5**	**3**	**13**	16	**19**	**11**	**11**	**3**	**2**	**3**
G_3	**5**	**3**	**1**	**2**	**9**	11	**8**	**2**	**7**	**2**	**1**	**1**
G_4	**4**	**4**	**5**	**6**	**14**	19	**13**	**4**	**12**	**6**	**1**	**2**
G_5	**6**	**5**	**5**	**7**	**16**	23	**16**	**5**	**12**	**7**	**4**	**2**
G_6	**7**	**8**	**8**	**10**	**23**	33	**32**	**17**	**24**	**9**	**0**	**9**
G_7	**7**	**4**	**4**	**7**	**15**	22	**17**	**7**	**13**	**7**	**2**	**3**
G_8	**8**	**2**	**1**	**7**	**11**	18	**13**	**4**	**10**	**7**	**1**	**1**
G_9	**4**	**1**	**1**	**6**	**6**	12	**7**	**2**	**7**	**5**	**0**	**2**

MST 알고리즘을 종료시키는 최적의 시점은 $|n|-1$을 찾는 시점으로 Kruskal 알고리즘의 $|\alpha+\beta+\gamma|$이다. Borůvka 알고리즘은 9개의 그래프 중 7개의 그래프에서 첫 번째 단계에서 간선을 $\{i,j\}$와 $\{j,i\}$를 중복 선택함으로 인해 α 개수를 너무 작게 선택함과 동시에 불필요한 간선을 삭제하며, 두 번째 단계에서 추가로 선택하는 단점이 있다. Kruskal 알고리즘은 $|\alpha+\beta+\gamma|$에서 알고리즘을 종료시킬 수 있으나 추가적으로 $|\delta|$ 만큼 알고리즘을 수행하는 단점이 있다. 반면에 제안된 선택-삭제 알고리즘은 G_1, G_6와 G_9 그래프만이 ϵ를 각각 1개씩 선택하였으며, G_2, G_3, G_4, G_7 그래프에서는 ϵ를 선택하지 않고 $\beta=\beta-2$개를, G_5 그래프에서는 $\beta=\beta-4$개, G_8 그래프에서는 $\beta=\beta-1$개만을 대상으로 선택하는 효과를 보였다. 선택-삭제 알고리즘의 삭제 과정을 Kruskal 알고리즘을 적용할 경우에도 최적 수행 횟수인 $|\alpha+\beta+\gamma|$와 비교하면 G_1, G_6와 G_9 그래프만 1회 더 수행하며, 나머지 6개 그래프는 모두 적게 수행하는 효과를 얻었다.

V. 결론 및 향후 연구과제

본 논문에서는 기존의 Borůvka, Prim, Kruskal과 역-삭제 알고리즘을 고찰하고, MST를 빠르게 얻기 위한 알고리즘 수행 횟수를 단축시키는 방법을 제안하였다. 먼저, 모든 정점들에 대해 Borůvka의 첫 번째 단계를 수행하고, 특정 정점에 대해 다시 수행하는 방법으로 모집단 간선의 수를 획기적으로 줄이는 효과를 얻었다. 이는 9개의 그래프에 제안된 알고리즘을 적용한 결과 Kruskal 알고리즘을 최적으로 수행하는 횟수인 간선의 수보다 6개의 그래프는 적은 결과를, 3개 그래프만이 1개가 큰 간선을 선택하여 최적으로 축소된 모집단 간선을 얻을 수 있었다. 다음으로 알고리즘 수행 횟수를 줄이기 위해 먼저, 3개 정점 간에 사이클이 발생하는 경우 최대 가중치 간선을 삭제하고, 다음으로 그래프의 최대 가중치 간선에 대해 결합가 기법을 적용하여 삭제한 후 마지막으로 결합가가 큰 정점 들 간의 사이클이 발생하는 경우 최대 가중치 간선을 삭제하는 방법을 적용하였다. 그 결과 1개 그래프는 1단계만, 5개 그래프는 2단계를, 나머지 3개 그래프는 3단계의 삭제 과정을 수행하여 알고리즘 수행 횟수를 최적으로 줄이는 효과를 얻었다.

제안된 선택-삭제 알고리즘의 삭제 과정은 다소 복잡한 단점을 갖고 있다. 따라서 역-삭제 알고리즘과 동일하게 삭제 과정을 보다 간단히 구현할 수 있는 알고리즘을 연구할 계획이다.

References

[1] Wikipedia, "Minimum Spanning Tree," 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] Wikipedia, "Reverse-Delete Algorithm," http://en.wikipedia.org/wiki/Reverse_delete_algorithm, Wikimedia Foundation, Inc., 2007.

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

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

[9] Wikipedia, "Prim's Algorithm," http://en.wikipedia.org/wiki/Prims_algorithm, Wikimedia Foundation, Inc., 2007.

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

[11] M. B. Choi, S. U. Lee, "A Prim Minimum Spanning Tree Algorithm for Directed Graph," IIBC, v.12 no.3, 2012.

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

저자 소개

최 명 복(종신회원)

- 2001년 : 아주대학교 컴퓨터공학과 (박사)
- 1997년~현재 : 강릉원주대학교 멀티미디어공학과 교수
- 2004년 1월~현재 : 한국인터넷방송통신학회 이사

<주관심분야 : 지능형 정보검색, 알고리즘 등>

- e-mail : cmb5859@gmail.com

이 상 운(정회원)

- 1998년~2001년 : 경상대학교 컴퓨터과학과 (박사)
- 2007년 3월~현재 : 강릉원주대학교 과학기술대학 멀티미디어공학과 부교수

<주관심분야 : 소프트웨어 프로젝트 관리, 알고리즘 등>

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