The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 14, No. 2, pp.35-42, Apr. 30, 2014. pISSN 2289-0238, eISSN 2289-0246

http://dx.doi.org/10.7236/JIIBC.2014.14.2.35
JIIBC 2014-2-5

한 사이클 내에서 최대 가중치 간선을 제거하기 위한 최소 신장트리 알고리즘

Minimum Spanning Tree Algorithm for Deletion of Maximum Weight Edge within a Cycle

최명복*, 한태용**, 이상운***

Myeong-Bok Choi*, Tae-Yong Han, Sang-Un Lee*****

요 약 본 논문은 최소신장트리를 쉽고 빠르게 구하는 방법을 제안하였다. 제안된 알고리즘은 먼저, 그래프의 가중치 간선의 수를 축소시키는 방법으로 그래프를 단순화 시켰다. 간선 수를 축소시키는 방법으로는 그래프 정점의 결합가가 3 이상인 경우, 최대 가중치 간선을 제거하는 방법을 적용하였다. 다음으로, 그래프를 단순화 시킨 축소된 모집단 간선들을 대상으로 사이클이 발생하는 부분을 확인하여 사이클 발생 간선들 중에서 최대 가중치를 갖는 간선을 삭제하는 방법을 적용하였다. 다양한 9개 그래프에 대해 제안된 사이클 최대 가중치 간선 제거 알고리즘을 적용한 결과 그래프의 사이클 개수만큼만 수행하여 MST를 쉽게 구하는 장점을 보였다. 모집단 축소 기법을 적용한 결과, 9개 그래프의 사이클 개수를 66%로 감소시키는 결과를 얻었으며, 최소 2개에서 최대 8개의 사이클에서의 최대 가중치 간선만 삭제하면 MST를 얻는 효과를 얻었다.

Abstract This paper suggests a method that obtains the minimum spanning tree (MST) far more easily and rapidly than the present ones. The suggested algorithm, firstly, simplifies a graph by means of reducing the number of edges of the graph. To achieve this, it applies a method of eliminating the maximum weight edge if the valency of vertices of the graph is equal to or more than 3. As a result of this step, we can obtain the reduced edge population. Next, it applies a method in which the maximum weight edge is eliminated within the cycle. On applying the suggested population minimizing and maximum weight edge deletion algorithms to 9 various graphs, as many as the number of cycles of the graph is executed and MST is easily obtained. It turns out to lessen 66% of the number of cycles and obtain the MST in at least 2 and at most 8 cycles by only deleting the maximum weight edges.

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

Ⅰ. 서 론

그래프 $G=(V,E)$는 정점들 (Vertices, v)과 간선들 (Edges, e)로 구성되어 있으며, 일반적으로 그래프란 모든 정점들을 연결하는 간선들이 무방향성 (Undirected)을 갖고 있는 무방향 그래프 (Undirected Graph)를 의미

*종신회원, 강릉원주대학교 멀티미디어공학과
**정회원, 강릉원주대학교 여성인력개발학과
***정회원, 강릉원주대학교 멀티미디어공학과
접수일자 : 2014년 2월 6일, 수정완료 : 2014년 3월 7일
게재확정일자 : 2014년 4월 11일

Received: 6 February, 2014 / Revised: 7 March, 2014 / Accepted: 11 April, 2014
*Corresponding Author: cmb5859@gmail.com
Dept. of Multimedia Engineering, Gangnung-Wonju National University, Korea

한다. 무방향 그래프의 간선들이 가중치 (Weight)를 갖고 있는 경우 이를 가중 그래프라 한다. 가중 그래프에서 모든 정점들이 간선들로 연결되어 있고, 사이클이 발생하지 않으면서 간선들의 가중치 합 ($\Sigma w(e)$)이 최소가 되는 트리를 최소신장트리 (Minimum Spanning Tree, MST)라 한다.[1] 즉, 그래프에서 사이클이 발생하지 않으면서 모든 정점들을 최소 가중치 합의 간선들로 연결된 MST는 $|e|=|v|-1$가 성립한다. MST는 전기, 전화, 가스, 수도, 컴퓨터, 도로 등 망 설계 (Network Design) 분야에 활용되고 있다.[2,3]

MST를 찾는 대표적인 알고리즘으로 Borůvka[4,5], Prim[6], Kruskal[7]과 역-삭제 (Reverse-Delete) 알고리즘[8]이 있다. 이들 알고리즘은 모든 간선들의 가중치가 서로 다르다(Distinct)는 가정에 기반하고 있다. 그러나 현실적으로 동일한 가중치 값을 가진 간선들이 다수 포함될 수 있다. 이들 알고리즘은 모두 욕심쟁이 알고리즘 (Greedy Algorithm)으로 한번에 현 시점에서 최적이라고 판단되는 하나의 최소 가중치 간선을 선택하는 방식을 채택하고 있다. 또한 알고리즘 수행 시간 (Running Time)은 $O(|e|\cdot log|e|)$, $O(|e|+|v|\cdot log|v|)$, $O(|e|\cdot log|v|)$로 별 차이가 없어 우열을 판가름하기 어렵다.

Borůvka 알고리즘[4,5]은 MST를 찾는 알고리즘으로 1926년에 처음 제안되었으며, 현재는 1950년대에 제안된 Prim과 Kruskal 알고리즘이 널리 사용되고 있다. 역-삭제 알고리즘은 Kruskal 알고리즘을 역으로 수행하는 방식으로 현재에는 일반적으로 사용되지 않고 있다.[1]

Borůvka, Prim과 Kruskal 알고리즘은 가산 모델 (Additive Model)로 최소 가중치 간선을 한번에 하나씩 선택하여 그래프에 추가하는 구성 방식 (Construction Method)이다. 반면에 역-삭제 알고리즘은 감산 모델 (Subtractive Model)로 최대 가중치 간선을 한번에 하나씩 선택하여 그래프에서 제거하는 파괴 방식 (Destruction Method)이다. Borůvka와 역-삭제 알고리즘은 그래프를 시각적으로 찾는 그래픽 방법 (Graphical Method)이 보다 이해가 쉽다. 반면에 Prim과 Kruskal 알고리즘은 프로그램을 구현하는 계산방법 (Computational Method)으로 찾는 것이 보다 쉽다.

MST는 모집단 $|e|$에서 사이클이 발생하지 않도록 가중치가 적은 $|v|-1$개를 찾는 것으로, 결국, MST의 $|v|-1$ 간선은 그래프에서 사이클이 발생한 곳에서 최대 가중치 간선을 제거하면 쉽게 얻을 수 있다. 본 논문은 이러한 개념에 기반하여 사이클의 최대 가중치 간선을 제거하는 MST 알고리즘을 제안한다. 제안된 알고리즘은 프로그램으로 구현하기 보다는 시각적으로 보다 쉽게 구할 수 있기 때문에 이 방법을 적용한다. 실제 그래프는 사이클을 다수 포함하고 있어 육안으로 모든 사이클을 판단하는데 실패할 수 있는 복잡성을 갖고 있다. 따라서 그래프에서 MST에 기여하지 못하는 최대 가중치 간선들을 사전에 삭제하는 전처리 과정을 거친 축소된 모집단 간선을 대상으로 하는 사이클 최대 가중치 간선 제거 알고리즘을 제안한다.

2장에서는 그래픽으로 쉽게 MST를 구하는 Borůvka와 역-삭제 알고리즘 수행 방법을 고찰해보고, 실제 그래프에 적용하여 MST를 찾고 문제점을 고찰해 본다. 3장에서는 MST 알고리즘의 수행 횟수를 단축시키는 방법을 제시하고, 사이클에서 최대 가중치 간선을 제거하는 간단한 알고리즘을 제안한다. 4장에서는 실제 그래프들을 대상으로 제안된 알고리즘의 효율성을 검증해 본다.

II. 관련 연구와 연구 배경

1. MST 알고리즘과 적용

본 절에서는 그래픽 방법으로 MST를 쉽게 구할 수 있는 Borůvka[4,5]와 역-삭제 알고리즘[8]을 적용하는 방법을 고찰한다.

그래프 $G=(V,E)$에서 간선은 무방향성이므로 $\{i,j\}$로 표기한다.[9] 또한 간선의 가중치 $w\{i,j\}$를 편의상 $\{i,j\}$로 표기한다. 그래프의 정점 개수를 $|v|$, 간선 개수를 $|e|$라 하자. 신장트리는 $|v|$ 개의 정점을 사이클이 발생하지 않게 간선들로 모두 연결하는 경우이므로 간선의 수는 $|v|-1$ 개로 구성되어 있다.

Borůvka 알고리즘은 다음과 같이 수행된다.[4,5]

(1^{st} Stage) 그래프의 각 정점 i에서 중복에 무관하게 최소 가중치 간선을 선택하여 간선들을 연결한 부분 신장트리 (Partial Spanning Tree, PSP)를 형성한다. 이 과정에서 사이클 (루프)이 발생하였을 경우 최대 가중치 간선을 삭제한다.

(2^{nd} Stage) 만약, PSP가 1개이면 알고리즘을 종료한다. 그렇지 않으면 PSP 들 간을 연결하는 최소 가중치 간선을 선택한다.

역-삭제 알고리즘[8]은 Kruskal 알고리즘을 역으로 수행하는 방식으로 다음과 같이 수행된다.

(1) T는 모든 정점들과 간선들로 구성되어 있다.

(2) E에 있는 모든 가중치를 갖는 간선들을 내림차순으로 정렬시킨다.

(3) 최대 가중치를 갖는 임의의 간선을 한번에 하나씩 선택하여 해당 간선을 T에서 삭제하였을 때 T의 임의의 정점을 분리 (Disconnect)시키지 않으면 삭제한다.

(4) E에서 더 이상 선택할 간선이 없을 때까지 (3)을 반복적으로 수행하고 알고리즘을 종료한다.

그림 1의 G_1 그래프를 대상으로 Borůvka와 역-삭제 알고리즘을 구해 보자. G_1 그래프는 Wikipedia[1]에서 인용되었다.

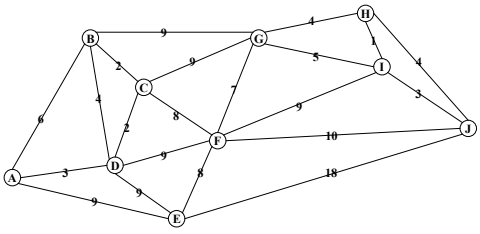


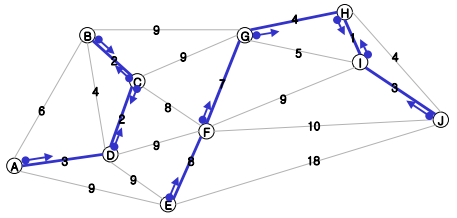
그림 1. G_1 그래프
Fig. 1. Graph G_1

그래픽 방법으로 MST를 찾는 Borůvka와 역-삭제 알고리즘을 G_1 그래프에 적용한 결과는 그림 2에 제시하였다.

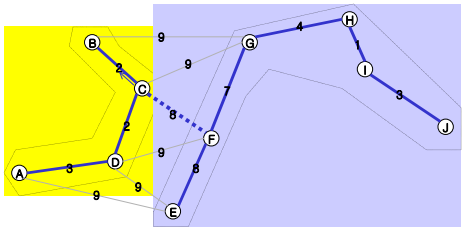
2. MST 알고리즘의 문제점과 연구 배경

MST는 그래프 속성 (Property)들 중 절단 속성 (Cut Property)와 사이클 속성 (Cycle Property)를 활용한다.[2,3] S를 정점들의 부분집합, $e=\{i,j\}$를 정확히 하나의 끝단 (Endpoint) i 또는 j가 S에 존재하는 최소 가중치 간선이라 할 때 MST는 e를 포함한다는 속성이 절단 속성이다. 반면에, C를 임의의 사이클, f를 C에 포함된 최대 가중치 간선이라 할 때 MST는 f를 포함하지 않는다는 속성이 사이클 속성이다. Borůvka, Prim, Kruskal 알고리즘은 절단속성을, 역-삭제 알고리즘은 절단 속성과 사이클 속성을 적용하고 있다.[2]

Borůvka 알고리즘은 1st Stage에서 각 정점에서의 최소 가중치 간선을 선택하였을 경우 사이클이 발생할 수 있다. 따라서 사이클을 제거하기 위해 사이클에서의 최대 가중치 간선을 삭제하는 알고리즘이 추가로 요구된다. 2nd Stage에서 PSP가 여러 개 발생할 경우 이들 상호간에 최소 가중치 간선을 선택하는 복잡한 알고리즘을 구현해야만 한다. 사이클 확인 방법은 Kruskal 알고리즘을 적용하면 쉽게 해결할 수 있다.

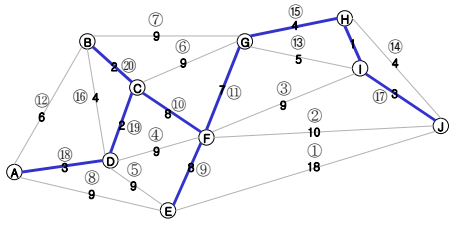


(1st Stage)



(2nd Stage)

(a) Borůvka 알고리즘



수행 순서	간선 내림차순 정렬	그래프 정점 분리	정점 분리 여부 확인 (사이클)	남은 간선 수
1	{E,J}=18	X	{E,J}, {J,F}, {F,E}	20
2	{F,J}=10	X	{F,J}, {J,I}, {I,F}	19
3	{F,I}=9	X	{F,I}, {I,G}, {G,F}	18
4	{D,F}=9	X	{D,F}, {F,C}, {C,D}	17
5	{D,E}=9	X	{D,E}, {E,F}, {F,C}, {C,D}	16
6	{C,G}=9	X	{C,G}, {G,F}, {F,C}	15
7	{B,G}=9	X	{B,G}, {G,F}, {F,C}, {C,B}	14
8	{A,E}=9	X	{A,E}, {E,F}, {F,C}, {C,D}, {D,A}	13
9	**{E,F}=8**	O	-	13
10	**{C,F}=8**	O	-	13
11	**{F,G}=7**	O	-	13
12	{A,B}=6	X	{A,B}, {B,D}, {D,A}	12
13	{G,I}=5	X	{G,I}, {I,H}, {H,G}	11
14	{H,J}=4	X	{H,J}, {J,I}, {I,H}	10
15	**{G,H}=4**	O	-	10
16	{B,D}=4	X	{B,D}, {D,C}, {C,B}	**9**
17	**{I,J}=3**	O	-	
18	**{A,D}=3**	O	-	
19	**{C,D}=2**	O	-	
20	**{B,C}=2**	O	-	
21	**{H,I}=1**	O	-	

(b) 역-삭제 알고리즘

그림 2. G_1 그래프의 Borůvka와 역-삭제 알고리즘 적용
Fig. 2. Application of Borůvka and reverse-delete algorithm of graph G_1

역-삭제 알고리즘은 그래프의 가중치 간선을 내림차순으로 정렬시키고 최대 가중치 간선부터 삭제 여부를 결정한다. 이는 비교 정렬 (Comparison Sort)에 $O(|e| \cdot log|e|)$의 수행 시간이 필요하다. 또한, 해당 간선을 제거하였을 경우 특정 정점을 그래프에서 분리시키는지 확인하기 위해서는 깊이우선탐색 (Depth-First Search, DFS)로 그래프의 사이클을 확인하는 $O(|v|+|e|)$의 시간이 필요하다.[8]

Borůvka, Prim, Kruskal 알고리즘에서 사이클이 발생하지 않는 최소 가중치 간선 선택 방법이나, 역-삭제 알고리즘에서 "임의의 정점을 그래프에서 분리시키지 않는 최대 가중치를 갖는 간선을 삭제하는 방법으로 얻은 MST의 공통점은 선택되지 않거나 삭제되는 간선은 그래프에서 사이클을 형성하는 최대 가중치 간선이 됨을 알 수 있다. 예로, 역-삭제 알고리즘으로 MST를 얻는 그림 2의 (b)를 재해석하면 그림 3과 같이 12개의 사이클이 존재함을 알 수 있다. 이 12개 사이클을 랜덤하게 탐색하여 최대 가중치 간선을 삭제하면 알고리즘을 12회 수행하면 종료되나 역삭제 알고리즘을 적용하여 간선들을 내림차순으로 정렬시키고 역-삭제 알고리즘으로 수행하면 16회를 수행해야만 한다. 따라서 불필요하게 추가적으로 4회 더 수행하는 결과를 나타낸다. 결국, 12개의 사이클을 랜덤하게 찾는 방법을 적용하면 간선들을 내림차순 정렬에 필요한 수행시간을 단축시킬 수 있다.

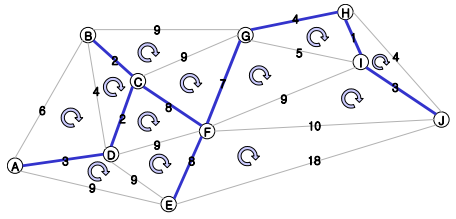


그림 3. 사이클 최대 가중치 간선 삭제로 MST를 쉽게 찾는 방법
Fig. 3. Method that finds MST based on deletion of maximum weight edge within the cycle

즉, MST의 사이클 속성만을 적용하여 그래프에서 사이클이 발생하는 부분에서의 최대 가중치 간선을 삭제하면 MST를 쉽게 얻을 수 있다.

또한, 그래프의 간선 $|e|$개에서 MST에 기여하는 간선 $|v|-1$를 선택하기 위해 기존의 MST 알고리즘은 모두 모집단을 $|e|$개 사용한다는 공통점이 있다. 이로 인해 그래프에서 선택되지 않거나 삭제될 최대 가중치 간선의 수가 너무 많아 MST 얻는데 시간이 많이 소요될 뿐 아니라 잘못된 결과를 얻을 수 있다. 따라서 전체 간선들을 모집단으로 하지 않고 MST에 기여하지 못하는 간선을 최대한 삭제하여 모집단을 축소하는 방법이 요구된다.

실제로 주어진 그래프의 간선 전체를 모집단으로 하여 MST를 찾고자 할 때 직면하는 문제점은 그래프가 복잡하게 연결되어 있어 육안으로 사이클이 몇 개가 존재하는지 확인하는데 어려움이 있을 수 있다. 또한, 사이클 개수만큼 간선을 삭제하지 않아도 MST를 얻을 수 있다. 예로 그림 4의 완전 그래프 (Complete Graph)를 살펴보면 정점의 개수 $|v|$는 6, 간선의 개수 $|e|$는 12이다. 이 그래프의 사이클은 12개가 존재한다. 이 그래프의 MST는 $|e|=|v|-1=5$ 개로 구성되어야 하나 사이클 12개의 최대 가중치 간선을 모두 제거하면 간선은 하나도 없는 잘못된 결과를 얻는다.

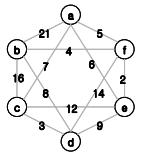


그림 4. 완전 그래프의 사이클 개수
Fig. 4. Cycle number of complete graph

결국, 원래 그래프의 사이클 개수를 쉽게 파악하기 위해서는 모집단 크기를 축소시킬 필요가 있다.

Ⅲ. 사이클 최대 가중치 간선 제거 MST 알고리즘

본 장에서 제안하는 MST 알고리즘은 모집단의 크기를 축소시키고, 축소된 사이클 개수에서 최대 가중치 간선들을 제거하는 단순한 방법이다.

1. 그래프의 간선 모집단 축소

본 장에서는 결합가 (Valency, δ) 개념[10]을 도입하여 불필요한 간선을 제거하는 모집단 축소 기법을 제안한다. 이를 전처리 (Preprocessing)라 하며, 모집단의 대상을 축소하는 기법이라 하자. 결합가는 어느 한 정점 i에 연

결된 간선의 수로 계산된다. 예로, 그림 1의 G_1 그래프에서 정점 A는 $\delta(3)$, 정점 D는 $\delta(5)$, 정점 C는 $\delta(4)$이다.

모집단의 대상을 축소하는 전처리 과정은 다음과 같이 수행된다.

(1) 그래프에 대한 가중치 간선 정방행렬을 작성한다.

(2) 가중치 간선들 정방행렬에서 결합가 $\delta(i) \geq 3$인 행 정점 i에 대해 최대 가중치 간선 $e_{\max}$ $\{i,j\}$를 선택한다. (동일한 가중치는 모두 선택한다.)

	A	B	C	D	E	F	G	H	I	J	$\delta(i)$
A		6		3	9						3
B	6		2	4			9				4
C		2		2		8	9				4
D	3	4	2		9	9					5
E	9			9		8				18	4
F			8	9	8		7		9	10	6
G		9	9			7		4	5		5
H							4		1	4	3
I						9	5	1		3	4
J					18	10		4	3		4

(3) $\delta(i) - |e_{\max}| \geq 2$이면 $e_{\max}$인 최대 가중치 간선들 $\{i,j\}$를 삭제한다.

	A	B	C	D	E	F	G	H	I	J	$\delta(i)$	$\delta(i) - \lvert e_{\max} \rvert$
A		6		3	9						3	3-1 = 2
B	6		2	4			9				4	4-1 = 3
C		2		2		8	9				4	4-1 = 3
D	3	4	2		9	9					5	5-2 = 3
E	9			9		8				18	4	4-1 = 3
F			8	9	8		7		9	10	6	6-1 = 5
G		9	9			7		4	5		5	5-2 = 3
H							4		1	4	3	3-2 = 1
I						9	5	1		3	4	4-1 = 3
J					18	10		4	3		4	4-1 = 3

(4) $\{i,j\} = \{j,i\}$이므로 $\{j,i\}$를 삭제한다. 만약, 이 과정에서 특정 정점 i의 $\delta(i) = 0$이 되면 정점 i의 최소 가중치 간선은 삭제하지 않는다.

	A	B	C	D	E	F	G	H	I	J
A		6		3						
B	6		2	4						
C		2		2		8				
D	3	4	2							
E						8				
F			8		8		7			
G						7		4	5	
H							4		1	4
I							5	1		3
J								4	3	

이와 같이 전처리 과정을 거친 결과 모집단인 간선의 수는 상삼각행렬에서 보면 기존의 21개에서 13개로 축소됨을 알 수 있다.

2. 사이클 최대 가중치 간선 제거 MST 알고리즘

3.1절에서 전처리 과정으로 획득한 13개의 간선을 모집단으로 그래프를 표현하면 12개의 사이클이 4개의 사이클로 감소함을 알 수 있다. 즉, 4개 사이클에서 최대 가중치 간선 4개를 삭제하면 $|v|-1=9$ 개의 간선으로 구성된 MST를 얻을 수 있다. 축소된 모집단을 대상으로 사이클 최대 가중치 간선을 제거하는 MST 알고리즘은 다음과 같이 수행된다.

(1) 전처리로 얻어진 모집단 간선들을 대상으로 상삼각행렬 간선들만으로 그래프를 구성한다.

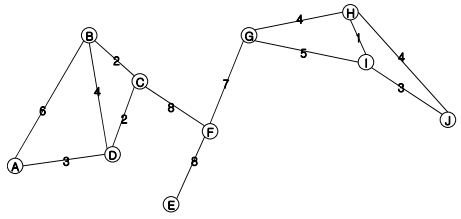


그림 5. 전처리 후의 그래프
Fig. 5. Graph after Preprocessing

(2) 그래프에서 사이클이 발생하는 부분을 DFS로 랜덤하게 모두 확인하고 해당 사이클에서 최대 가중치 간선을 삭제한다. 만약, 동일한 값이 다수 존재 시 하나만 삭제한다.)

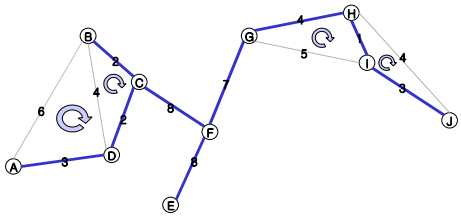


그림 6. 사이클 점검 후의 그래프
Fig. 6. Graph after Cycle Checking

참고로, 전처리 과정을 거친 모집단을 대상으로 Borůvka 알고리즘을 구현하면 다음과 같다.

(1) 전처리로 얻어진 모집단 간선들을 대상으로 정방행렬의 각 행 정점에 대해 최소 가중치 간선 $e_{\min}$을 선택한다.

	A	B	C	D	E	F	G	H	I	J
A		6		3						
B	6		2	4						
C		2		2		8				
D	3	4	2							
E						8				
F			8		8		7			
G						7		4	5	
H							4		1	4
I							5	1		3
J								4	3	

(2) 선택된 간선들을 연결하여 부분신장트리를 구성한다.

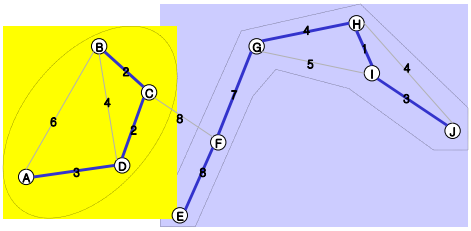


그림 7. 부분 신장 트리 구성
Fig. 7. Sub Spanning Tree Construction

(3) 두 부분 신장트리를 연결하는 최소 가중치 간선을 선택하여 하나의 신장트리를 구성한다. $\{C,F\}=8$

3. 알고리즘 성능 분석

본 절에서는 G_1 그래프에 대해 전처리 과정을 거치지 않은 전체 간선을 대상으로 하는 방법과 전처리 과정을 거쳐 모집단을 축소한 상태에서 MST를 얻는 방법의 성능을 비교하여 표 1에 제시하였다.

표 1. G_1 그래프의 MST 알고리즘의 수행 횟수 비교
Table 1. Running time comparison of MST algorithm of graph G_1

알고리즘	그래프의 간선 전체 모집단 이용	전처리 과정을 거친 축소된 모집단 이용
Borůvka 알고리즘	1st Stage : 21개중 8개 선택 2nd Stage : 6개중 1개 선택 **(27회 수행)**	1st Stage : 13개중 8개 선택 2nd Stage : 1개중 1개 선택 **(14회 수행)**
역-삭제 알고리즘 (내림차순 삭제)	알고리즘 종료 : 전체 간선 대상 **21회 수행** 알고리즘 종료 : $\lvert e_{mst}\rvert=\lvert v\rvert-1$ **16회 수행**	알고리즘 종료 : 전체 간선 **13회 수행** 알고리즘 종료: $\lvert e_{mst}\rvert=\lvert v\rvert-1$ 8회 수행
제안 알고리즘 (랜덤 삭제)	12회 수행	**4회 수행**

제안된 알고리즘은 단지 4회 수행으로 MST를 얻을 수 있었다. 이 결과를 Borůvka 알고리즘과 비교하면 그래프의 전체 간선을 모집단으로 할 경우보다는 85.2% 감소, 전처리 과정으로 축소된 모집단을 대상으로 하는 경우에 72.4% 감소 효과를 얻었다. 역-삭제 방법과 비교시 전체 모집단을 대상으로 하는 방법은 80%와 75% 감소 효과를, 축소된 모집단을 대상으로 하는 방법은 69%와 50%의 감소 효과를 얻을 수 있었다. 결국, 제안된 알고리즘이 가장 쉽고 빠르며 정확하게 MST를 얻을 수 있음을 알 수 있다.

Ⅳ. 알고리즘 적용성 평가

본 장에서는 제안된 알고리즘의 적용성을 평가하기 위해 그림 8의 다양한 그래프에 적용 가능한지 평가하여 본다. G_2는 Wikipedia[13-15], G_3, G_4, G_6, G_8, G_9은 Peiper[12]에서, G_5, G_7는 Chen[11]에서 인용되었다.

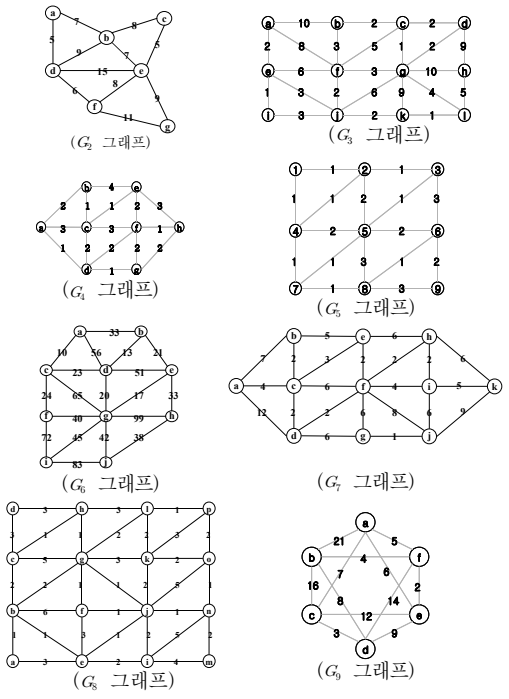


그림 8. 알고리즘 평가에 사용된 그래프
Fig. 8. Graphs used for algorithm evaluation

그림 8의 8개 그래프에 대해 알고리즘을 수행하는 과정은 생략한다. 알고리즘 수행 과정을 종합한 결과는 표 2와 표 3에 제시하였다. 9개 그래프에서 축소된 모집단을 적용할 경우 평균적으로 볼 때 원 그래프의 간선 수를 38%, 사이클 개수를 66% 감소시키는 결과를 얻을 수 있

었다. 역-삭제 알고리즘과 비교한 결과, 전체 모집단을 대상으로 할 경우 평균 47%, 축소된 모집단을 대상으로 할 경우 평균 43%의 수행횟수를 단축시키는 효과를 나타내었다.

결국, 전체 간선을 모집단으로 하는 경우와 축소된 모집단을 대상으로 하는 경우 모두 제안된 알고리즘은 그래프의 사이클 개수만큼만 수행하기 때문에 가장 쉽고, 빠르게 MST를 얻음을 알 수 있다.

표 2. 전체 모집단 대상 MST 알고리즘의 수행 횟수 비교
Table 2. Running time comparison of MST algorithm of the whole of population

그래프	$\|v\|$	$\|e\|$	Borůvka 알고리즘 (대상/선택/PSP 수/사이클)		역-삭제 알고리즘 (대상/수행/사이클)	사이클 최대 가중치 간선 제거 알고리즘 (대상/수행/사이클)
			1st Stage	2nd Stage		
G_1	10	**21**	21/ 8/1/0	6/1/1/0	21/21/12	**21/12/12**
G_2	7	**11**	11/ 6/0/0	-	11/11/ 5	**11/ 5/ 5**
G_3	12	**23**	23/10/3/1	5/1/2/0 8/1/1/0	23/23/12	**23/12/12**
G_4	8	**15**	15/ 5/3/0	6/1/2/0 6/1/1/0	15/15/ 8	**15/ 8/ 8**
G_5	9	**16**	16/10/0/2	-	16/16/ 8	**16/ 8/ 8**
G_6	10	**19**	19/ 7/3/0	3/1/2/0 6/1/1/0	19/19/10	**19/10/10**
G_7	11	**22**	22/ 9/2/0	5/1/1/0	22/22/12	**22/12/12**
G_8	16	**33**	33/19/0/4	-	33/33/18	**33/18/18**
G_9	6	**12**	12/ 4/2/0	6/1/1/0	12/12/ 7	**12/ 7/ 7**

표 3. 축소된 모집단 대상 MST 알고리즘의 수행 횟수 비교
Table 3. Running time comparison of MST algorithm of the reduced population

그래프	$\|v\|$	$\|e\|$	Borůvka 알고리즘 (대상/선택/PSP 수/사이클)		역-삭제 알고리즘 (대상/수행/사이클)	사이클 최대 가중치 간선 제거 알고리즘 (대상/수행/사이클)
			1st Stage	2nd Stage		
G_1	10	**13**	13/ 8/2/0	1/1/1/0	13/ 8/4	**13/4/4**
G_2	7	**8**	8/ 6/1/0	-	8/ 3/2	**8/2/2**
G_3	12	**14**	14/10/3/1	2/1/2/0 3/1/1/0	14/ 7/4	**14/4/4**
G_4	8	**9**	9/ 5/3/0	2/1/2/0 2/1/1/0	9/ 2/2	**9/2/2**
G_5	9	**11**	11/10/1/2	-	11/ 8/3	**11/2/3**
G_6	10	**12**	12/ 7/3/0	2/1/2/0 2/1/1/0	12/ 8/3	**12/3/3**
G_7	11	**14**	14/ 9/2/0	2/1/1/0	14/ 7/4	**14/4/4**
G_8	16	**23**	23/14/3/1	4/1/2/0 2/1/1/0	23/17/8	**23/8/8**
G_9	6	**8**	8/ 4/2/0	3/1/1/0	8/ 4/3	**8/3/3**

V. 결 론

본 논문은 최소신장트리를 쉽게 구하기 위해 먼저 그래프의 간선 모집단의 크기를 줄이는 방법을 제안하였다. 모집단의 크기를 축소하는 방법은 정점의 결합가 개념을 적용하였다. 다음으로 축소된 모집단 간선을 그래프로 표현하여 사이클을 확인하여 사이클에서 최대 가중치 간선을 제거하여 쉽게 MST를 구하는 방법을 제안하였다. 제안된 알고리즘은 Borůvka와 역-삭제 알고리즘의 공통점을 도출한 결과 Borůvka 알고리즘에서 선택되지 않거나 역-삭제 알고리즘에서 삭제되는 간선이 그래프의 사이클 최대 가중치 간선이 된다는 사실에 기반하였다. 즉, Borůvka, Prim, Kruskal 알고리즘은 MST의 절단 속성을, 역-삭제 알고리즘은 절단속성과 사이클 속성을 모두 적용한 반면 제안된 알고리즘은 사이클 속성만을 적용하였다.

9개의 그래프에 제안된 MST 알고리즘을 적용한 결과 최소 2개에서 최대 8개의 사이클만 확인하여 최대 가중치 간선을 제거하면 MST를 구할 수 있는 가장 간단하면서도 쉬운 방법임을 증명하였다.

본 제안 모델은 시각적으로 MST를 쉽게 구할 수 있다. 이를 프로그램으로 구현하면 사이클을 확인하는데 $O(|v|+|e|)$의 시간이 필요하며, 이는 Borůvka, Prim, Kruskal 알고리즘의 수행시간 $O(|e|\cdot log|e|)$ 보다 큰 단점이 있다. 따라서 추후 그래프의 사이클을 보다 빠르게 확인하는 구현 방법을 연구하고자 한다.

References

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

[2] P. Bille, "Advanced Algorithms - Greedy Algorithms," IT-Universiteter i København, http://www.itu.dk/courses/SAA/E2007/4.pdf, 2007.

[3] S. Parthasarathy, "CMSC 451: Design and Analysis of Computer Algorithms, Department of Computer Science, University of Maryland, http://www.cs.umd.edu/class/sum2005/cmsc451/mst.pdf, 2005.

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

[5] 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.

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

[7] 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.

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

[9] Wikipedia, "Graph (mathematics)," http://en.wikipedia.org/wiki/Graph_(mathematics), Wikimedia Foundation, Inc., 2007.

[10] Wikipedia, "Glossary of Graph Theory," http://en.wikipedia.org/wiki/Glossary_of_graph_theory, Wikimedia Foundation, Inc., 2007.

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

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

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

[14] 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.

[15] 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)

- 1997~현재 : 강릉원주대학교 멀티미디어공학과 교수
- 2004. 1~현재 : 한국인터넷방송통신학회 이사
- 주관심분야 : 지능형 정보검색, 소프트웨어 공학, 알고리즘
- e-mail : cmb5859@gmail.com

한 태 용(정회원)

- 강릉원주대학교 문화대학 여성인력개발학과 교수
- 관심분야 : 컴퓨터 경영분야
- e-mail : tyhan@gwnu.ac.kr

이 상 운(정회원)

- 2007년 3월 ~ 현재 : 강릉원주대학교 멀티미디어공학과 부교수
- 주관심분야 : 소프트웨어 척도, 분석과 설계 방법론, 소프트웨어 신뢰성, 그래프 알고리즘
- e-mail : sulee@gwnu.ac.kr