너비 우선 신장 트리 갱신문제를 위한 분산알고리즘

박 정 호† · 박 윤 용† · 황 석 형†

요 약

너비 우선 신장 트리가 이미 구성되어 있는 비동기식 네트워크상에서 네트워크 형상이 변할 경우, 이로인해 구성되어 있던 너비 우선 신장 트리를 갱신해야 하는 경우가 발생한다. 본 논문에서는 이러한 경우 너비 우선 신장 트리를 효율적으로 갱신하는 메시지 복잡도와 이상시간 복잡도 모두 $O(p\sqrt{q}+q+a+n')$인 분산알고리즘을 제안한다. 여기서, a는 추가 링크의 수, n'는 네트워크의 토폴로지가 변한후의 네트워크상에 존재하는 노드수를 각각 나타낸다. 그리고, p는 삭제 또는 추가 링크를 가진 이중결합요소에 속하는 전체 노드 수를 나타내며, q는 삭제 또는 추가 링크를 가진 이중결합요소에 속하는 전체 링크수를 나타낸다.

An Efficient Distributed Algorithm to Solve Breadth-First Spanning Tree Updating Problem

Jung-Ho Park† · Yoon-Young Park† · Suk-Hyung Hwang†

ABSTRACT

Consider the problem to update Breadth-First Spanning Tree in response to topology change of the network. This paper proposes an efficient distributed algorithm that solves such a problem after several processors and links are added and deleted. Its message complexity and its ideal-time complexity are $O(p\sqrt{q}+q+a+n')$ respectively, where n' is the number of processors in the network after the topology change, a is the number of added links, p is the total number of links in the biconnected component (of the network before the topology change) including the deleted links or added links, and q is the total number of processors in the biconnected component including the deleted links or added links.

1. 서 론

네트워크상에서 프로세서와 링크의 추가 및 삭제가 빈번히 발생함으로 인해 네트워크 형상은 일정하지 않고 동적으로 변화하는 것이라고 생각할 수 있다. 동적으로 네트워크 형상이 변화하는 네트워크 환경에 있어서는 그 토폴로지(topology) 정보를 하나의 프로세서가 통괄해서 일괄적으로 관리하는 것보다 각 프로세서가 자기에 관한 토폴로지 정보만을 분담해서 관리하는 것이 좋다. 이와같이 어떤 문제를 해결하는데 필요한 정보가 네트워크상의 프로세서에 분산되어 있는 상황에서 그들 정보를 교환하면서 그 문제를 해결하는 알고리즘을 분산 알고리즘(Distributed Algorithm)이라고 한다 [1-9].

네트워크상에서 임의의 프로세서를 중심으로 하여 다른 모든 프로세서와의 너비 우선 신장 트리(Breadth-First Spanning Tree)를 구하는 것은 효율적으로 메시지를 전송하는 등의 경우에 있어서 중요한 문제이다.

네트워크의 토폴로지는 동적으로 변화하기 때문에, 너비 우선 신장 트리를 포함하여 각종 문제를 해결했다고 하더라도 네트워크의 변화에 따라 이들 문제를

※ 본 연구는 정보통신부에서 시행하는 대학기초연구 지원사업에서 일부를 지원받았음.
† 종신회원 : 선문대학교 컴퓨터정보학부 교수
논문접수 : 2000년 2월 25일, 심사완료 : 2000년 4월 17일

다시 해결해야 하는 경우가 발생한다. 이와같이 네트워크의 변화에 따라 다시 해결하는 문제를 갱신문제(Updating Problem)라고 한다. 즉, 임의의 네트워크상에서 임의의 프로세서에 대한 너비 우선 신장 트리가 구성되어 있는 상황에서 네트워크가 변화했을 때, 너비 우선 신장 트리를 다시 구성하는 문제를 너비 우선 신장 트리 갱신문제라고 한다.

갱신문제는 네트워크의 변화에 따라 문제를 다시 해결한다는 점을 고려하지 않고, 변화후의 새로운 네트워크에 대해 처음부터 그 문제를 해결하는 기존의 알고리즘을 적용시키더라도 해결할 수 있다. 즉, 문헌 [7]에 제안된 알고리즘을 새로운 네트워크에 적용할 경우, 너비 우선 신장 트리 갱신문제를 해결할 수 있다. 그러나, 갱신문제의 경우, 각 프로세서는 토폴로지 변화전의 네트워크에 대한 답을 가지고 있으므로, 각 프로세서가 가지고 있는 여러 가지 정보를 이용하여 갱신문제를 효율적으로 해결할 수 있다. 토폴로지 변화전의 네트워크에 대한 답을 포함하여 각 프로세서가 가지고 있는 정보를 보조정보(*auxiliary information*)라고 하는데, 보조정보를 이용하여 이중결합요소 갱신문제를 효율적으로 해결하는 알고리즘이 제안되었다[8, 9]. 문헌 [8]에서는 보조정보로서 토폴로지 변화전의 네트워크에 대한 이중결합요소의 답을 이용한다.

지금까지 보조정보를 이용하여 너비 우선 신장 트리 갱신문제를 해결하는 분산알고리즘은 제안되지 않았으며, 문헌 [7]에 제안된 너비 우선 신장 트리 문제를 해결하는 분산알고리즘을 토폴로지 변화후의 새로운 네트워크 $N' = (P', L')$에 적용시켰을 경우, 너비 우선 신장 트리 갱신문제를 메시지 복잡도와 이상시간 복잡도 모두 $O(n'\sqrt{e'})$에 해결할 수 있다. 단, $n' = |P'|$, $e' = |L'|$이다. 즉, 문헌 [7]의 알고리즘에서는 토폴로지 변화전의 네트워크에 대해 각 프로세서가 가지고 있는 너비 우선 신장 트리 등에 대한 답을 보조정보로서 이용하지 않고, 완전히 처음부터 너비 우선 신장 트리를 재구성한다.

본 논문에서는 보조정보를 효율적으로 이용함으로써 너비 우선 신장 트리 갱신문제를 효율적으로 해결하는 분산알고리즘을 제안하는데, 보조정보로는 토폴로지가 변하기 전의 네트워크에 대한 너비 우선 신장 트리와 이중결합요소에 대한 답을 이용한다. 즉, 너비 우선 신장 트리에 대한 답으로서, 각 프로세서는 토폴로지 변화전의 네트워크상에서 어느 링크가 부모와 연결된 링크이고, 어느 링크가 자식과 연결된 링크라는 것을 알고 있다. 또한, 어느 링크가 같은 이중결합요소에 속하는지를 알고 있다.

본 논문에서 제안하는 분산알고리즘의 메시지 복잡도와 이상시간 복잡도 모두 $O(p\sqrt{q} + q + a + n')$이다. 여기서, a는 추가 링크의 수, p는 삭제 또는 추가 링크를 가진 이중결합요소에 속하는 전체 노드 수를 나타내며, q는 삭제 또는 추가 링크를 가진 이중결합요소에 속하는 전체 링크수를 나타낸다.

2. 정 의

본 논문에서 고려하는 네트워크 $N = (P, L)$는 프로세서의 집합 P와 링크의 집합 L로 구성되어 있으며, 네트워크 N은 무방향 연결 그래프(connected undirected graph) $G = (V, E)$와 같으므로, 본 논문에서는 무방향 연결 그래프에 관한 용어를 네트워크에 대해서도 동일하게 사용한다. 단, 모든 링크에 할당되어 있는 가중치(weight)는 모두 동일한 네트워크, 즉 링크에는 가중치가 할당되지 않은 것으로 가정하며, 다중링크는 없는 것으로 한다.

[정의 1] 이중결합요소(biconnected component)란 네트워크 N상의 임의의 두 프로세서 사이에 두 개이상의 서로다른 경로가 존재하는 N의 극대부분 그래프로서, 이러한 이중결합요소를 구하는 문제를 이중결합요소 문제라고 한다. 이중결합요소 문제에 대한 답으로서 각 프로세서는 같은 이중결합요소에 속하는 인접링크에 대해서 같은 레이블을 가지며, 서로 다른 이중결합요소에 대해서는 다른 레이블을 갖는다. □

[정의 2] 최단경로 $sp(u, v)$란 두 프로세서 u와 v를 연결하는 경로중에서 그 경로의 길이가 가장 짧은 경로를 말한다. □

[정의 3] 너비 우선 신장 트리 문제란 루트(root)로 지정된 하나의 프로세서 r과 r이외의 모든 프로세서에 대해 최단경로를 구하는 문제를 말한다. □

네트워크와 분산 알고리즘에 관해 다음과 같은 가정을 둔다.

[가정 1] 네트워크상에 공유 메모리는 없고, 프로세서간의 통신은 링크를 통한 메시지 교환만으로 행한

다. 프로세서 u가 인접프로세서 v에게 보낸 메시지는 보낸 순서대로 유한시간내에 도중에 없어지지 않고, v에게 반드시 보내진다. □

[정의 4] 너비 우선 신장 트리 갱신문제(Breadth-First Spanning Tree Updating Problem, 이후 BTUP라고 한다)는 다음의 초기상태에서 시작하여 최종상태에 이르는 문제를 말한다.

(초기상태) 토폴로지 변화전의 네트워크 N에 대한 너비 우선 신장 트리가 이미 구성되어 있으며, 각 프로세서는 토폴로지 변화후의 네트워크 N'에서 어느 링크가 새로 추가되고 삭제되었는지를 알고 있다.

(최종상태) 네트워크 N'에 대한 너비 우선 신장 트리가 구성되어 있다. □

[정의 5] 이중결합요소 갱신문제란 너비 우선 신장 트리 갱신문제에 대한 정의에서와 같이, 이중결합요소가 이미 구성되어 있는 초기상황에서, 토폴로지 변화후에 새로운 네트워크에 대한 이중결합요소를 구하는 문제를 말한다. □

[정의 6] 분산알고리즘이란 각 프로세서가 실행하는 프로그램으로 구성한다. 자발적으로 프로그램을 개시하는 프로세서를 시작 프로세서라고 한다. 시작 프로세서이외의 프로세서는 인접 프로세서로부터 메시지를 받으면 프로그램 실행을 새기하는데, 여기서는 시작 프로세서는 하나라고 가정한다. 메시지 전송 지연의 차이, 프로세서 동작 속도의 차이등에 의해 여러 가지 실행 과정이 발생할 수 있으나, 어느 실행과정에 있어서도 유한개의 명령 실행후에 모든 프로세서가 프로그램 실행을 끝냈을 때 분산알고리즘이 종료했다고 한다. 어떤 분산알고리즘 A가 BTUP를 해결한다는 것은 유한시간내에 A가 종료되고, 그때 각 프로세서 v는 프로세서 r을 루트로 하는 너비 우선 신장 트리상에서의 v의 부모와 자식을 알고 있다. □

[가정 2] 각 프로세서가 실행하는 프로그램은 동일하다. □

분산알고리즘의 평가는 일반적으로 메시지량과 시간을 이용해서 행한다. 시간평가는 메시지의 전송지연과 각 프로세서의 동작 속도에 차이가 있기 때문에 용이하지 않아서 일반적으로 이상시간 복잡도라는 개념을 이용한다.

[정의 7] 메시지 복잡도는 분산알고리즘 실행도중에 네트워크의 모든 프로세서 사이에서 교환되는 메시지 수이다. 또, 이상적인 시간복잡도는 프로세서내에서 처리 시간을 무시하고, 메시지가 링크에 전달되는 전송시간을 1단위 시간으로 했을 때 알고리즘이 종료할 때까지의 단위시간수이다. □

3. BTUP를 해결하는 분산알고리즘 ABU

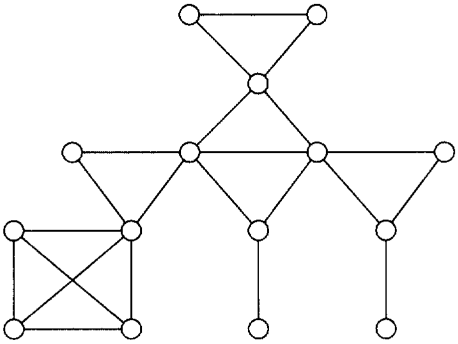
본 논문에서는 네트워크 $N=(P, L)$과 프로세서 r을 루트로 하는 너비 우선 신장 트리 $BT(N, r)=(P_{BT}, L_{BT})$ (단, $P_{BT}=P$, $L_{BT}=\{(u, v) \mid (u, v)$는 모든 프로세서 $p(\in P)$에 대한 $sp(r, p)$상의 링크$\}$)가 구성되어 있는 상황에서, 네트워크의 토폴로지가 변해서 $N'=(P', L')$로 되었을 때, N'에 대한 너비 우선 신장 트리 $BT(N', r)=(P'_{BT}, L'_{BT})$를 재구성하는 분산 알고리즘(Algorithm of Breadth-First Spanning tree Updating problem, 이후 ABU라고 한다)을 제안한다. 본 논문에서는 설명을 간단하게 하기 위해 연결된 네트워크만을 대상으로 하므로, 네트워크 토폴로지의 변화에 따라 네트워크가 분리되는 경우는 없는 것으로 가정한다.

3.1 ABU의 아이디어

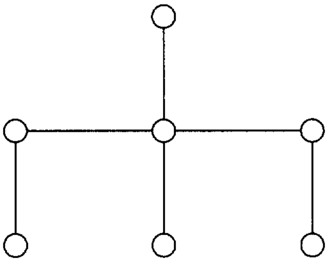
본 논문에서는 BTUP를 효율적으로 해결하기 위해, 두 종류의 보조정보 즉, 이중결합요소 갱신문제의 답과 너비 우선 신장 트리 갱신문제의 답을 이용한다.

네트워크 N상에 있는 하나의 이중결합요소를 하나의 프로세서로 치환하고, 단절점(*articulation point*)에 의해 인접하는 두 개의 이중결합요소에 대해서는 하나의 새로운 링크로 연결시키면 새로운 트리 $T'=(P_{T'}, L_{T'})$를 얻을 수 있다. 즉, $P_{T'}=\{v \mid v$는 N에 있어서 하나의 이중결합요소$\}$, $L_{T'}=\{(u, v) \mid u(\in P_{T'})$와 $v(\in P_{T'})$는 N에 있어서 인접한 이중결합요소$\}$이다.

네트워크 N의 토폴로지가 바뀌어서 N'로 되었다고 가정하자. 이때 N상에 있는 이중결합요소가 삭제링크 또는 추가링크를 포함했는지 여부에 따라 다음의 세 경우로 나누어서 생각할 수 있다.



(a) 네트워크 N



(b) 트리 $T' = (P_{T'}, L_{T'})$

(그림 1) 트리 $T' = (P_{T'}, L_{T'})$의 예

(경우 1) 어떤 이중결합요소도 삭제링크와 추가링크를 포함하지 않은 경우

토폴로지가 바뀌었지만 삭제된 링크가 다시 복구가 되었거나, 추가된 링크가 다시 삭제되어 어떤 이중결합요소에도 아무런 영향을 미치지 않았기 때문에 N가 N'로 바뀌었지만, $N = N'$인 경우로서, 트리 T'의 토폴로지도 변하지 않고, 루트 r과 다른 프로세서사이의 최단경로도 바뀌지 않는다. 따라서, BTUP를 해결하기 위한 추가 작업을 할 필요가 없으며, 토폴로지 변화전의 네트워크에 대한 너비 우선 신장 트리 $BT(N, r)$이 그대로 토폴로지 변화후의 네트워크에 대한 $BT(N', r)$으로 된다.

(경우 2) 삭제링크가 존재하는 이중결합요소가 존재할 경우

삭제링크가 존재하는 이중결합요소를 B라고 하자. 이 경우에 B상에 있는 각 프로세서 u에 대해 경로 r-u가 바뀔지 모른다. 따라서, N'에 있어서 B에 대한 너비 우선 신장 트리를 처음부터 다시 구성해야 한다.

그러나, T'상에 있어서 B의 후손인 다른 이중결합요소 B'에 대해서, B의 변화는 B'에 대한 토폴로지에 영향을 미치지 않으므로, B'에 대해 추가로 변경작업을 할 필요는 없다. 단, B'에 있어서 토폴로지 변화가 있을 경우에는 이들 B'에 대해서만 너비 우선 신장 트리를 다시 구성하는 것으로 충분하다.

앞에서도 설명했듯이, 네트워크의 연결성이라는 가정으로 인해 삭제링크 때문에 네트워크가 분리되는 경우는 없다.

(경우 3) 추가링크를 포함한 이중결합요소가 존재할 경우

추가 링크에 의해 여러개의 이중결합요소(이후, BS로 표현)이 결합되어서 하나의 합병된 이중결합요소 M이 되는 경우로서, M상에 있는 각 프로세서 u에 대해 경로 r-u가 바뀔지 모른다. 따라서, N'에 있어서 BS에 대한 너비 우선 신장 트리를 처음부터 다시 구성해야 한다.

합병된 이중결합요소 M을 하나의 프로세서로 대치시키면, N상에서의 T'가 N'상에 있어서는 T''로 되는데, T'상에서 BS의 후손인 이중결합요소들은 그대로 T''상에서 M의 후손이 된다. 즉, N'에서 M으로 합병되더라도 후손들의 토폴로지에는 아무런 영향을 미치지 않으므로, 이들 후손들에 대해 추가로 변경작업을 할 필요는 없다. 단, B'에 있어서 토폴로지 변화가 있을 경우에는 이들 B'에 대해서만 너비 우선 신장 트리를 다시 구성하는 것으로 충분하다.

3.2 알고리즘 ABU의 개요

너비 우선 신장 트리 $BT(N, r)$로부터 $BT(N', r)$를 재구성하기 위해, 루트 r은 $BT(N', r)$의 서브그래프 $S_{BT(N',r)} = (SV_{BT(N',r)}, SE_{BT(N',r)})$를 반복해서 확장해 나간다. 단, 초기상황에서 $SV_{BT(N',r)} = \{r\}$이고 $SE_{BT(N',r)} = \emptyset$이다. 앞에서 설명한 바와같이, 이중결합요소의 상황에 따라 너비 우선 신장 트리를 재구성함으로써, 루트 r은 토폴로지 변화후의 네트워크 N'에 대한 너비 우선 신장 트리를 재구성하게 된다. 확장과정에서, 루트 r은 이중결합요소를 두 종류의 그룹 즉, 손상된 이중결합요소(*injured biconnected component*)과 무결 이중결합요소(*uninjured biconnected component*)으로 구분한다. 이중결합요소 B상에 삭제링크와 추가링크가 존재하지 않으면, B를 무결 이중결합요소이라고 하고, B상에 삭제링크 또는 추가링크가 존재할 경우, B를 손상된 이중결합요소이라고 한다.

(단계 1) 이중결합요소의 재구성

네트워크 N'에 대한 너비 우선 신장 트리 $BT(N', r)$를 재구성하기 위해, 먼저 N'에 대한 이중결합요소 갱신문제를 해결하는데, 이를 위해서는 문헌 [8]에 있는 저자들의 알고리즘을 이용하여 N'에 대한 이중결합요소 갱신문제를 해결한다. 이때 (단계 3)에서 각 이중결합요소에 대한 너비 우선 신장 트리를 재구성하는데 사용하기 위해 N'상에서의 각 이중결합요소에 속하는 프로세서와 링크수를 구한다. 또한, N'상에서의 각 이중결합요소에는 문제의 답으로서 레이블이 할당된다.

(단계 2) 이중결합요소의 그룹 분류

이중결합요소의 그룹을 분류하는데, 이를 위해서는 네트워크 N에 있어서의 이중결합요소의 레이블과 네트워크 N'에 있어서의 이중결합요소의 레이블을 이용한다. 즉, 이중결합요소에 할당된 레이블이 같으면 그 이중결합요소는 무결 이중결합요소가 되고, 레이블이 다르면 손상된 이중결합요소으로 분류한다.

(단계 3) 너비 우선 신장 트리의 재구성

루트 r은 다음과 같이 해서 각 이중결합요소에 대한 너비 우선 신장 트리를 재구성한다.

(단계 3-1) 무결 이중결합요소의 경우

무결 이중결합요소에 대해, 루트 r과 무결 이중결합요소상에 있는 다른 프로세서와의 사이의 최단경로는 바뀌지 않으므로, 무결 이중결합요소에 대한 너비 우선 신장 트리는 N'에 있어서도 그대로 남는다. 따라서, 이 경우에는 별다른 처리없이 다른 이중결합요소에 대한 처리를 하게 된다.

(단계 3-2) 손상된 이중결합요소의 경우

문헌 [7]에 있는 저자들의 알고리즘에서는 네트워크 $N=(P, L)$에 대한 너비 우선 신장 트리를 구하는데 있어서 한번에 트리의 높이를 $k(=n/\sqrt{e})$씩 반복확장해 간다. 단, $n=|P|$, $e=|L|$이다. 한번에 적어도 k개의 프로세서가 트리에 흡수되어 가므로, 모든 프로세서로 구성된 너비 우선 신장 트리를 구하려면 $n/k(=\sqrt{e})$번의 반복확장이 필요하다.

(단계 3-2)에서는 손상된 이중결합요소 B에 대해서, 문헌 [7]에 있는 저자들의 알고리즘을 이용하여 B에 대한 너비 우선 신장 트리를 재구성하는데, 이때 B에 대한 너비 우선 신장 트리의 높이도 한번에 $k(=p/\sqrt{q})$씩 증가하면서 반복확장되어 간다. (p와 q는 모든 손상된 이중결합요소에 속하는 프로세서와 링크수). 한번에 너비 우선 신장 트리에 흡수되는 프로세서수는 적어도 $k(=p/\sqrt{q})$개이므로, 손상된 이중결합요소상에 존재하는 모든 프로세서를 포함한 너비 우선 신장 트리를 구하려면 $p/k(=\sqrt{q})$번의 반복확장이 필요하며, 이들 프로세서와 링크수는 (단계 1)에서 구해진다.

너비 우선 신장 트리 $BT(N', r)$의 서브그래프 $S_{BT(N',r)}=(SV_{BT(N',r)}, SE_{BT(N',r)})$가 구성되어 있다고 가정하자. 이때, 서브그래프 $S_{BT(N',r)}$를 이용하여, 서브그래프 $S_{BT(N',r)}$에 속하는 프로세서 $u(\in SV_{BT(N',r)})$로부터 거리 k이내에 있는 모든 프로세서 $v(\not\in SV_{BT(N',r)})$와 링크를 서브그래프 $S_{BT(N',r)}$에 추가함으로써 서브그래프 $S_{BT(N',r)}$를 확장한다. 즉, $SV_{BT(N',r)}=SV_{BT(N',r)} \cup \{v|$각 $u\in SV_{BT(N',r)}$에 대해, u로부터 거리 k이내에 있는 프로세서 $v(\not\in SV_{BT(N',r)})\}$와 $SE_{BT(N',r)}=SE_{BT(N',r)} \cup \{(y, z)|$각 $u(\in SV_{BT(N',r)})$에 대해, u로부터 거리 k이내에 있는 두 프로세서 y와 z사이의 링크 $(y, z)\}$이 된다.

4. 알고리즘 ABU의 평가

[정리 1] 알고리즘 ABU는 BTUP를 제대로 해결한다.

(증명) (단계 1)에서는 문헌 [8]에 있는 이중결합요소 갱신알고리즘을 이용하여 이중결합요소 갱신문제를 해결하므로, (단계 1)의 정당성은 이미 검증되었다. (단계 2)에서 이중결합요소에 할당되어 있는 레이블을 이용하여 이중결합요소를 분류하는데, 이의 정당성은 자명하다. 그리고, (단계 3)에서는 각 이중결합요소에 대한 너비 우선 신장 트리를 재구성한다. 즉, 만일 이중결합요소 B가 손상된 이중결합요소인 경우에만 문헌 [7]에 있는 너비 우선 신장 트리를 구성하는 알고리즘을 이용하여 B에 대한 너비 우선 신장 트리를 재구성하므로, (단계 3)의 정당성도 이미 검증되었다.

따라서, 본 논문에서 제안한 알고리즘이 BTUP를 제대로 해결한다는 것은 명백하다. □

다음 정리에서 n'는 네트워크의 토폴로지가 변한후의 네트워크상에 존재하는 노드수를 각각 나타내고, a는 추가 링크의 수를 나타낸다. 그리고, p는 삭제 또는

추가 링크를 가진 이중결합요소에 속하는 전체 노드수를 나타내며, q는 삭제 또는 추가 링크를 가진 이중결합요소에 속하는 전체 링크수를 나타낸다.

먼저, 문헌 [7]과 [8]의 알고리즘의 결과와 본 논문의 결과를 보면 <표 1>과 같다.

〈표 1〉 문헌 [7]과 [8]의 결과

	메시지 복잡도	이상 시간 복잡도
너비 우선 신장 트리 알고리즘[7]	$O(n'\sqrt{e'})$	$O(n'\sqrt{e'})$
이중 결합 요소 갱신 알고리즘[8]	$O(q+a+n')$	$O(q+a+n')$
본 논문의 알고리즘	$O(p\sqrt{q}+q+a+n')$	$O(p\sqrt{q}+q+a+n')$

일반적으로 $p \le n'$와 $q \le e'$ 그리고 $q \le n'\sqrt{e'}$와 $a \le n'\sqrt{e'}$가 성립한다. 따라서, 본 논문의 알고리즘은 메시지복잡도와 이상시간 복잡도에 관해서 문헌 [7]의 알고리즘보다 뛰어나다. 즉, 보조정보를 이용하는 것이 보조정보를 이용하지 않는 경우보다 효율적이다.

[정리 2] 알고리즘 ABU의 메시지 복잡도와 이상시간 복잡도는 모두 $O(p\sqrt{q}+q+a+n')$이다.

(증명) (단계 1)에서는 이중결합요소 갱신문제를 해결하므로, (단계 1)에서의 메시지 복잡도와 이상시간 복잡도 모두 $O(q+a+n')$이 된다[8].

(단계 2)에서는 이중결합요소를 무결 이중결합요소 또는 손상된 이중결합요소으로 분류하는데, 이때 토폴로지 변화전의 네트워크 N상에서의 이중결합요소에 할당된 레이블과 토폴로지 변화후의 네트워크 N'상에서의 이중결합요소에 할당된 레이블을 이용하여 간단히 판단할 수 있으므로, (단계 2)에서의 메시지 복잡도와 이상시간 복잡도는 모두 $O(1)$이 된다.

(단계 3-1)에서는 무결 이중결합요소에 대한 너비 우선 신장 트리를 구성하는데, 이때 N상에서의 너비 우선 신장 트리가 그대로 N'에서의 너비 우선 신장 트리로 된다. 트리에는 n-1이하의 링크가 있으므로, (단계 3-1)에서의 메시지 복잡도와 이상시간 복잡도는 모두 $O(n)$이 된다.

문헌 [7]의 알고리즘은 프로세서와 링크수가 각각 n과 e인 네트워크에 대한 너비 우선 신장 트리를 구하는데 메시지 복잡도와 이상시간 복잡도가 각각 $O(n\sqrt{e})$이다. (단계 3-2)에서는 문헌 [7]의 알고리즘을 이용하여 손상된 이중결합요소에 대한 너비 우선 신장 트리를 구하므로, 모든 손상된 이중결합요소에 속하는 프로세서와 링크수를 각각 p와 q라고 가정하면, (단계 3-2)에서의 메시지 복잡도와 이상시간 복잡도는 모두 $O(p\sqrt{q})$가 된다.

따라서, 본 논문에서 제안하는 알고리즘 ABU의 메시지 복잡도와 이상시간 복잡도는 모두 $O(p\sqrt{q}+q+a+n')$이 된다. □

5. 결 론

본 논문에서는 비동기식 네트워크에서 너비 우선 신장 트리 문제를 해결하는 메시지 복잡도와 이상시간 복잡도가 $O(p\sqrt{q}+q+a+n')$인 분산알고리즘을 제안한다. 기존 결과로는 문헌 [7]의 알고리즘을 들 수 있는데, 이 알고리즘을 네트워크 토폴로지가 변한 경우에 적용하면 메시지 복잡도와 이상시간 복잡도가 $O(n'\sqrt{e'})$가 되어, 본 논문의 알고리즘이 메시지 복잡도와 이상시간 복잡도 면에서 기존 결과보다 효율적인 알고리즘이 된다.

참 고 문 헌

[1] B. Awerbuch, "Complexity of network synchronization," *Journal of ACM*, Vol.32, No.4, pp.804-823(Oct. 1985).

[2] B. Awerbuch and R.G.Gallager, "A new distributed algorithm to find breadth first search trees," *IEEE Trans. on Information Theory*, Vol.IT-33, No.3, pp. 315-322(1987).

[3] B. Awerbuch, "Distributed shortest paths algorithm," *Proc. of 21st Symposium on Theory of Computing*, pp.490-500(1980).

[4] M Ahuja and Y.Zhu, "An efficient distributed algorithms for finding articulation points, bridges and biconnected components in asynchronous network," *In Porc. 9th Conference on Foundations of Software Technology and Theoretical Computer Science* (LNCS 405), pp.99-108(1989).

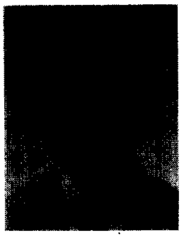
[5] T.H. Cormen, C.E.Lerserson and R.L.Rivest, "Introduction to Algorithms," The MIT Press(1990).

[6] T.Kameda and M.Yamashita, "Distributed Algorithms," Kindai-Kagaku-sya(1994).

[7] J. Park, T.Masuzawa, K.Hagihara and N.Tokura, "An efficient distributed algorithm for breadth first spanning tree problem," *Journal of IEICE*(D), Vol. J71-D, No.7, pp.1576-1188(1988).

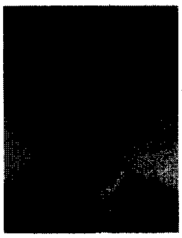
[8] J. Park and Y. Park, "An Algorithm Solving the Biconnected-components Reconstruction Problem," *Journal of KIPS*, Vol.5, No.10, pp.2513-2520(1998).

[9] B. Swaminathan and K.J.Goldman, "An incremental distributed algorithm for computing biconnected components," *Proc. 8th International Workshop on Distributed Algorithms*(LNCS 857), pp.238-252(1994).



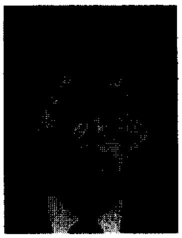
박 정 호

e-mail : jhpark@omega.sunmoon.ac.kr
1980년 성균관대학교 사범대학 졸업(문학사)
1980년~1982년 성균관대학교 경영대학원 정보처리학과 (경영학석사)
1985년~1987년 日本 오사까대학교 대학원 정보공학전공 (공학석사)
1987년~1990년 日本 오사까대학교 대학원 정보공학전공(공학박사)
1996년~현재 한국정보처리학회 총무이사
1991년~현재 선문대학교 컴퓨터정보학부 부교수
1999년~현재 선문대학교 연구처장
관심분야 : 분산알고리즘, 원격교육, XML, 소프트웨어공학



박 윤 용

e-mail : yypark@omega.sunmoon.ac.kr
1983년 숭전대학교 계산통계학과 졸업(학사)
1985년 서울대학교 대학원 계산통계학과(이학석사)
1994년 서울대학교 대학원 계산통계학과(이학박사)
1985년~1992년 한국전자통신연구소 연구원
1992년~현재 선문대학교 컴퓨터정보학부 부교수
관심분야 : 분산처리운영체제, 분산객체지향처리시스템, 분산객체표준화



황 석 형

e-mail : shwang@omega.sunmoon.ac.kr
1991년 강원대학교 전자계산학과 조기 졸업(이학사)
1992년 日本 오사카대학 기초공학부 연구생
1994년 日本 오사카대학 대학원 정보공학과(공학석사)
1997년 日本 오사카대학 대학원 정보공학과(공학박사)
1997년~현재 선문대학교 컴퓨터정보학부 조교수
관심분야 : 소프트웨어공학(특히, Adaptive Object-Oriented Software, Formal Method), Adaptive Programming 기법, CAI