동적 네트워크에서 최소 신장 트리를 유지하는 분산 알고리즘

김형식[0] 좌경룡
한국과학기술원 전 산 학 과
{morethan, kychwa}@tclab.kaist.ac.kr

A Distributed Algorithm for Maintaining a Minimum Spanning Tree in Dynamic Network

Hyoung-Shick Kim[0] Kyung-Yong Chwa
Dept. of Computer Science, KAIST

요 약

본 논문은 동적 네트워크에서 최소 신장 트리를 유지하는 문제에 대한 알고리즘을 제안한다. 동적 네트워크란 새로운 간선이 추가되거나 기존의 간선이 삭제 가능한 네트워크를 의미한다. 최소 신장 트리를 찾는 여러 분산 알고리즘은 동적 변화를 고려하지 않거나 혹은 별도의 자료 구조를 이용하였다.
제안한 알고리즘은 간선의 변화에 대응하여 인접한 노드들에게 변화를 알리고 서로 협력하여 최소 신장 트리를 찾는다.
네트워크 G의 전체 노드의 수를 N, 전체 간선의 수를 E, 찾은 최소 신장 트리의 지름을 D라고 할 때, k개의 간선 추가와 삭제에 대하여 각각 $\min(O(kD) + O(k), O(N \log N + E))$와 $O(N \log k + E)$의 메시지 복잡도를 갖는다.
보한 각 경우에 대한 하한 비용을 증명하였다.

1. 서론

최소 신장 트리(minimum spanning tree)는 네트워크 동기화(network synchronization)[1], 너비 우선 탐색(breadth-first-search)[2], 교착상태 해결(deadlock resolution)[3], 네트워크의 노드 수 계산 (counting problem)[7], 상호배제(mutual exclusion)[4] 등과 같은 문제에 있어서 주요한 구조로 사용되며, 네트워크 상에서 정보를 방송하는 경우(broadcasting)에 있어서도 최소의 통신 비용으로 전달할 수 있게 한다[5][6]. 또한, 임의의 네트워크에서 대표자를 선출하는 문제는 분산 시스템에서 대칭성을 해결하는 중요한 도구이며 이 문제는 최소 신장 트리를 찾는 문제로 유도될 수 있다.
Gallager 등은 이 분야에서 기초가 되는 알고리즘을 제안하였고 여러 중요한 개념들을 도입하였다[8]. 이 알고리즘은 $O(N \log N + E)$의 메시지 복잡도와 $O(N \log N)$의 시간 복잡도를 갖는다. 여기서, N와 E는 각각 네트워크의 노드와 간선의 수를 의미한다. 최소 신장 트리를 찾는 문제에 대한 메시지 복잡도의 하한이 $\Omega(N \log N + E)$이기 때문에, 메시지 복잡도에 대하여 최적인 결과임을 알 수 있다. 이는 신장 트리를 찾는 문제에 대한 하한인 $\Omega(E)$[8]와 원형(ring) 구조에서 대표자를 선출하는 문제의 하한인 $\Omega(N \log N)$[9]으로부터 유도되었다.
반면에, 시간 복잡도에 대한 하한은 $\Omega(N)$이고 Gallager 등이 제안한 알고리즘의 시간 복잡도는 $O(N \log N)$이기 때문에 개선의 여지를 가진다. 이후, Chin과 Ting[10] 그리고 Gafni[11]는 $O(N \log^* N)$의 시간 복잡도를 갖는 알고리즘을 제안하였고, Awerbuch는 $O(N)$의 시간 복잡도를 갖는 알고리즘[7]을 제안하였다. 이후, Faloutsos와 Molle은 Awerbuch가 제안한 알고리즘의 부결성 증명에서의 오류를 보정하였다[12].
본 논문은 동적 네트워크에서 최소 신장 트리를 유지하는 문제에 대하여 메시지 복잡도가 효율적인 알고리즘을 제안한다. 네트워크에 동적 변화가 발생할 경우, 이전에 찾은 최소 신장 트리는 더 이상 최소 신장 트리를 보장할 수 없기 때문에 새롭게 최소 신장 트리를 찾아야 한다. 정적 네트워크만을 고려한 기존에 제안된 알고리즘들은 이전에 찾은 최소 신장 트리와는 무관하게 새롭게 최소 신장 트리를 찾는 방식이기 때문에 $O(N \log N + E)$의 메시지 복잡도를 감수해야만 한다.
반면에 본 논문에서는 동적 변화가 일어난 경우에 새롭게 최소 신장 트리를 찾는 방식이 아니라 이전에 찾은 트리의 정보를 이용하여 유지 보수하는 접근 방법을 사용하기 때문에 최소 신장 트리를 찾는 비용을 보다 효과적으로 개선할 수 있다. 네트워크에서 기존의 간선이 k개 삭제되는 경우에는 $O(N \log k + E)$의 메시지 복잡도를 가지며, 새로운 간선이 k개 추가되는 경우에 $\min(O(kD) + O(N), O(N \log N + E))$의 메시지 복잡도를 가진다.
본 논문에서 제안하는 알고리즘의 모델은 다음과 같다. 네트워크 G는 N개의 노드들과 E개의 간선들을 가지는 무방향 가중치 그래프(undirected weighted graph)이다. 각 노드는 노드 식별자(node identification)를 갖는다. 노드 식별자와 G의 간선들의 가중치는 유일하다.
G의 최소 신장 트리는 N개의 노드로 구성되며 포함된 모든 간선들의 가중치의 합이 최소가 되는 연결 부분 그래프(connected subgraph)이다. 조각(fragment)은 최소 신장 트리에 포함되는 연결 부분 그래프이다. 조각의 외부 간선(outgoing edge)은 조각에 포함된 노드들의 인접한 간선 중, 간선의 반대편에 인접한 노드가 다른 조각에 포함된 간선을 말한다. 최소 가중치의 외부 간선(minimum outgoing edge)은 외부 간선들 중에서 최소의 가중치를 갖는 간선을 가리킨다.
각 노드는 개별적인 프로세서로서 행동하고 간선은 양방향으로 통신 가능한 고장이 일어나지 않는 채널로 가정한다. 또한 간선을 통하여 전달되는 메시지는 비동기적으로 전송되지만, 순차적이고 유한한 시간 내에 전송됨을 가정한다. 각 프로세서에서 동작하는 알고리즘은 오직 인접한 이웃들과 메시지 교환을 통해서만 이루어진다. 초기에 모든 노드는 "asleep" 상태에 있고, 노드에 인접한 간선은 "branch" 혹은 "rejected" 상태로 주어진다. "branch"는 최소 신장 트리에 포함되는 간선을 의미하고 "rejected"는 최소 신장 트리에 포함되지 않는 간선을 의미한다. "branch" 중에서는 루트 노드로의 간선이 존재하고 이를 구별할 수 있다. 또한, 모든 노드는 루트 노드에 대한 식별자(root node identification) "root"를 가진다. 루트 노드는 "root"가 자신의 식별자인 노드이며 부모 노드로의 간선을 가지지 않는다. 루트 노드를 포함한 어떤 노드도 자신의 상태와 인접한 간선들을 제외하고는 네트워크 위상에 관련된 어떤 정보도 유지하고 있지 않다.
동적 네트워크에서 발생할 수 있는 동적인 변화는 새로운 간선의 추가, 존재하는 간선의 삭제, 간선 가중치의 변화가 있다[13]. 동적 네트워크는 동적 변화가 발생할 수 있는 네트워크를 가리킨다.

2. 동적 네트워크에서의 알고리즘

동적 네트워크에서의 알고리즘은 발생하는 동적인 변화에 대응하여 최소 신장 트리를 찾는다. 새로운 간선이 추가되거나 존재하는 간선이 삭제될 수 있고, 간선의 가중치가 변할 수 있다. 동적인 변화가 일어나는 수는 k로 놓는다. 만약 동적인 변화에 대응하여 알고리즘이 동작하는 도중에 새로운 동적 변화가 발생한다면, 이전의 변화를 모두 반영한 후에 새로운 동적 변화에 대응하는 제약을 가진다.

2.1 간선 추가 알고리즘

네트워크에 새로운 간선이 1개 이상 추가된다면, 이전에 찾은 트리는 더 이상 네트워크에서의 최소 신장 트리임을 보장하지 않는다. 이는 다음과 같은 보조 정리 1의 결과로부터 유도할 수 있다.

보조 정리 1 T는 G의 모든 노드들을 포함하는 트리이고 e는 T에 포함되지 않는 그래프 G의 간선이고 e'은 $(T \cup \{e\}) - \{e\}$에서 임의의 간선이라고 놓았을 때, 만약 $(T \cup \{e\}) - \{e'\}$이 연결 그래프라면, $(T \cup \{e\}) - \{e'\}$은 트리이다.

보조 정리 1의 결과로부터 이전에 찾은 트리에 포함된 기존의 간선을 한 개 제거하고 새롭게 추가된 간선을 이전에 찾은 트리에 포함시키는 경우에 만약 모든 노드들이 연결되어 있다면 트리임을 알 수 있다. 그리고 새롭게 추가한 간선의 가중치가 제거된 기존의 간선 가중치보다 작다면 이전에 찾은 트리보다 간선 가중치의 합이 더 작은 최소 신장 트리를 찾을 수 있다.

보조 정리 2 T는 G의 모든 노드들로 구성된 트리이고 e는 T에 포함되지 않는 그래프 G의 간선이라고 놓았을 때, $(T \cup \{e\})$은 회로를 만든다. 또한, $(T \cup \{e\})$가 만든 회로로부터 회로 내의 임의의 간선을 삭제하고 남은 그래프는 연결 그래프이다.

최소 신장 트리를 찾기 위해서는 보조 정리 3의 결과를 이용한다.

보조 정리 3 네트워크에 존재하는 모든 간선들의 가중치가 유일하고 간선 e가 회로에서 가장 큰 가중치를 갖는 간선이라면, 간선 e는 그래프의 최소 신장 트리에 포함되지 않는다.

제안된 알고리즘은 새롭게 추가된 간선을 포함하여 만들어진 회로들부터 회로 내의 가장 큰 가중치를 가지는 간선을 제거하는 방법을 수행함으로써 최소 신장 트리를 유지한다. 보조 정리 1과 2는 다음과 같이 확장될 수 있다.

보조 정리 4 T는 G의 모든 노드들을 포함하는 트리이고 $\{e_1, e_2, \ldots, e_k\}$는 T에 포함되지 않는 그래프 G의 간선들이고 $e'_1, e'_2, \ldots, e'_k$은 $(T \cup \{e_1, e_2, \ldots, e_k\}) - \{e'_1, e'_2, \ldots, e'_k\}$이 연결 그래프라면, $(T \cup \{e_1, e_2, \ldots, e_k\}) - \{e'_1, e'_2, \ldots, e'_k\}$은 트리이다.

보조 정리 4의 결과로부터 k개의 새로운 간선이 추가될 경우, 최소 신장 트리를 찾는 문제는 간선들이 추가된 그래프에서 연결 그래프의 성질을 만족시키면서 제거할 수 있는 가장 큰 k개의 가중치를 갖는 간선들을 찾는 문제로 유도된다. 따라서 알고리즘은 회로 내에서 가장 큰 가중치를 갖는 간선을 결정하고, 결정된 간선을 제거하는 과정을 거친다. 이때, 고려해야 할 점은 회로들이 서로 중첩될 수 있고, 이로 인하여 회로 내의 가장 큰 가중치의 간선들이 중복될 수 있다는 사실이다. 간선이 중복되는 것을 막기 위해서는 이미 결정된 간선의 가중치에 대해서는 다시 결정하지 않는 방법이 고안되어야 한다. 더욱이 트리의 성질을 보장하기 위하여 루트 노드를 제외한 모든 노드는 부모 노드에 대한 유일한 간선을 항상 유지해야 한다.

알고리즘은 보고 단계와 합계 단계, 유지 단계의 세 가지 단계로 구분된다. 보고 단계는 간선이 추가되었음을 루트 노드에게 보고하는 단계이다. 이 단계의 목적은 네트워크의 모든 노드들에게 새로운 간선들이 추가되었음을 알리고 일괄적으로 합계 단계에 진입하도록 하는데 있다. 메시지가 비동기적으로 전달되기 때문에 보고 단계가 존재하지 않을 경우에 합계 단계에서 교착 상태가 일어날 수 있다. 루트 노드는 네트워크에 새로운 간선이 추가되었다는 것을 보고 받으면 방송을 통하여 네트워크의 모든 노드들에게 합계 단계에 진입할 것을 지시한다. 합계 단계는 루트 노드가 네트워크의 노드 수와 간선 수, 추가된 간선들의 수, 그리고 추가된 간선에 인접한 노드들 사이의 가장 긴 거리를 계산하는 단계이다. 루트 노드는 계산한 결과로부터 보다 효율적인 최소 신장 트리를 찾는 알고리즘을 선택한다. 마지막 단계인 유지 단계는 모든 노드들이 협력하여 선택한 알고리즘을 수행하는 단계이다. 루트 노드는 회로 제거 알고리즘과 정적 네트워크에서의 최소 신장 트리를 찾는 알고리즘[12] 중, 보다 효율적인 알고리즘을 결정하고 네트워크의 노드들에게 지시한다. 각 단계에서의 상세한 동작 과정은 다음과 같다.

네트워크에 새로운 간선이 추가된 경우, 추가된 간선에 인접한 노드는 보고 단계를 시작한다. 노드는 추가된 간선의 상태를 " new" 라고 놓고, REPORTINSERTION 메시지를 부모 노드에게 전달한다. REPORTINSERTION 메시지를 받은 노드가 루트 노드가 아니고 처음으로 메시지를 받은 경우라면, 중간 전달자가 되어 다시 부모 노드에게 메시지를 전달한다. 이전에 REPORTINSERTION 메시지를 받은 노드가 다시 메시지를 받는다면 아무런 동작을 취하지 않는다. 루트 노드가 REPORTINSERTION 메시지를 받고 " counting" 상태가 아니라면, 루트 노드는 노드의 상태를 " counting"으로 바꾸고 각 노드들이 합계 단계에 진입할 수 있도록 COUNT 메시지를 네트워크에 방송한다.

COUNT 메시지를 받은 노드들은 노드의 상태를 " counting"으로 바꾸고 하위 그래프에 COUNT 메시지를 방송한다. COUNT 메시지를 받은 노드가 말단 노드인 경우에는 인접한 간선 중에서 " new" 인 간선들의 수와 나머지 간선들의 수, 그리고 자신을 포함한 하위 노드들의 수와 홉 카운트의 네 가지 항목으로 구분하여 COUNTED 메시지에 포함하여 부모 노드에게 전달하고 노드의 상태를 COUNTED로 놓는다. COUNTED 메시지를 전달하는 노드가 말단 노드인 경우에는 자신을 포함한 하위 노드들의 수에 대한 항목은 1로 놓고, 홉 카운트는 인접한 간선 중 " new" 상태인 간선이 있는 경우에만 1로 놓는다. 만약 " new" 상태인 간선이 없다면 0으로 설정한다. " counting" 상태인 노드가 모든 자식 노드들로부터 COUNTED 메시지를 받는다면 그 수를 역시 네 가지 항목으로 나누고 자신의 지역 정보를 추가로 더하여 합산한 뒤, 부모 노드에게 메시지를 전달한다. 이때 네트워크에 추가된 간선들과는 관계가 없는 자식 노드에 대한 간선은 상태를 " ignored" 로 놓는다. 홉 카운트는 세 가지 경우로 나누어 설정된다. 만약 모든 자식 노드들에 대한 간선이 " ignored" 로 놓이고 자신의 인접한 간선들 중에도 " new" 상태인 간선이 없다면 홉 카운트는 0으로 설정된다. 자식 노드들에 대한 간선 중에서 적어도 한 개 이상의 간선이 " ignored" 상태가 아니라면 그러한 간선들 중에서 가장 큰 홉 카운트를 선택하고 1을 더한다. 모든 자식 노드들에 대한 간선이 " ignored" 상태이고 자신의 인접한 간선 중에서 " new" 상태인 간선이 존재한다면 홉 카운트는 1로 설정된다. " counting" 상태인 노드가 COUNTED 메시지 외의 메시지를 전달 받는다면 이는 무시하고 아무런 동작을 취하지 않는다. " counting" 상태에 있는 루트 노드가 COUNTED 메시지를 받으면, 메시지에 포함된 " new" 상태인 간선들의 수 k와 나머지 간선들의 수 $E-k$, 루트 노드를 제외한 네트워크에 속하는 노드들의 수 $N-1$, 추가된 간선에 인접한 노드들 사이의 가장 긴 거리 D로부터 $O(kD)$와 $O(N \log N + E)$를 계산한다. $O(kD)$가 $O(N \log N + E)$보다 크다면 루트 노드는 네트워크 전체에 RECONSTRUCT 메시지를 방송한다. RECONSTRUCT 메시지를 받은 노드들은 Faloutsos와 Molle가 수정한 Awerbuch의 정적 네트워크에서의 최소 신장 트리를 찾는 알고리즘[12]을 사용하여 최소 신장 트리를 찾는다. 만약 $O(kD)$가 $O(N \log N + E)$보다 크지 않은 경우에는 루트 노드는 CONSERVE 메시지를 " ignored" 상태가 아닌 간선들을 통하여 네트워크에 방송한다. CONSERVE 메시지를 받은 모든 노드들은 노드의 상태를 " conserving" 상태로 놓는다. 만약 CONSERVE 메시지를 받은 노드가 말단 노드이거나 모든 자식 노드들에 대한 간선들이 " ignored" 인 경우, 부모 노드에게 INSERTEDGE 메시지를 보낸다. INSERTEDGE 메시지를 받은 노드는 메시지가 전달된 간선을 " cycle" 상태로 놓는다. 메시지에 포함되는 정보는 회로의 이름과 회로 내에서 삭제되어야 할 후보 간선의 쌍으로 이루어진다. 회로의 이름은 새로 추가된 간선의 가중치가 된다. 가장 처음으로 보내는 INSERTEDGE 메시지에 포함되는 회로의 이름은 새로 추가된 간선 중에서 가장 큰 가중치를 선택한다. 인접한 간선 중에서 " new" 상태인 간선을 가지는 노드는 " new" 상태인 간선들 중에서 가장 큰 가중치를 선택하여 INSERTEDGE 메시지에 포함한 뒤, 자신의 큐에 메시지를 저장한다. 모든 자식 노드들로부터 INSERTEDGE 메시지를 받은 노드는 INSERTEDGE 메시지에 포함된 회로의 이름들이 중복되는지를 검사한다. 중복된 경우에는 부모 노드에게 메시지를 전달하지 않고, 자신의 상태를 " agent" 라고 놓는다. 중복되지 않은 경우에는 부모 노드에게 INSERTEDGE 메시지에 포함될 정보를 다음과 같은 방법으로 갱신하여 전달한다. 회로의 이름은 자식 노드들로부터 받은 메시지들에 포함된 회로의 이름과 인접한 새로 추가된 간선들의 가중치를 비교하여 가장 큰 값을 회로의 이름으로 선택한다. 후보 간선은 " cycle" 상태인 인접한 간선들의 가중치와 자식 노드들로부터 받은 메시지에 포함된 후보 간선의 가중치를 비교하여 그 중 가장 큰 값을 선택한다. " agent" 상태인 노드는 하위 그래프에 회로가 존재하는 회로 내에서 가장 큰 가중치의 간선을 제거하도록 동작한다. " agent" 인 노드는 회로 내의 가장 큰 가중치를 DETERMINED 메시지에 포함하여 " cycle" 상태인 간선들을 통하여 전달한다. DETERMINED 메시지를 받은 노드는 메시지에 포함된 값과 동일한 가중치를 가지는 인접한 간선이 존재하는지를 검사한다. 존재하지 않는 경우에는 " cycle" 상태인 간선들을 통하여 자식 노드들에게 DETERMINED 메시지를 전달한다. 만약, 메시지에 포함된 값과 동일한 가중치를 가지는 인접한 간선이 존재한다면, 이 간선을 통하여 REMOVE 메시지를 전달하고 간선을 " rejected" 상태에 놓는다. REMOVE 메시지를 전달 받은 노드는 REMOVE 메시지가 전달된 간선을 " rejected" 상태에 놓는다. REMOVE 메시지를 주고 받은 제거된 간선에 인접한 두 노드는 간선 제거 작업을 완료를 FINISHED 메시지를 통하여 " agent" 상태인 노드에게 알린다. " agent" 상태인 노드는 ORIENT 메시지를 새로 추가된 간선을 포함한 하위 그래프에 방송함으로써 간선들의 방향이 루트 노드를 향하도록 교정한다. ORIENT 메시지를 처음으로 받은 노드는 메시지가 전달된 간선을 " branch" 로 설정하고 부모 노드로의 간선으로 놓는다. ORIENT 메시지를 받은 노드가 " ignored" 인 간선들을 제외하고 유일한 간선을 갖는다면, 간선을 " ignored" 로 설정하고 간선을 통하여 이를 알린다. ORIENT 메시지를 받은 노드가 모든 자식 노드들에 대한 간선들이 " ignored" 이거나 말단 노드인 경우, CONSERVE 메시지를 받은 것과 부모 노드에 대한 간선이 " ignored" 인 경우에만 동작을 취하지 않는다는 점만을 제외하고 동일하게 동작한다. 이와 같은 과정은 INSERTEDGE 메시지를 보내는 노드가 존재하지 않을 때까지 계속해서 반복한다.

2.2 간선 삭제 알고리즘

네트워크의 간선이 삭제될 때, " branch" 상태인 간선이 삭제되는 경우와 " rejected" 상태인 간선이 삭제되는 두 가지 경우로 나눌 수 있다. " rejected" 상태인 간선이 삭제되는 경우에는 이전에 찾은 최소 신장 트리는 간선의 삭제와 상관없이 유지되지만, 최소 신장 트리에 포함되는 " branch" 상태인 간선이 삭제되는 경우에 최소 신장 트리는 두 개의 조각으로 나누어진다. 이러한 경우에 최소 신장 트리를 유지하기 위해서는 분리된 두 개의 조각을 연결시켜주는 최소 가중치의 간선을 찾아서 연결시켜 주어야 한다.

보조 정리 5 최소 신장 트리의 임의의 조각 f가 주어지고, e를 f의 최소 가중치의 외부 간선이라고 한다. 이 경우에 임의의 조각 f에 간선 e와 조각 f에 포함되지 않는 간선 e의 인접한 노드를 결합하면, 최소 신장 트리의 조각 f'이 된다.

보조 정리 6 연결 그래프의 모든 간선들의 가중치가 유일하다면 최소 신장 트리는 유일하다.

보조 정리 7 트리 T가 주어지고 T에 포함된 k개의 간선들이 삭제된 경우에 $(k+1)$개의 최소 신장 트리에 포함되는 조각들이 생긴다.

알고리즘은 보고 단계, 합계 단계, 초기화 단계, 유지 단계의 네 가지 단계로 구분된다. 보고 단계는 간선 삭제가 이루어졌음을 보고하는 단계이다. 보고 단계는 노드가 여러 번 초기화되는 것을 방지한다. 인접한 간선 중에서 임의의 간선들이 삭제된 노드는 부모 노드가 존재할 경우에는 부모 노드에게 REPORTDELETION 메시지를 보내고, 노드의 상태를 " reported" 로 놓는다. REPORTDELETION 메시지를 부모 노드에게 전달한 노드는 이후에 REPORTDELETION 메시지를 전달하지 않는다. REPORTDELETION 메시지를 받거나 인접한 간선들 중에서 삭제된 간선이 존재하는 노드가 부모 노드를 가지지 않는다면, 그 노드는 자신을 조각의 루트 노드로 놓고 하위 노드들에게 COUNTNODES 메시지를 전송한다. 조각의 루트 노드는 COUNTNODES 메시지의 방송을 통하여 조각에 포함된 노드들에게 합계 단계를 기시한다. COUNTNODES 메시지를 받은 노드는 자식 노드들로부터의 노드 수에 대한 응답을 기다린다. COUNTNODES 메시지를 받은 노드가 말단 노드라면 COUNTED 메시지에 1을 더하여 응답한다. 모든 자식 노드들로부터 COUNTED 메시지를 받는다면, 이를 합산하고 자신의 노드 수 1을 더하여 루트 노드에게 COUNTED 메시지를 전달한다. 루트 노드가 모든 자식 노드들로부터 COUNTED 메시지를 받는다면, 이 메시지들에 포함된 값들의 합에 자신의 노드 수 1을 더하여 조각의 전체 노드 수 N_f를 구한다. 루트 노드는 $\log N_f$를 조각의 레벨 L_f로 설정하고 L_f와 루트 노드의 식별자를 조각 식별자로 설정하여 INITIATE 메시지에 첨부한다. INITIATE 메시지를 조각의 모든 노드들에게 방송한 후에는 Gallager 등이 제안한 알고리즘과 유사하게 동작한다. Gallager 등이 제안한 알고리즘과의 차이점은 TEST 메시지를 받은 노드가 자신이 포함된 조각의 루트 노드로부터 INITIATE 메시지를 받을 때까지는

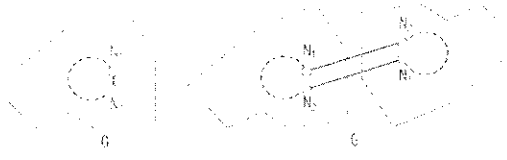

그림 4.1: 최대 가중치의 간선이 두 개 선택되는 네트워크

TEST 메시지에 대하여 응답하지 않는다는 점이 다르다.

정리 1 간선의 가중치가 변화하는 사건과 노드의 추가, 삭제에 대해서도 알고리즘을 적용할 수 있다.

제안된 알고리즘의 무결성을 증명하기 위해서 알고리즘이 반드시 종료하고 알고리즘이 종료되었을 때, 최소 신장 트리가 유지된다는 것을 증명해야 한다.

정리 2 제안된 알고리즘은 교착 상태를 발생시키지 않는다.

최소 신장 트리를 찾는 문제는 탐망법(greedy method)에 의하여 해결될 수 있다. 그래프에서 최소 신장 트리를 찾는 방법은 보조 정리 2와 보조 정리 4의 결과를 이용하는 두 가지의 접근 방법으로 나눌 수 있다. 본 논문에서 제안한 알고리즘은 간선 추가 알고리즘의 경우에는 보조 정리 2의 접근 방법을 이용하고, 간선 삭제 알고리즘의 경우에는 보조 정리 4의 접근 방법을 이용한다. 제안된 알고리즘이 최소 신장 트리를 유지하는 것을 보장하기 위해서는 다음과 같은 두 가지 조건이 지켜지는 것에 대하여 증명하여야 한다. 첫 번째 조건은 알고리즘이 종료되었을 때, " branch" 상태인 간선들로 이루어지는 회로가 존재하지 않아야 한다는 것이다. 두 번째 조건은 알고리즘이 종료되었을 때, 네트워크에 존재하는 노드들이 연결 가능한 경우, 연결된 상태가 되어야 한다는 것이다.

정리 3 제안된 알고리즘은 회로를 만들지 않는다.

정리 4 제안된 알고리즘은 그래프가 연결 그래프가 가능한 경우에 연결 그래프를 보장한다.

4. 복잡도 분석 및 문제의 하한

메시지 복잡도는 인접한 노드들 사이에 교환하는 메시지의 전체 수를 의미한다. 메시지의 길이는 메시지 종류와 노드 식별자를 포함한 몇 가지 상수 정보를 초과하지 않으므로 $O(\log N)$으로 가능하다.

전체 알고리즘의 메시지 복잡도를 분석하기 위해서 각 단계에서 메시지 복잡도를 분석한다. 간선 추가 알고리즘과 간선 삭제 알고리즘의 경우 모두 보고 단계, 합계 단계, 초기화 단계는 메시지를 전송하는 노드가 네트워크의 전체 노드 수보다 클 수 없으므로 복잡도는 $O(N)$이다. 간선 추가 알고리즘에서의 유지 단계의 복잡도는 두 가지의 경우로 구분된다. 루트 노드가 최소 신장 트리를 유지하기 위하여 [12]에서 제안한 정적 네트워크에서 최소 신장 트리를 찾는 과정을 선택하였다면 알고리즘의 메시지 복잡도는 $O(N \log N + E)$이다. 루트 노드가 네트워크에 존재하는 회로들로부터 최대 가중치의 간선들을 제거하는 과정을 사용한다면 복잡도는 $O(kD)$이다. 알고리즘은 회로 내의 간선을 제거하는 과정을 k번 반복하고, 1개의 간선을 제거하는 비용은 트리의 지름 D에 비례하기 때문이다. 따라서 간선이 추가된 경우의 전체 알고리즘의 메시지 복잡도는 $\min(O(kD) + O(N), O(N \log N + E))$가 된다. 간선 삭제 알고리즘에서의 유지 단계는 Gallager 등이 제안한 알고리즘의 중간 과정과 동일하다. 모든 조각들이 동시에 유지 단계에 진입하지는 않지만 유지 단계에 진입한 조각의 노드는 오직 유지 단계에 진입한 조각의 노드로부터 메시지를 받을 수 있기 때문에 유지 단계에서의 메시지 복잡도는 $(k + 1)$개의 부분들이 모두 유지 단계에 진입한 경우만을 고려할 수 있다.

보조 정리 8 유지 단계에서 가장 많은 메시지 수를 갖는 알고리즘의 일정은 $(k + 1)$개의 조각들이 서로 같은 레벨을 갖는 경우에 관찰될 수 있다.

정리 5 제안된 알고리즘의 유지 단계의 메시지의 복잡도는 $O(N \log k + E)$이다.

증명 유지 단계는 $(k + 1)$개의 조각들로부터 알고리즘이 시작된다는 사실이다. 보조 정리 8을 통하여 알고리즘에서 가장 큰 메시지 복잡도를 관찰할 수 있는 경우는 $(k + 1)$개의 조각들이 같은 레벨을 갖는 경우이다. Gallager 등이 제안한 알고리즘에서 노드는 새로운 레벨에 도달한 경우에만 상수개의 메시지를 전달한다[8]. 따라서 각 노드가 보내는 메시지 수는 노드가 도달할 수 있는 레벨에 의하여 제한된다. $(k + 1)$개의 조각들의 레벨이 모두 같기 때문에 임의의 노드가 증가시킬 수 있는 레벨은 $O(\log k + 1)$이다. 외부 간선에 대한 검사는 네트워크에 존재하는 간선의 수 E를 초과할 수 없으므로 전체 메시지 복잡도는 $O(N \log k + E)$이다.

간선이 추가된 경우의 하한은 다음과 같이 증명된다. 보조 정리 1, 2, 3의 결과로부터 새로운 간선이 추가된 경우에 이전에 찾은 트리가 반드시 네트워크의 최소 신장 트리라는 것을 보장하지 못한다는 사실을 알 수 있다. 최소 신장 트리를 찾는 문제는 새로운 간선을 포함한 회로 내에서 가장 큰 간선을 결정하고 제거하는 문제로 유도할 수 있다.

보조 정리 9 회로 C에서 최대 가중치의 간선을 찾는 문제에 대한 메시지의 하한은 $\Omega(|C|)$이다.

증명 귀류법으로 증명한다. 네트워크 G에 대해서 $|C|$보다 적은 메시지 복잡도를 가지는 일정 S를 갖는 최대 가중치의 간선을 찾는 알고리즘 A가 존재한다고 가정한다. $|C|$보다 적은 비용을 갖는다는 것은 회로 C에서 사용하지 않는 간선이 존재한다는 것을 의미한다. S에서 사용하지 않는 간선에 인접한 두 노드를 M과 N로 놓는다. 모순을 유도하기 위하여 네트워크 G'을 만든다. G'은 일정 S에서 사용하지 않은 간선을 삭제하고 (그림 4.1)과 같이 G의 두 복사본을 연결하는 두 개의 간선을 추가하여 만든다. 네트워크 G'의 두 복사본인 양쪽의 부분 네트워크에서 간선들의 가중치는 G에서의 상대적인 순서와 같다. 네트워크 G'에서 양쪽의 부분 네트워크에서 동시에 일정 S로 동작한다면, 최대 가중치의 간선이 두 개 선택되므로 이는 모순이다.

정리 6 최소 신장 트리에 간선이 추가되었을 때, 최소 신장 트리를 찾는 문제에 대한 메시지의 하한은 $\Omega(D)$이다.

정리 7 최소 신장 트리에서 k개의 간선들이 삭제되었을 때, 최소 신장 트리를 유지하는 문제의 하한은 $\Omega(k \log k + E)$이다.

5. 결론 및 향후 과제

본 논문에서는 동적 네트워크에서 최소 신장 트리를 찾는 분산 알고리즘을 제안하였다. 동적 변화가 발생한 경우, 트리의 성질을 이용하여 동적 변화에 관련된 노드들에게 동적 변화가 발생했음을 알리고 서로 협력하여 최소 신장 트리를 찾는다. 네트워크 G의 노드의 수를 N, 간선의 수를 E, 트리의 지름을 D라고 놓았을 때, G에 k개의 간선이 추가되는 사건에 대하여 제안된 알고리즘은 $\min\{O(kD) + O(N), O(N \log N + E)\}$의 메시지 복잡도를 갖고, k개의 간선이 삭제되는 사건에 대하여 $O(N \log k + E)$의 메시지 복잡도를 갖는다. 제안된 알고리즘은 교착 상태를 발생시키지 않는다는 것을 무결성 증명을 통하여 규명하였고, 동적 네트워크에서의 최소 신장 트리를 찾는 문제에 대한 메시지 복잡도의 하한을 증명하였다.

현재 제안된 알고리즘은 시간 복잡도 면에 있어서 개선의 여지를 가진다. 그러므로 향후 연구과제로서 시간 복잡도를 줄이는데 초점을 맞추려 한다. 시간 복잡도를 개선하기 위해서는 Awerbuch[7]의 알고리즘에서처럼 조각의 크기에 따라 각기 다른 단계의 알고리즘을 적용함으로서 시간복잡도를 개선할 수 있을 것이다. 아직까지 동적 네트워크에서 최소 신장 트리를 찾는 문제에 대한 시간 복잡도의 하한은 연구되지 않았기 때문에, 이에 대한 연구도 함께 수행하려 한다. 또한, 동적인 변화를 반영하는 도중에 새로운 동적 변화가 발생하면, 이전의 변화를 모두 반영한 후에 새로운 동적 변화에 대응하는 제약을 가지기 때문에, 앞으로는 현재 반영하는 변화와 새로운 동적 변화를 모두 고려하는 알고리즘을 연구하려 한다.

참고문헌

[1] B. Awerbuch, *Complexity of Network Synchronization*, Journal of the ACM, Vol. 32, No. 4, October 1985, pp. 804-823.

[2] B. Awerbuch and R. Gallager, *Distributed Breadth-First-Search Algorithms*, IEEE Symposium on Foundations of Computer Science, October 1985, pp. 250-256.

[3] B. Awerbuch and S. Micali, *Dynamic Deadlock Resolution Protocols*, IEEE Symposium on Foundations of Computer Science, October 1986, pp. 196-207.

[4] K. Raymond, *A Tree-Based Algorithm for Distributed Mutual Exclusion*, ACM Transactions on Computer Systems, Vol. 7, No. 1, February 1989, pp. 61-77.

[5] Y. Dalal and R. Metcalfe, *Reserve Path Forwarding of Broadcast Packets*, Communications of the ACM, Vol. 21, No. 12, pp. 1040-1048, December 1978.

[6] B. Awerbuch and S. Even, *Efficient and Reliable Broadcast is Achievable in an Eventually Connected Network*, Proceedings of the ACM Symposium on Principles of Distributed Computing, pp. 278-281, August 1984.

[7] B. Awerbuch, *Optimal Distributed Algorithms for Minimum Weight Spanning Tree, Counting, Leader Election and related problems*, Proceedings of the ACM Symposium on Theory of Computing, pp. 230-240, May 1987.

[8] R. Gallager, P. Humblet and P. Spira, *A Distributed Algorithms for Minimum Weight Spanning Trees*, ACM Transactions on Programming Languages and Systems, Vol. 5, No. 1, pp. 66-77, January 1983.

[9] G. Frederickson and N. Lynch, *The Impact of Synchronous Communication on the Problem of Electing a Leader in a Ring*, Proceedings of the ACM Symposium on Theory of Computing, pp. 493-503, April 1984.

[10] F. Chin and H. Ting, *O(V log V + E) messages Distributed Algorithm for Minimum Weight Spanning Trees*, IEEE Symposium on Foundations of Computer Science, pp. 257-266, October 1985.

[11] E. Gafni, *Improvements in the complexities of two message-optimal algorithms*, Proceedings of the ACM Symposium on Principles of Distributed Computing, pp. 175-185, August 1985.

[12] M. Faloutsos and M. Molle, *Optimal Distributed Algorithm for Minimum Spanning Trees Revisited*, Proceedings of the ACM Symposium on Principles of Distributed Computing, pp. 231-237, August 1995.

[13] Giuseppe Amato, Giuseppe Cattanco, Giuseppe F. Italiano, *Experimental Analysis of Dynamic Minimum Spanning Tree Algorithms (Extended Abstract)*, Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, pp. 314-323, January 1997.