DOI: 10.3745/KIPSTA.2010.17A.5.213

스타이너 트리를 이용한 입력 선분의 연결

김 준 모[†] · 김 인 범[††]

요 약

본 논문에서는 스타이너 트리를 이용하여 최소 길이로 입력 선분들을 모두 연결하는 방법을 제안한다. 선분은 통신선, 도로 및 철도망 또는 움직이는 물체의 궤적 등으로 변환될 수 있다. 본 논문에서 제안된 방법은 이러한 선분들을 최소 비용으로 연결하는 응용 등에 활용가능하다. 입력 선분의 수와 각 선분 당 최대 연결 선분의 수를 입력 인자로 설정한 실험 에서, 본 논문에서 제안된 방법은 최소 신장 트리를 이용한 방법과 비교하여 연결 생성 시간은 평균 192.0% 증가하였으나, 연결 길이는 평균 6.8%에 감소하였다. 이는 연결 방법을 찾는 시간보다는 연결 길이를 단축하는 것이 더 중요한 응용에 제안된 방법이 유용할 수 있음을 보인다.

키워드 : 최소 스타이너 트리, 선분 최소 신장 트리, 입력 선분, 연결선

Mechanism for Connecting Input Edges Using Steiner Tree

Joonmo Kim[†] · Inbum Kim[††]

ABSTRACT

In this paper, a mechanism connecting all input edges with minimum length through Steiner tree is proposed. Edges are convertible into communication lines, roads, railroads or trace of moving object. Proposed mechanism could be applied to connect these edges with minimum cost. In our experiments where input edge number and maximum connections per edge are used as input parameters, our mechanism made connection length decrease average 6.8%, while building time for a connecting solution increase average 192.0% comparing with the method using minimum spanning tree. The result shows our mechanism might be well applied to the applications where connecting cost is more important than building time for a connecting solution.

Keywords : Steiner Minimum Steiner Tree, Edge Minimum Spanning Tree, Input Edge, Connecting Line

1. 서 론

선분(edge)은 정적으로 연속적으로 인접한 노드들의 집합으로 해석될 수 있다. 이것은 통신 네트워크에서의 통신선으로 적용할 수 있으며, 동적으로 공간 또는 평면상에 위치한 특정 노드의 예측 가능한 이동궤적이 될 수 있다. 연결선(connecting line)은 이러한 선분(edge)을 서로 연결하는 선으로, 통신선 또는 경로가 이미 예정된 동적 장치들의 효율적인 연결에 활용될 수 있다. 입력 선분(input edge)들의 연결은 동적 라우팅, ad-hoc 네트워크, 회로 설계, 항로 결정, 도로 연결 등의 분야에 응용될 수 있다. 이러한 응용들의 어떤 경우에는, 연결 방법을 찾는 시간을 단축하는 것보다는 입력 선분들을 최소의 비용으로 연결하는 것이 중요할 수 있다. 본 논문에서는 이러한 응용을 위해서 스타이너 트리를 이용하여 입력 선분들을 최소 비용으로 연결하는 방법을 제안한다. 제안된 방법에서 입력 선분은 노드로, 입력 선분의 연결은 연결선으로 변환하여 최소 신장 트리를 생성하고, 이 트리에 대한 스타이너 포인트를 생성하여 스타이너 그래프를 생성한 후, 이것에 대한 후처리 작업을 시행하여 입력 선분들에 대한 연결선을 생성한다.

스타이너 트리는 네트워크 길이를 최소화하는 문제 중의 하나로, 평면상에 존재하는 노드들이 있을 때, 스타이너 포인트라는 점들의 집합을 새로 추가하여 주어진 노드들을 모두 연결하는 트리이다. 이들 중에서 최소 비용의 트리가 최소 스타이너 트리(Steiner Minimum Tree)이며, 이 문제는 NP-Complete 문제에 포함되는 것으로 알려져 있다[1-3]. 최소 스타이너 트리는 다항적 시간(polynomial time) 내에서 최적화된 해를 제공하는 최소 신장 트리보다 연결 비용을 줄일 수 있으나, 다항적 시간 내에 문제에 대한 해를 구할 수 없으므로, 실제 활용을 위해서는 적절한 휴리스틱을 통한 근사 최소 스타이너 트리를 생성하여 적용해야 할 것이다.

본 논문의 구성은 다음과 같다. 2장은 본 논문과 관련된

† 정 회 원 : 단국대학교 컴퓨터학부 조교수
†† 정 회 원 : 김포대학 IT학부 부교수(교신저자)
논문접수 : 2009년 12월 18일
수 정 일 : 1차 2010년 3월 25일
심사완료 : 2010년 4월 19일

연구 내용이고, 3장은 제안하는 연결 방법에 대한 기술이다. 4장은 제안된 방법에 대한 타당성을 보이기 위해, 이를 구현하고 입력 인자를 변경하면서 입력 선분을 연결하는 실험 및 그에 대한 분석이고 5장은 결론이다.

2. 관련 연구

본 논문에서 제안하는 방법인 입력 선분을 스타이너 트리를 이용하여 연결하는 것은 입력 노드를 대상으로 한 전통적인 최소 신장 트리의 생성과 관련이 있다. 일반적인 환경과 입력에서 트리의 길이가 최적화된 최소 신장 트리는 Kruskal 혹은 Prim의 알고리즘을 이용하여 얻을 수 있다[4-6]. 신장 트리(Spanning Tree)란 이차원 평면에 주어진 N개의 노드들을 서로 교차하지 않게 연결하는 트리이다. 주어진 노드와 입력 연결선이 있을 때 일부 입력 연결선을 이용하여 노드들을 모두 연결하는 최소 비용의 트리를 최소 신장 트리(Minimum Spanning Tree)라고 한다. N개의 노드를 연결하기 위해서는 N-1개의 연결선이 필요하다.

선분은 현실 세계의 여러 형태의 연결로 모델링될 수 있다. 이것은 유선 통신선이나 도로, 철도와 같은 정적인 형태뿐 아니라, 항로, 이동 물체의 궤적으로 적용될 수 있다. 이 선분들을 최소 신장 트리를 이용하여 연결하는 방법이 연구되어 발표되었다[7].

네트워크를 구성하는 연결들의 길이를 최소화하는 문제는 최적화 문제 중의 하나로 연구되었다[8]. 이들 중에서 평면상에 주어진 점들이 있을 때, 특정 노드 S와 입력 노드들을 연결하여 최소 길이의 트리를 얻을 수 있는 S를 찾는 문제는 수학자 Jakob Steiner에 의해 연구되었다. 또한 여러 개의 노드들을 최소 길이로 모두 연결하는 문제는 스타 형 토폴로지에서 새로운 하나의 노드를 찾아 이것과 다른 입력 노드들과 연결하는 것이다. 이를 Steiner Star라고 부른다. 한 개가 아닌 여러 개의 스타이너 포인트들을 도입하여 모든 입력 노드들을 연결하는 최단 길이의 네트워크를 찾는 방법을 Courant와 Robbins 등이 연구하였다. 이러한 네트워크를 최소 스타이너 트리(Steiner Minimum Tree)라고 한다[3]. 이 문제는 NP-Complete 문제에 포함되는 것으로 알려져 있다[1-3]. 최소 스타이너 트리 문제는 근사 트리를 구하여 근사 비율을 높이는 연구와 근사 트리를 구하는 휴리스틱에 관한 연구들이 수행되었다. J.Kim 등은 GOSST(Grade of Services Steiner Minimum Tree) 문제에 최소 스타이너 트리를 이용하여 (1+ε) 근사 트리를 다항적 시간 내에 얻을 수 있음을 증명하였다[9]. 또한 원격 검침 시스템을 구성하는 검침기, 중계기, 집중기의 배치 및 연결을 근사 스타이너 트리를 이용한 방안이 제안되었고, 이를 최소 신장 트리를 이용한 방법과 비교한 연구가 시행되었다[10]. 비 방향 스타이너 트리 문제를 해결하기 위한 연구로, 그래프 축소 규칙을 이용하여, 최적 해에 불필요한 노드들과 연결선들을 제거한 후, Prim의 알고리즘을 조합한 max-min ant colony optimization 방법을 적용한 연구가 발표되었다[11]. 멀티캐스팅 문제를 스타이너 트리를 이용하여 해결하려는 연구도 시도되었는데, 이 연구들에서는 멀티캐스팅 문제와 순회 판매원 문제의 차이를 분석하여, 기존의 개미 시스템(Ant System)을 변경한 엘리트 에이전트에 의한 개미 멀티캐스트 라우팅 모델과 멀티 캐스트 통신에서의 QoS(Quality of Service)를 처리할 수 있는 다중 제약 멀티 캐스트 처리 알고리즘이 발표되었다[12-13]. 또한 센서 네트워크의 효율적인 라우팅을 위하여 센서노드들을 최적으로 상호 연결하는 배치 문제를 스타이너 트리를 활용하여 해결할 수 있다는 전제에서, 센서 네트워크에서의 물리적인 특성이, 일반적인 그래프 노드 연결 문제의 범위를 축소할 수 있다는 점을 주장하며 이를 기반으로 실행시간과 최적 치에 대한 근사비율이 연구되어 발표되었다[14]. 본 연구와 유사한 연구 분야인, 선분 연결 관련 연구는 많이 발표되지 않았는데, 그 중에서 입력 선분의 연결 문제를 변형된 최소 신장 트리와 입력 선분 상의 임의의 가상 노드인 포탈을 이용하여 선분을 사용자의 요구에 따라 연결하려는 연구가 시도되었다. 이 연구에서 선분 연결 방법의 결정 시, 연결 방법을 찾는 시간과 연결 비용을 조절하면서 특정 응용에서 관심 있는 방법을 선택할 수 있는 방안을 제안하였다[7]. 현재까지 발표된 많은 근사 최소 스타이너 트리를 구성할 수 있는 휴리스틱 중에서, 정삼각형의 외접원을 활용하여 스타이너 포인트를 구하는 휴리스틱이 대표적인 방법이다[15].

본 논문에서 제안하는 방법은 <표 1>에 기술되어 있는 최소 신장 트리를 이용한 선분 연결하는 방법과 비교된다

〈표 1〉 최소 신장 트리를 이용한 입력 선분의 연결 알고리즘

단계	내용
1	각 입력 선분을 노드로, 각 선분 사이의 연결을 연결선으로 변환하고, 각 연결선의 길이를 가중치로 변환한다. 각 노드의 flag를 false로 저장한다. 본 논문에서 제안하는 선분 연결 문제는 이 단계에서 노드와 연결선, 그리고 가중치로 구성된 최소 신장 트리 생성 문제로 변환된다.
2	임의의 한 노드 S를 선정하고, 이 노드와 연결된 연결선들 중에서 최소 가중치를 가지는 연결선 ST를 선택하고, 이 선분을 구성하는 두 노드인 S와 T의 flag를 true로 변경한다. 선택된 선분 ST는 집합 MST에 저장한다.
3	flag값이 서로 다른 두 노드로 구성된 연결선들 중에서 최소 가중치를 가지는 선분 S′T′을 선택하고, false 인 T′의 flag를 true로 변환한다. 연결선 S′T′을 집합 MST에 저장한다.
4	모든 노드들의 flag가 true가 될 때까지 단계 3을 반복한다.
5	최소 신장 트리를 구성하는 MST의 모든 원소들을 이용하여 선분 연결 문제로 변환한다. 즉, 연결선들을 구성하는 두 노드를 원래의 두 입력 선분으로, 두 노드 사이의 연결선은 이들 입력 선분들의 연결로 변환시킨다.

[7]. 또한 이 최소 신장 트리 생성 방법은 제안된 방법의 초기 단계에서 일부 사용된다. 최소 신장 트리 생성은 Prim의 알고리즘을 변형하였는데, 이 알고리즘은 주어진 노드들과 이들 사이의 입력 연결선을 이용하여 최소 비용의 신장 트리를 생성할 수 있다. 본 논문에서 제안하는 문제는 입력 선분들을 최소 비용으로 연결하는 것이므로 최소 신장 트리 생성 알고리즘을 바로 응용할 수 없고 적절한 변환이 필요하다. 즉 입력 선분을 노드로, 선분 연결을 연결선으로 변환한 후에 그 연결선의 길이를 가중치로 이용하여 순수한 Prim의 알고리즘을 적용한다. 최소 신장 트리를 이용한 입력 선분의 연결 과정은 <표 1>과 같고, 이에 대한 간단한 예는 (그림 3)의 (a), (b), (c)에 나타나 있다.

3. 본 론

<표 2>는 본 논문에서 제안하는 스타이너 트리를 이용한 입력 선분의 연결 과정을 나타내는 알고리즘이다. 이 알고리즘에서 주어진 입력 선분과 그 연결에 대하여 2장의 <표 1>에서 기술한 변형된 Prim의 알고리즘을 응용한 선분 최소 신장 트리를 먼저 생성하고 이를 기반으로 하여 스타이너 포인트를 생성하고 이를 연결하여 스타이너 그래프를 생성한다. 이 스타이너 그래프를 구성하는 노드와 연결들을 노드와 연결선으로 입력하여 새로운 최소 신장 트리를 생성한 후에, 필요 없는 연결선의 제거 및 새로운 최단 거리의 연결선을 생성하는 후처리 작업을 시행하고 이에 대한 결과를 최종 연결로 결정한다. 본 논문에서 다루는 문제는 입력 선분의 연결 문제는 기존의 최소 신장 트리 생성과는 다르게, 입력은 노드가 아닌 선분이고, 따라서 노드간의 연결 대신 선분간의 연결이므로 적절한 변환이 필요하다. 본 논문에서 두 선분 사이의 연결은 두 선분 사이의 최단 연결로 정의한다.

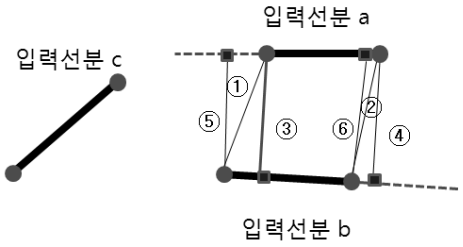
위 과정의 실행시간을 분석하면 다음과 같다. 단계 1에서 전체 입력 선분의 수를 E, 선분 간 연결의 수를 L로 했을 때, Prim의 최소 신장 생성 알고리즘을 이용하면 O(ElogE) 시간 내에 선분을 노드로 변환한 최소 신장 트리의 생성 가능하다. Prim의 알고리즘은 우선순위 큐(priority queue)를 피보나치 힙(Fibonacci heap)으로 구현하면 실행시간은 O(L+ElogE)이고, $L \ll E^2$인 환경에서는 O(ElogE)이다[6]. 최소 신장 트리는 (E-1)개의 연결로 구성되어 있다. 단계 2에서 최소 신장을 구성하는 연속적인 연결의 수는 스타(star) 토폴로지일 때 최대가 되며, 이것은 $_{E-1}C_2=O(E^2)$ 이다. 따라서 생성되는 스타이너 포인트의 수는 $O(E^2)$이다. 단계 3에서 다루어야 할 노드의 수는 기존의 입력 선분의 수 E에 생성된 스타이너 포인트의 수를 합한 값인 $E+O(E^2)=O(E^2)$이고, 연결선의 수는 $(E-1)+3\times O(E^2)=O(E^2)$이다. 따라서 이것에 대한 최소 신장 트리 생성 시간은 $O(E^2\log E^2)=O(E^2\log E)$이다. 단계 4에서 차수가 1인 스타이너 포인트를 찾고 이를 처리하는 시간은 스타이너 포인트의 수와 동일한 $O(E^2)$이고, 단계 6에서 스타이너 포인트와 입력 선분간의 최단거리를 검사하고 처리하는 시간은 $O(E^2)$이다. 따라서 본 논문에서 제안하는 방법의 실행 시간은 $O(E^2\log E)$ 이다. 그러나 입력 선분과 입력 연결에 대해 생성되는 최소 신장 트리의 토폴로지가 스타 형일 가능성은 적으므로, 이보다는 빠른 시간 내에 실행이 가능할 것이다.

(그림 1)은 세 입력 선분 a, b, c 사이의 연결 생성 과정을 보인다. 먼저 두 선분 a, b의 연결에서, 입력 선분 a의 양 끝 점에서 선분 b에 대한 수선 ③, ④를 생성한다. 이 때 수선 ④는 입력 선분 b의 연장선에서 교점이 위치하므로 이것과 가장 가까운 입력 선분 b의 끝점을 연결한 연결 ②를 생성한다. 마찬가지로 입력 선분 b의 양 끝 점에서 선분 a에 대한 수선⑤, ⑥을 생성한다. 이 때 수선 ⑤는 입력 선분 a의 연장선에서 교점이 위치하므로 이것과 가장 가까운 입력 선분 a의 끝점을 연결한 연결 ①를 생성한다. 각 선분의 연장선에서 교점을 생성하는 수선 ④, ⑤를 제외한 4개의 수선 ①, ②, ③, ⑥ 가운데 가장 작은 길이의 수선이 두 입력 선분 a, b의 연결로 결정된다. 같은 방법으로 두 입력 선분 a와 c, b와 c의 연결을 생성한다.

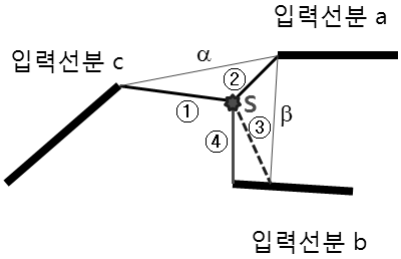
(그림 2)는 제안하는 스타이너 트리를 이용한 선분 연결 알고리즘의 단계 5에서 실행되는 후처리 작업에 관한 내용이다. 세 입력 선분 a, b, c 사이에서 생성된 스타이너 포인트 S는 연결 ①, ②, ③을 이용하여 각각 입력 선분 a, b, c를 연결하고 있다. 그러나 이 연결이 S와 각 입력 선분을 최단거리로 연결한다고 장담할 수 없다. 그 이유는 스타이너 포인트는 각 선분들 사이의 최단 거리인 연결선을 입력

〈표 2〉 스타이너 트리를 이용한 입력 선분 연결 알고리즘

단계	내용
1	<표 1>에 기술한 것과 같이, 입력 선분은 노드로, 입력 연결은 연결선으로 변환한 후, 이를 이용하여 최소 신장 트리를 생성한다.
2	최소 신장 트리에서 연속적으로 연결된 세 노드로 이루어진 모든 부분 트리들에 대해 각각 스타이너 포인트를 생성하고, 세 노드와 스타이너 포인트를 연결하여 최소 길이의 스타이너 그래프 생성한다.
3	스타이너 그래프를 구성하는 노드, 스타이너 포인트들과 연결선, 그리고 연결선의 길이들을 입력으로 하여 새로운 최소 신장 트리의 생성한다.
4	단계 3에서 생성된 트리에서 스타이너 포인트의 차수가 1인 연결선들을 찾고 이 연결선과 해당 스타이너 포인트를 모두 제거한다.
5	스타이너 포인트가 아닌 노드들을 원래의 입력 선분으로, 노드간의 연결선을 선분간의 연결로 변환한다. 또한 스타이너 포인트와 노드간의 연결선은 스타이너 포인트를 노드가 대표하는 선분과 연결한 후, 후처리 작업을 시행한다. 후 처리 작업은 스타이너 포인트와 해당 입력 선분간의 거리가 최단이 아니면, 기존의 연결을 제거하고, 이를 사이의 최단 연결을 생성하는 것이다.
6	단계 5에서 생성된 연결 구조를 최종 결과로 출력한다.



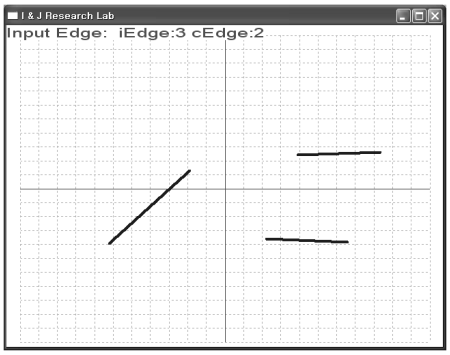
(그림 1) 선분의 연결 생성 과정



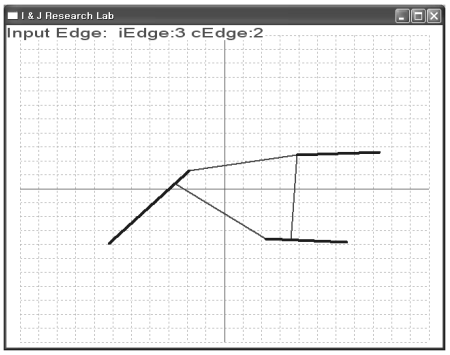
(그림 2) 후처리 작업 과정

으로 생성된 것이기 때문이다. 예를 들어 (그림 2)에서 스타이너 포인트 S는 연결선 α와 β를 이용하여 생성된 것이다. 따라서 후처리 작업에서는 스타이너 포인트와 연결된 입력 선분들에 대해 최단 거리의 연결인지의 여부를 검사 한 후에, 최단 거리가 아닌 경우에는 해당 연결을 삭제하고 최단 거리의 연결을 새로이 생성한다. (그림 2)에서 기존의 연결 ③을 제거하고 새로운 최단 연결 ④를 새로이 추가한다.

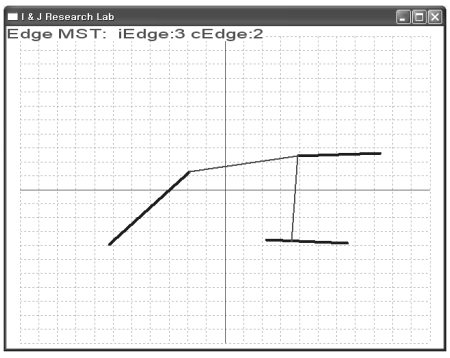
(그림 3)의 (a)와 (b)는 스타이너 트리를 이용한 입력 선



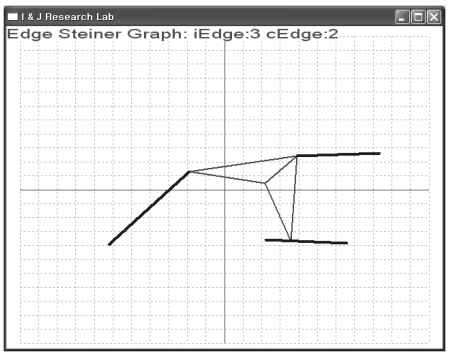
(a) 3개의 입력 선분



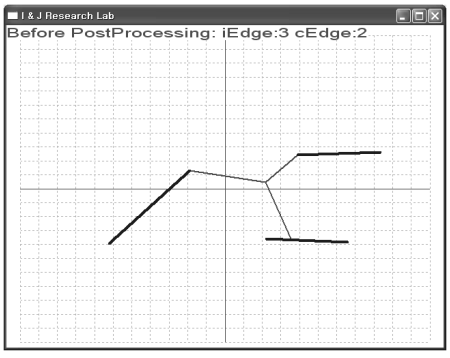
(b) 각 선분에서 최대 2개 다른 입력 선분의 연결



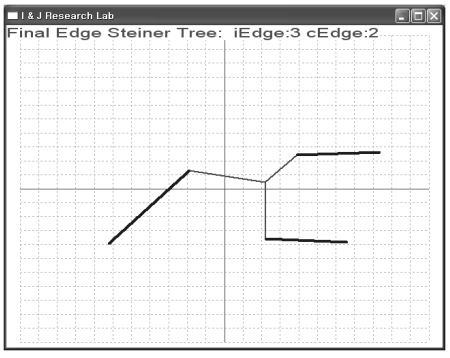
(c) 최소 신장 트리를 이용한 입력 선분의 연결



(d) 입력 선분과 입력 연결에 대한 선분 스타이너 그래프



(e) 불필요한 연결이 삭제된 스타이너 트리



(f) 후처리 작업 후의 입력 선분 연결

(그림 3) 스타이너 트리를 이용한 입력 선분의 연결 과정

분의 연결의 예를 보이기 위한 3개의 입력 선분과 한 선분 당 최대 2개의 연결을 보이고 있다. 그림 (c)는 이들 선분과 연결들에 대하여 적절히 변환한 후, 선분 최소 신장 트리 생성 알고리즘을 이용한 연결의 모습이다. 그림 (d)는 그림 (c)의 선분 최소 신장 트리 연결을 적절히 이용하여 스타이너 포인트를 생성하고, 생성된 스타이너 포인트를 입력 선분과 연결한 스타이너 그래프의 모습이고, 불필요한 연결이 삭제된 스타이너 트리를 이용한 연결이 그림 (e)에 나타나 있다. 마지막으로 후처리 작업을 반영한 세 입력 선분간의 최종 연결이 그림 (f)에 있다.

4. 실험 및 결과 분석

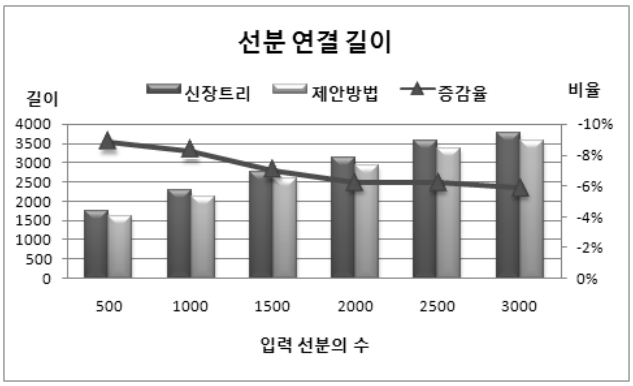
본 연구를 위해 사용된 실험 인자는 입력 선분의 수와 한 선분에서 최대로 연결할 수 있는 다른 입력 선분의 수이다. 관찰 결과는 입력 선분들을 연결한 연결선의 길이와 입력 선분의 연결 방법을 생성하는 시간이다. 연결 길이와 생성 시간은 <표 1>의 최소 신장 트리 알고리즘으로 생성된 선분 최소 신장 트리와 비교된다. 실험을 위해 무작위로 생성된 입력 선분의 수는 500, 1000, 1500, 2000, 2500, 3000개이다. 각 선분의 최대 연결의 수는 입력 선분 수의 20%, 40%, 60%, 80%, 100%로 하였는데, 예를 들어 입력 선분이 500개의 경우, 최대 연결 20%는 각 선분에서 최대 100개의 다른 선분과 연결할 수 있다는 것을 의미한다. 최대 연결 100%는 각 입력 선분에서 최대 가능한 연결 수, 즉 입력선분 수-1로 정의하였다. 각 입력 선분들은 2차원 평면상에서, -10.0과 10.0 사이의 x, y 좌표 값을 무작위로 선택하여 이를 선분의 시작점으로 하고, 이를 중심으로 하여 0과 360도 사이의 각도를 가지는, 0.2와 0.8 사이의 거리에 위치하는 노드를 선분의 끝점으로 하여 시작점과 연결하여 생성된다. 생성된 각 입력 선분들은 서로 교차하지 않게 하고 각 선분들 사이의 떨어진 거리가 0.15이상이 되도록 제한을 가하였다. 실험 환경은 Intel 1.83 GHz (T5600) 프로세서와 1기가 메모리 램의 랩탑 컴퓨터이고, 본 논문에서 제안하는 알고리즘을 C++로 구현하였다.

4.1 선분 연결선의 길이 및 길이 증감율

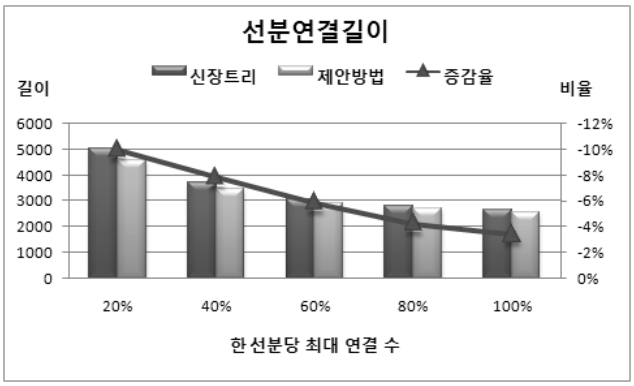
(그림 4)는 실험 인자인 입력 선분의 수와 한 선분 당 최대 연결 수를 변경하면서 본 논문에서 제안하는 방법에 의해 생성된 입력 선분의 연결선의 길이에 대한 분석 결과이다. (그림 4)의 (a)와 (b)에서 확인할 수 있는 것처럼 제안된 방법을 이용한 입력 선분의 연결은 최소 신장트리를 이용한 선분 연결 방법에 비해 길이는 감소된다. (그림 4)의 (a)에서 입력 선분의 수가 커질수록 제안하는 방법의 연결 길이의 절감율은 점차 떨어지지만, 그 변화량은 그리 크지 않음을 확인할 수 있다. 입력 선분의 수가 500일 때, 최고인 약 8.9%의 연결 길이 절감율을 보이고 입력 선분의 수가 3000인 경우에는 연결 길이의 절감율이 최저인 약 5.9%이다. 입력 선분의 수가 많으면, 한정된 공간에서 최소 신장 트리를 이용한 연결 방법보다 더 단축된 연결 방법을 찾을 수 있는 가능성이 감소함을 확인할 수 있다. (그림 4)의 (b)는 한 입력 선분에서 연결 가능한 최대 연결 수를 변경하면서, 제안된 방법으로 생성된 연결의 길이를 분석한 결과로서, 한 선분 당 최대 연결 수가 적을수록 연결 길이의 절감율은 높다. 한 선분 당 최대 연결 수가 20%인 경우에는 최고인 약 10.0%의 연결 길이의 절감율을 보이고, 최대 연결 수가 100%인 경우에는 약 3.4%의 연결 길이의 절감율을 나타낸다. 이는 선분간의 연결이 완전 연결에 근접할수록 최소 신장 트리를 이용한 연결 방법보다 더 단축된 연결방법을 확률적으로 찾기 어렵기 때문이다. 모든 입력 환경에 대하여 제안된 방법을 이용한 선분 연결 길이는 최소 신장 트리를 이용한 방법에 비해 평균 6.8% 감소되었다.

4.2 선분 연결선 생성 시간 및 시간 증감율

(그림 5)는 실험 입력 인자인 입력 선분의 수와 한 선분 당 최대 연결 수를 변경하면서, 본 논문에서 제안하는 방법으로 입력 선분의 연결선을 생성할 때, 필요한 실행시간에 대한 분석결과이다. (그림 5)의 (a)에서, 입력 선분의 수가

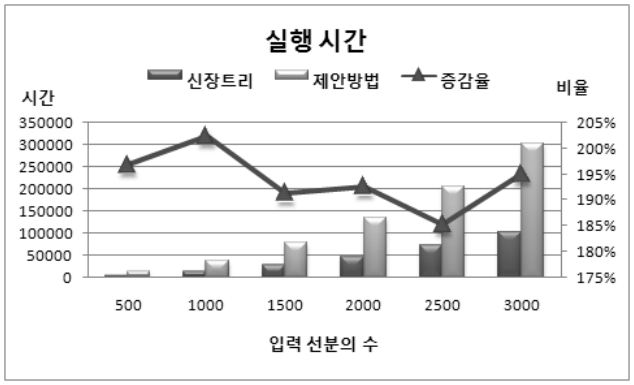


(a) 입력 선분 수의 변화

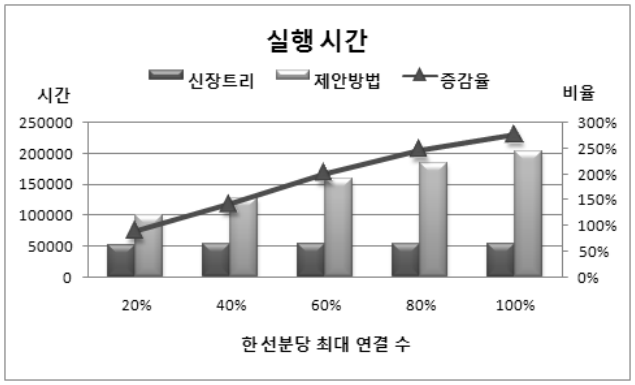


(b) 한 선분 당 최대 연결 수의 변경

(그림 4) 제안된 방법으로 생성된 입력 선분의 연결선 길이



(a) 입력 선분 수의 변화



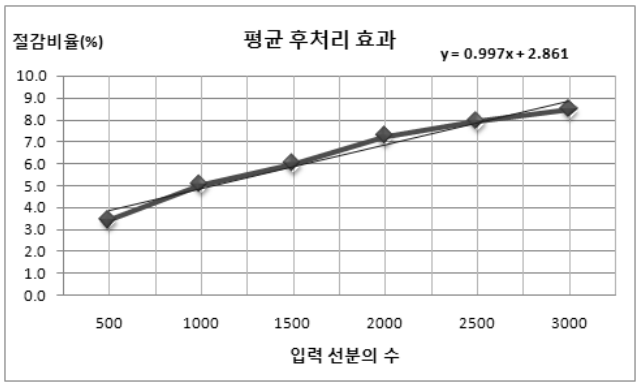
(b) 한 선분 당 최대 연결 수의 변경

(그림 5) 제안된 방법의 선분 연결선 생성 시간

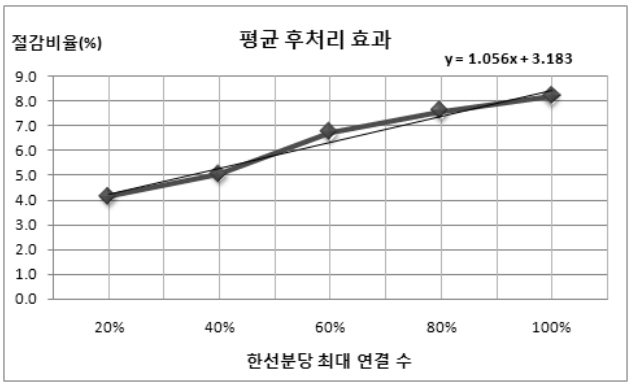
커질수록 제안된 방법과 최소 신장 트리를 이용한 방법 모두 실행 시간이 증가되고, 제안 방법이 좀 더 많은 시간이 소요됨을 보인다. 그러나 입력 선분의 수에 따른 제안된 방법의 실행 시간 증감율은 일정한 규칙을 보이지 않는다. (그림 5)의 (b)는 한 선분 당 최대 연결 수를 변경하며 선분 연결을 생성한 실험의 결과인데, 최소 신장 트리를 활용한 방법은 최대 연결 수를 증가시켜도 실행 시간의 변화가 없지만, 제안된 방법은 계속 증가함을 확인할 수 있다. 또한 제안된 방법은 한 노드 당 최대 연결의 수를 크게 하면, 실행 시간의 증가율이 높아져, 20%인 경우에는 실행시간이 최소 신장 트리를 사용한 연결보다 약 90.6%의 증가이지만, 완전 연결(100%)인 경우에는 약 276%가 증가함을 보인다. 모든 입력 환경에 대하여 제안된 방법을 이용한 선분 연결의 생성을 위한 실행 시간은 최소 신장 트리를 이용한 방법에 비해 평균 192% 증가되었다.

4.3 후처리 효과

(그림 6)은 본 논문에서 제안하는 후처리 효과에 대한 실험 결과이다. 실험 인자인 입력 선분의 수와 한 선분 당 최대 연결 수를 변경하여 생성되는 연결들의 길이 절감 비율을 관찰하였다. 후처리 절감 비율은 후처리를 시행함으로 절약되는 길이의 비율이다. (그림 6)의 (a)에서처럼 입력 선분의 수가 커질수록 후처리 효과가 잘 나타나, 연결 길이의 절감 비율이 높아짐을 알 수 있다. 입력 선분의 수가 500인 경우 약 3.5%의 연결 길이의 절감율을 보이고, 3000 인 경우, 평균 약 8.5%의 연결 길이 절감율을 보인다. (그림 6)의 (b)는 한 선분 당 최대 연결 수를 변경하는 환경에서의 후처리 효과에 대한 분석 결과로, 한 선분 당 최대 연결 수가 많을수록 후 처리 효과가 잘 나타남을 알 수 있다. 노드 간의 연결이 20%인 경우에는 후 처리 효과가 평균 4.2%이고, 완전 연결(100%)인 경우에는 약 8.2%임을 보인다. 모든 입력 환경에 대하여 제안된 방법의 후처리 작업을 통한 길이 절감율은 평균 약 6.4%이다. 그러나 후처리 작업을 위해서는 추가적인 계산 작업을 수행해야 하므로 실행시간의 증가가 발생하는데, 입력 선분의 수가 1000이고 한 선분 당 최대 연결 수가 40%인 환경의 실험에서 최대 0.22%의 추가 실행 시간이 요구되었고, 전체적으로는 평균 0.014%의 실행시간이 추가되었다.



(a) 입력 선분 수의 변화



(b) 한 선분 당 최대 연결 수의 변경

(그림 6) 제안된 방법의 후처리 효과에 관한 실험 결과

5. 결 론

본 논문에서 스타이너 포인트를 적절히 생성하고 이를 이용한 스타이너 트리를 활용하여 최소 비용으로 입력 선분들을 연결하는 방법을 제안한다. 본 논문에서 다루는 입력 선분은 통신 네트워크에서의 통신선 또는 동적 객체의 정해진 이동궤적으로 간주될 수 있다. 따라서 입력 선분 연결 문제는 네트워크에서의 라우팅, ad-hoc 네트워크, 회로 설계, 항로 결정, 도로 연결 등에 적용될 수 있다. 제안된 방법은 비교 대상인 선분 최소 신장 트리에 비해 입력 선분의 연결 생성 시간을 증가시켰지만, 그 연결 길이를 감소시켰다.

최소 스타이너 트리는, 다항적 시간 (Polynomial Time) 내에 최적의 해를 구할 수 있는 최소 신장 트리보다 연결 비용은 적지만, 이를 구성하는 것은 비 다항 적 시간 (Non-Polynomial Time) 문제에 포함되므로 이를 위한 알고리즘은 오랫동안 발표되지 못했다. 입력 선분들을 연결하기 위해, 본 논문에서 제안하는 방법은 먼저 입력 선분과 입력 선분간의 연결을 적절히 변환하여 선분 최소 신장 트리를 생성하고, 이 트리의 연속적으로 연결된 세 개의 노드로 구성된 모든 부분 트리에 대하여 스타이너 포인트를 생성하고 이들을 연결한 스타이너 그래프를 생성한다. 이 그래프에서 불필요한 연결을 삭제하고 후처리 작업을 통해 좀 더 연결 비용을 절감한다. 실험에서, 본 논문에서 제안된 방법은 유사 연구인 입력 선분의 연결 문제를 최소 신장 트리를 이용하여 해결한 선분 최소 신장 트리 방법[7]과 비교하여 입력 선분을 연결하는 연결선을 생성하는 시간은 192.0% 증가시켰으나, 선분 연결선의 길이는 평균 6.8% 감소시켰다. 이는 본 논문에서 제안하는 방법이 연결 생성 시간보다는 연결 비용의 절감이 더 중요한 응용에 잘 적용될 수 있음을 보인다.

향 후 연구는, 매우 많은 입력 선분들을 연결해야 하는 응용에서, 신속하게 연결 방법을 찾기 위해 다항 적 시간 근사구조 (Polynomial Time Approximation Scheme)의 적용에 관하여 연구할 예정이다[9]. 이 다항 적 시간 구조는 NP 문제와 같은 현실 세계에서 계산하기 힘든 문제에 대하여, 그 문제를 작은 기본 단위로 세분화시키고, 이를 Brute-force 방법으로 해결한 후, 그 하위 단계의 결과를 상위 단계에서 통합하여 문제를 해결하는 방법이다. 이러한 시도를 통해 현실 세계에서 당면할 수 있는 매우 많은 수의 통신선 연결과 같은 문제에 대하여 스타이너 트리를 이용한 연결 비용의 절감뿐만 아니라, 연결 생성의 시간적 측면에서, 좀 더 개선된 방법을 제안할 수 있을 것이다.

참 고 문 헌

[1] F.K. Hwang, D.S. Richards and P. Winter, “The Steiner Tree Problem,” Annals of Discrete Mathematics, Vol.53, North-Holland, 1992.

[2] W. Shi and C. Su, “The Rectilinear Steiner Arborescence Problem is NP-Complete,” Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, pp.780-787, 2000.

[3] http://en.wikipedia.org/wiki/Steiner_tree, December, 2009.

[4] http://en.wikipedia.org/wiki/Minimum_spanning_tree, December, 2009.

[5] R.L. Grahan and P. Hell, “On the History of the Minimum Spanning Tree Problem,” Annals of the History of Computing, Vol.7, No.1, pp.43-57.

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, Introduction to Algorithm, 2ndEd., MITPress, 2001.

[7] 김인범, 김수인, “선분상의 포탈을 이용한 근사 선분 최소 신장 트리의 생성”, 정보처리학회논문지, 12월, 2009.

[8] A. Hayrapetyan, C. Swamy and E. Tardos, “Network Design for Information Networks,” Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp.933-942, 2005.

[9] J. Kim, M. Cardei, I. Cardei and X. Jia, “A Polynomial Time Approximation Scheme for the Grade of Services Steiner Minimum Tree Problem”, Journal of Global Optimization Vol.24, pp.437-448, 2002.

[10] 김재각, 김인범, 김수인, 원격 검침 시스템에서 근사 최소 스타이너 트리를 이용한 집중기 및 중계기의 효율적인 배치와 연결, 한국통신학회논문지: 네트워크 및 서비스, Vol.34 No.10, pp.994-1003, 2009.

[11] 서민석, 김대철, “스타이너 트리 문제를 위한 Mar-Min Ant Colony Optimization”, 경영과학, Vol.26, No.1, pp.65-76, 2009

[12] 이승관, “멀티캐스트 라우팅 문제 해결을 위한 엘리트 개미 시스템”, 한국컴퓨터정보학회논문지, Vol.13, No.3, pp.147-152, 2008.

[13] 이성근, 한치근, “다중 제약이 있는 멀티캐스트 트리 문제에 관한 연구”, 한국인터넷정보학회논문지, Vol.5, No.5, pp.129-138, 2004.

[14] 김준모, “센서 네트워크 구축에서의 Combinatorial 기법 적용”, 대한전자공학회논문지TC, Vol.45, No.7, pp.9-16, 2008.

[15] B. Bell, “Steiner Minimal Tree Problem”, http://www.css.taylor.edu/~bbell/steiner/, January, 1999.

김 준 모

e-mail : q888@dankook..ac.kr

1989년 서울대학교 컴퓨터공학과(학사)

2001년 University of Minnesota 전산학 (공학박사)

2002년~2004년 한국정보보호진흥원 연구원

2004년~현 재 단국대학교 컴퓨터학부 조교수

관심분야 : Approximations for NP-hard problems

김 인 범

e-mail : ibkim@kimpo.ac.kr

1989년 서울대학교 컴퓨터공학과(학사)

1991년 서울대학교 컴퓨터공학과(공학석사)

2007년 위스컨신주립대-밀워키 전산학(공학박사)

1996년~현 재 김포대학 IT학부 부교수

관심분야 : 네트워크 알고리즘, 데이터베이스, 컴퓨터 이론