DOI: 10.3745/KIPSTA.2009.16A.6.509

선분상의 포탈을 이용한 근사 선분 최소 신장 트리의 생성

김 인 범[†] · 김 수 인[†]

요 약

본 논문에서는 입력 선분들 상에 위치하며, 이들을 일정한 길이로 분할하는 가상 노드 포탈을 이용하여 입력 선분들을 모두 연결하는 근사 선분 최소 신장 트리를 빠른 시간 내에 찾는 방법을 제안한다. 이 근사 선분 최소 신장 트리는 통신선, 도로 및 철도망의 연결 등에 활용될 수 있다. 3000개의 입력 선분에 대해 제안된 방법으로 생성된 근사 트리는, 포탈 간격이 0.3인 경우에 최적 선분 최소 신장 트리와 비교하여 1.8%의 길이가 증가한 반면에 트리 생성 시간은 29.74%의 감소를 보였고, 0.75의 경우 2.96%의 길이의 증가와 39.96%의 트리 생성 시간의 절감을 보였다. 이는 약간의 길이 증가를 허용하면서 짧은 시간 내에 선분 연결 트리를 생성해야 하는 응용에 잘 적용될 수 있음을 보인다. 또한 제안된 방법은 포탈 간격, 포탈 포기 비율 등을 외부 인자로서 조절하여, 목적에 따른 트리 길이 또는 트리 생성 시간에 중점을 둔 근사 선분 최소 신장 트리 생성이 가능함을 보인다.

키워드 : 근사 선분 최소 신장 트리, 포탈, 포탈 간격, 포탈 포기 비율, 입력 선분, 연결 선분

Mechanism for Building Approximation Edge Minimum Spanning Tree Using Portals on Input Edges

Inbum Kim[†] · Soo-In Kim[†]

ABSTRACT

In this paper, a mechanism that produces an approximation edges minimum spanning tree swiftly using virtual nodes called portals dividing given edges into same distance sub-edges. The approximation edges minimum spanning tree can be used in many useful areas as connecting communication lines, road networks and railroad systems. For 3000 random input edges, when portal distance is 0.3, tree building time decreased 29.74% while the length of the produced tree increased 1.8% comparing with optimal edge minimum spanning tree in our experiment. When portal distance is 0.75, tree building time decreased 39.96% while the tree length increased 2.96%. The result shows this mechanism might be well applied to the applications that may allow a little length overhead, but should produce an edge connecting tree in short time. And the proposed mechanism can produce an approximation edge minimum spanning tree focusing on tree length or on building time to meet user requests by adjusting portal distance or portal discard ratio as parameter.

Keywords : Approximation Edge Minimum Spanning Tree, Portal, Portal Distance, Portal Discard Rate, Input Edge, Connecting Edge

1. 서 론

최소 신장 트리는 정점과 입력 선분이 주어져 있을 때, 주어진 일부 입력 선분을 이용해서 주어진 정점들을 최소 비용으로 모두 연결하는 트리로 정의된다. 일반적으로 비용은 트리의 길이를 의미한다. 이 최소 신장 트리는 네트워크 설계, 통신 라우팅, 회로 설계, 도로 건설 등에 활용될 수 있다. 이러한 트리를 구축하기 위한 다양한 알고리즘들이 많은 문헌에 발표되었다[1-3].

선분 신장 트리는 입력된 정점 없이, 선분만 주어졌을 때, 이 선분들을 직선으로 연결하는 트리이다. 선분 최소 신장 트리는 이러한 트리 중에서 가장 작은 비용의 트리로 정의한다. 선분이란 정적으로는 연속적으로 인접한 정점들의 집합으로 해석될 수 있거나, 통신 네트워크에서의 통신선으로 적용할 수 있다. 또한 선분은 공간 또는 평면상에 위치한 한 정점의 예측 가능한 동적 이동궤적이 될 수 있다. 이 선분을 연결하는 것은 이러한 통신선, 또는 경로가 이미 예정된 동적 장치들의 효율적인 연결에 잘 적용될 수 있을 것이다. 구체적인 예로, 이 선분 신장 트리는 동적 라우팅, ad-hoc 네트워크, 회로 설계, 항로 결정, 도로 연결 등 여러 분

※ 이 논문은 2009학년도 김포대학의 연구비 지원에 의하여 연구되었음.
† 정 회 원 : 김포대학 IT학부 부교수
논문접수 : 2009년 6월 22일
수 정 일 : 1차 2009년 8월 21일
심사완료 : 2009년 9월 2일

야에 응용될 수 있을 것이다. 이러한 응용들의 어떤 경우에는, 주어진 선분들을 연결함에 있어, 최적인 선분 최소 신장 트리에 비해 약간의 비용 혹은 길이의 증가를 허용하면서 빠른 시간 내에 트리의 생성이 더 중요할 수 있다.

본 논문에서는 이러한 응용을 위해서 빠른 시간 내에 근사 선분 최소 신장 트리를 생성하는 방법을 제안한다. 이 메커니즘에서는 주어진 선분을 일정한 간격으로 나누며 다른 선분들과의 연결 지점이 되는 가상 노드, 포탈을 생성한다. 이 포탈 간격에 의해 주어진 선분들에 대한 포탈의 개수가 결정된다. 또한 두 입력 선분 사이의 연결 선분은 한 입력 선분의 포탈들과 다른 입력 선분의 포탈들 사이의 연결 선에 의해 결정된다. 포탈 간격이 크면 최적 선분 최소 신장 트리와의 오차는 커지는 대신, 근사 선분 최소 신장 트리를 생성하는 시간은 줄어들며, 반대로 포탈 간격이 작으면 각 선분에 속한 포탈의 수는 많아지므로 선분 최적 최소 신장 트리와의 오차는 줄어들지만, 근사 선분 최소 신장 트리를 생성하는 시간은 많이 소요된다. 입력 선분의 길이가 포탈 간격보다 작거나, 어떤 입력 선분에 대해 포탈 간격에 따라 포탈을 모두 할당하고 남은 나머지 구간의 길이가 포탈 간격보다 작을 경우에 대해, 포탈을 어떻게 할당할 것인가에 대한 정책도 포탈의 개수를 결정하는 한 요인이 될 수 있다. 즉 주어진 포탈 간격의 어느 비율까지의 길이에 대해 포탈을 별도로 할당 하는 여부를 결정하는 비율에 따라 전체 입력 선분에 대한 포탈의 수의 증감이 결정된다. 본 논문에서는 이 비율을 포탈 포기 비율로 정의한다.

본 논문에서 제안하는 방법은 생성되는 근사 선분 최소 트리의 길이를, 최적 선분 최소 신장 트리에 비해 비록 증가시키지만, 포탈 간격 및 포탈 포기 비율을 적절히 설정하여 그 증가분의 크기를 늘리면서 트리의 생성 시간을 감소시키거나, 생성 시간을 늘려서 트리 길이의 증가분을 줄이는 선택이 가능하다.

본 논문의 구성은 다음과 같다. 2장은 본 논문에서 제안하는 메커니즘의 기본이 되는 최소 신장 트리에 대한 관련 연구 내용이고, 3장은 제안하는 메커니즘에 대한 기술이다. 4장에서는 제안된 메커니즘을 구현하고 입력 인자를 변경하면서 근사 선분 최소 신장 트리를 생성하는 실험 및 결과에 대한 분석이고 5장은 결론이다.

2. 관련 연구

신장 트리(Spanning Tree)란 이차원 평면에 주어진 N개의 정점들을 서로 교차하지 않는 입력선분으로 연결하는 트리이다. 주어진 정점과 입력 선분이 있을 때 일부 입력 선분을 이용하여 정점들을 모두 연결하는 최소 비용의 트리를 최소 신장 트리(Minimum Spanning Tree)라고 한다. N개의 정점을 연결하기 위해서는 N-1개의 선분이 필요하다. Kruskal과 Prim의 알고리즘이 최소 신장 트리를 구하는 대표적인 알고리즘이다[1-3].

유클리드 최소 신장 트리 문제는 선분이 입력으로 주어지지 않는 것이 최소 신장 트리 문제와의 가장 큰 차이점이다. 유클리드 최소 신장 트리는 주어진 N개의 정점들의 집합 P에 대하여, 이 점들을 연결한 삼각형들의 집합으로 표현될 수 있다. 이것을 삼각화(triangular)라고 하는데, 모든 삼각형의 외접원의 내부에 P의 점이 존재하지 않는 경우를 특별히 딜러니 삼각화(Delaunay Triangular)라 한다. 이 딜러니 삼각화는 P에 대한 평면 그래프로 표현되며, P에 대한 최소 신장 트리는 딜러니 삼각화 평면 그래프의 서브 그래프임이 증명되었다[4]. 그러므로 딜러니 삼각화된 그래프에서 선분은 정점의 수에 비례하므로, 이를 최소 신장 트리 알고리즘에 적용하여 O(E + NlogN) = O(kN + NlogN) = O(NlogN) 시간에 최소 신장 트리를 구할 수 있다.

본 논문에서 제안하는 입력 선분 연결 최소 신장 트리에 관한 연구는 입력 노드를 대상으로 한 전통적인 최소 신장 트리의 생성과 관련이 있다. 특수한 환경이나 제한에서의 최소 신장 트리를 효율적으로 생성하는 연구는 많이 시도되었지만, 일반적인 환경과 입력에서 트리의 길이가 최적화된 최소 신장 트리는 Kruskal 혹은 Prim의 알고리즘을 이용하여 얻을 수 있다.

최소 신장 트리의 생성 시간을 단축하기 위한 연구는 병렬처리, 분산처리, 그리고 랜덤기법(randomized)등을 이용하여 발표되었다. K.W. Chong 등은 Exclusive-Read Exclusive-Write Parallel Random Access Machine (EREW PRAM)에서 선형 개수의 프로세서를 이용하여 N개의 입력에 대하여 O(Log N) 시간 내에 최소 신장 트리를 생성할 수 있는 알고리즘을 발표하였다[5]. S. Pettie 등은 N개의 정점과 M개의 선분에 대해, pointer machine을 이용하여 $O(T^*(M,N))$ 시간 내에 최소 신장 트리를 생성하는 알고리즘을 구현하여 발표하였다[6]. 여기서 $T^*(M, N)$은 M개의 입력 정점과 N개의 입력 선분의 환경에서 최소 신장 트리를 생성하기 위해 필요한 선분 가중치의 비교 회수이다. D.R. Karger 등은 입력 선분의 가중치에 대한 이진 비교 연산만 허용되는 Unit-Cost Random-Access Machine에서 랜덤 샘플링 기법을 이용하여, 최소 신장 트리를 생성하는 랜덤 선형시간(Randomized Linear-Time) 알고리즘을 연구하였다[7]. M. Ahuja 등은 최소 신장 트리를 생성하기 위해 Echo 알고리즘을 활용한 분산 알고리즘을 구현하였는데, d를 입력 네트워크의 지름이라고 정의할 때, 이 알고리즘은 최대 (2M+2(N-1)log(N/2)) 의 메시지 교환과 (2dLogN)의 시간을 필요로 하고, 평균적으로 M개의 메시지와 O(d)의 시간을 필요로 한다[8].

그 밖의 최소 신장 트리와 관련된 연구로는 최소 신장 트리의 변형들이 있다. 최소 지름 신장 트리 (Minimum Diameter Spanning Tree)는 주어진 정점들에 대한 모든 신장 트리 중에서 지름이 가장 작은 트리이다. 여러 지점을 연결하는 트리 네트워크를 구축하고자 할 때, 일반적인 최소 신장 트리를 사용하면 네트워크를 최소 비용으로 구축할 수 있다. 그러나 구축비용보다는 가장 멀리 떨어진 두 지점간의 통신 비용이 더 관심 있는 요소라고 하면 네트워크의 지름을 최

소화하는 최소 지름 신장 트리를 구축하는 것이 오히려 바람직하다. J.M. Ho 등은 $O(N^3)$ 에 최소 지름 신장 트리를 계산하는 알고리즘을 발표하였다[9]. 이 연구에서, 주어진 정점들에 대한 최소 지름 신장 트리는 단항 트리 (monopolar tree)이거나 이항 트리(dipolar tree)이고, 단항 트리는 O(NlogN) 시간에, 이항 트리는 $O(N^3)$ 시간에 최소 지름 신장 트리를 생성하여, 결국 $O(N^3)$에 생성할 수 있음을 증명하였다. J. Gudmundsson등은연구 실행속도를 개선하여, 임의의 실수 $0<\varepsilon<1$에 대해 $(1+\varepsilon)$의 근사율을 가지는 근사 최소 지름 신장 트리를 선형 시간으로 계산할 수 있음을 발표하였다[10]. 통신망을 구성하는 각 노드의 작업양은 어느 정도 균일해야 한다. 한 노드에서 처리 작업 양은 노드에 연결된 통신 선분의 수와 밀접한 연관이 있다. 최소 지름 신장 트리의 경우는 단항 트리 또는 이항 트리이므로 한 정점에 연결된 선분의 수는 최대 N-1 이다. C. Monmna 등은 최소 신장 트리의 한 정점에 연결된 선분의 수는 최대 5를 넘지 않음을 증명하였다[11]. K.M. Chandy 등은 N개의 정점들로 구성된 네트워크에서 특정 정점 C에 나머지 N-1개의 정점들을 최소 비용으로 연결시키는 용량 최소 신장 트리문제를 연구하여 발표하였다[12]. 이 때 정점 C에 연결되는 모든 부-트리(sub-tree)들은 M개 이하의 정점들로 구성되어야 한다. 용량 최소 신장 트리는 정보 통신망의 Local Access Tree Network, 통신 시스템의 Local Loop System, 교통 시스템의 경로 설계 등에 활용 가능하다.

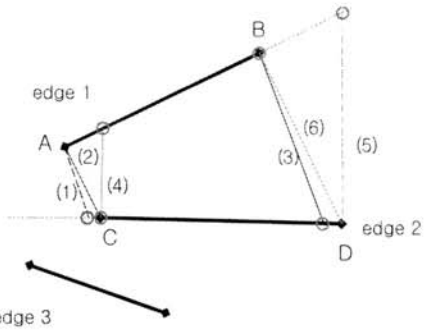
3. 본 론

(그림 1)은 Prim의 알고리즘을 활용한 최적 선분 최소 신장 트리의 생성 알고리즘이다. Prim의 알고리즘은 주어진 정점들과 이들 사이의 입력 선분을 이용하여 최소 신장 트리를 구하는 방법이다[3]. 선분 최소 신장 트리 문제는, 입력이 정점이 아닌 선분이고, 이 선분 사이의 연결 선분이 주어지지 않으므로 순수한 Prim의 알고리즘을 바로 적용할 수 없다. 따라서 모든 선분 쌍 사이의 연결 선분들을 별도의 방법으로 구하고, 그 거리를 가중치로 하여 Prim의 알고리즘을 다음과 같이 적용한다.

(그림 1)의 최적 선분 최소 신장 트리 생성 알고리즘을 실행하기 위해 각 입력 선분 사이의 최단 거리 연결 선분을 다음과 같이 구한다. (그림 2)는 두 개의 선분 AB와 CD의 연결 선분을 구하는 과정을 보인다. 먼저 입력 선분 AB의

단계 1: 입력된 선분들의 양 끝 점에서 다른 선분과의 최단 거리의 연결 선분을 모두 구함
단계 2: 입력 선분의 각 쌍의 선분들의 연결 선분들 중에서 최소 길이의 연결 선분을 선택
단계 3: 각 입력 선분을 정점으로, 단계 2에서 선택된 선분 사이의 연결 선분을 정점을 연결하는 연결 선분으로 변환하고, 각 연결 선분의 길이를 가중치로 정의하여 Prim의 알고리즘을 적용
단계 4: 단계 3에서 생성한 최소 신장 트리의 선택된 연결 선분을 이용하여 입력 선분을 연결
단계 5: 단계 4의 연결구조를 근사 선분 최소 신장 트리의 결과로 출력

(그림 1) 최적 선분 최소 신장 트리 생성 알고리즘

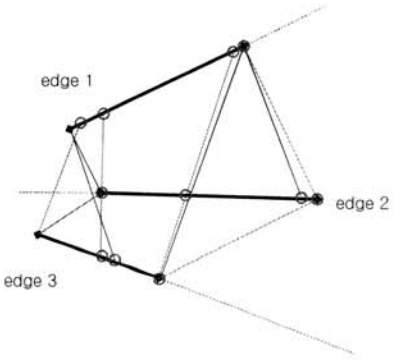


(그림 2) 두 입력 선분의 연결 선분을 구하는 과정

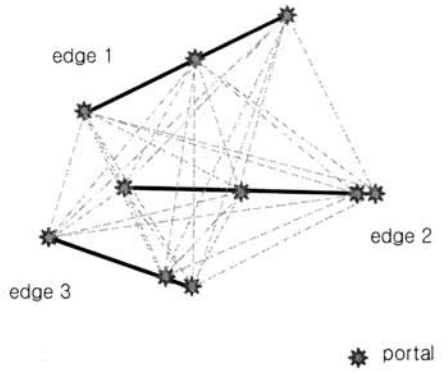
왼쪽 끝점 A에서 직각으로 입력 선분 CD에 선 (1)을 긋고, 그 교점을 구한다. 그 교점은 선분 CD 내부 또는 외부 일 것이다. (그림 2)에서 외부인 경우에는 교점과 가장 가까운 선분 CD의 끝점 C와 A를 연결하는 선 (2)를 긋는다. 마찬가지로 선분 AB의 또 다른 끝점 B에서 수직이면서 CD와 교차하는 선 (3)을 긋는다. 이번에는 선분 CD에서 한 끝점 C에서 수직이고 선분 AB와 만나는 선 (4)를 긋는다. 또 다른 끝점 D에서 수직이면서 AB와 만나는 선 (5)를 긋는다. 이 경우 교점이 선분 AB의 외부에 존재하므로 선분 AB와 가장 가까운 끝점인 B와 연결하는 선 (6)을 긋는다. 선분 AB와 CD의 최단 길이의 연결 선분은 선분 외부와 연결되는 선 (1)과 (5)를 제외한, 선 (2), (3), (4), (6) 중에서 최단 길이의 선이다.

(그림 3)은 세 입력 선분의 연결 선분을 구하는 방법을 나타나낸다. 이 세 선분의 거리를 구하기 위해서는 총 12개 연결 선분의 길이를 구하여 비교해야 할 것이다. 즉 선분 1의 양 끝점에서 선분 2, 선분 3의 연결 선 2개씩 총 4개, 선분 2의 양 끝점에서 선분 1, 선분 3의 연결 선 2개씩 총 4개, 선분 3의 양 끝점에서 선분 1, 선분 2의 연결 선 2개씩 총 4개의 길이가 필요하다. 선분 1, 2의 연결 선 4개를 비교하여 최소 길이의 연결 선이 선분 1, 2의 최단 길이 연결 선분이 되고, 선분 2, 3의 연결 선 4개를 비교하여 최소 길이의 연결 선이 선분 2, 3의 연결 선분이 되고, 선분 1, 3의 연결 선 4개를 비교하여 최소 길이의 연결 선이 선분 1, 3의 연결 선분이 된다.

(그림 4)는 본 논문에서 제안하는 포탈을 이용하여 세 입



(그림 3) 세 입력 선분의 연결 선분

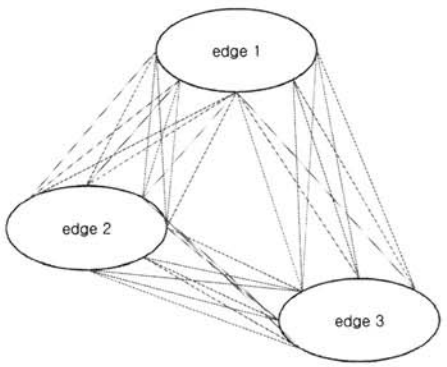


(그림 4) 포탈을 이용한 세 입력 선분의 연결 선분

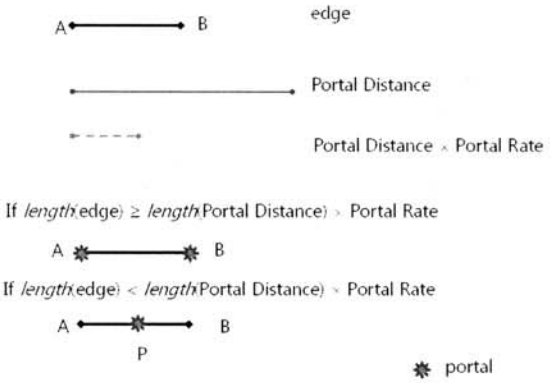
력 선분의 연결 선분을 구하는 방법을 나타낸다. 각 포탈의 개수는 입력 인자(parameter)로 주어진 포탈 사이의 거리, 즉 포탈 간격으로 결정된다. 각 선분의 왼쪽 끝점의 위치에 한 개의 포탈이 위치하게 되고 이 끝점에서부터 포탈 간격의 양의 정수 배 거리만큼 떨어진 위치에 포탈이 하나씩 위치하게 된다. 이렇게 배정 했을 때 해당 선분의 마지막 구간의 길이가 포탈 간격보다 작을 경우의 포탈은 포탈포기 비율에따라 결정된다. 선분 1에는 포탈이 3개, 선분 2에는 포탈이 4개, 선분 3에는 포탈이 3개 존재한다. 어떤 선분 X와 선분 Y의 거리는 선분 X의 포탈과 선분 Y에 위치하는 각 포탈 사이의 모든 거리들 중에서 최소 거리로 결정된다. 포탈 간격이 작으면 포탈의 개수가 많아지므로 이 거리를 구하는 과정이 많아질 것이지만, 선분 사이의 최단 거리와의 오차는 줄어들 것이다.

(그림 5)는 (그림 4)의 내용을 Prim의 알고리즘에 적용하기 위해 변형시킨 것이다. 즉 입력 선분들은 최소 신장 트리 문제의 입력 정점이 되고, 선분들 사이의 거리를 구하기 위해 구한 포탈들간의 연결 선들은 최소 신장 트리 문제의 선분이 되도록 변형시킨 것이다. (그림 5)와 같이 (그림 4)의 세 개의 입력 선분은 3개의 입력 정점이 되고 각 선분에 위치한 포탈들의 연결 선들은 각 포탈이 속한 선분을 의미하는 정점들을 연결하는 연결 선분들로 표현된다.

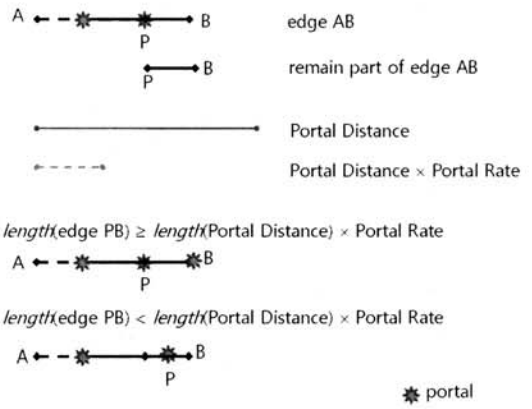
(그림 6)은 입력 선분의 길이가 포탈 간격보다 작은 경우



(그림 5) Prim의 알고리즘 이용을 위한 변형



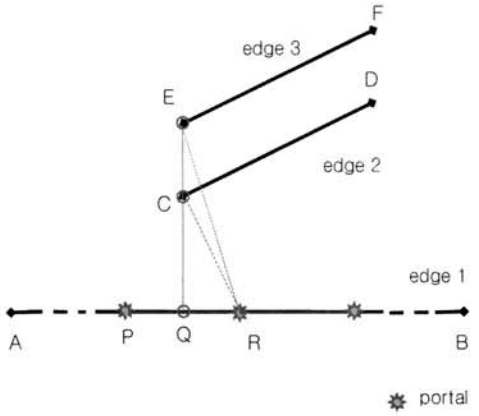
(그림 6) 포탈간격보다 짧은 선분의 포탈생성



(그림 7) 포탈간격보다 짧은 최종 구간의 포탈생성

의 처리 방법을 나타낸다. 포탈 간격과 포탈 포기 비율의 곱보다 입력 선분의 길이가 크거나 같으면 해당 선분의 양 끝점에 포탈을 각각 생성하고, 그렇지 않으면 선분의 중앙에 하나의 포탈을 생성한다. (그림 7)은 입력 선분의 길이가 포탈 간격보다 큰 경우, 앞에서 기술한 것처럼 순서대로 선분의 왼쪽부터 차례로 포탈을 생성하여 위치시키지만, 남은 마지막 구간의 길이가 포탈 간격보다 작은 경우에, 남은 구간의 길이가 포탈 간격과 포탈 포기 비율의 곱보다 크거나 같은 경우이면 선분의 마지막 오른쪽 끝점에 포탈을 생성하여 할당한다. 그렇지 않은 경우에는 남은 구간의 중앙 위치에 포탈을 생성한다. 이 때 이 구간의 왼쪽에 위치한 이미 생성된 포탈은 버려지게 된다.

입력 선분 사이의 거리와 포탈 간격과의 관계를 분석하면 다음과 같다. (그림 8)은 포탈 간격의 길이가 입력 선분들간의 거리들 중에서 최소 값과 동일한 경우를 가정한다. edge 1인 선분 AB와 edge 2인 선분 CD간의 거리 CQ가 최단 거리 연결 선분이라 가정하고, 포탈 간격인 구간 PR의 길이 *dis*(PR)와 CQ의 길이 *dis*(CQ)가 같다고 가정한다. 이 경우, 포탈을 사용함으로 발생되는, 선분 AB와 선분 CD의 최단 거리에 대한 오차는 $min\{dis(CR), dis(CP)\} - dis(CQ)$이다. 여기서 *min*(X,Y)의 결과는 X, Y의 최소 값이다. Q가 PR의 중점에 위치하도록 두 선분이 배치될 때, 최단 거리에 대한



(그림 8) 선분 간격에 따른 오차 계산

오차는 최대가 될 것이다. *dis*(PR)=*dis*(CQ)이고, *dis*(PQ)=*dis*(RQ) 일 때, 오차는 $(\sqrt{5}-2)\times dis(CQ)/2$ 이고 오차율은 11.8%이다. edge 1과 이 최소 거리보다 더 큰 거리만큼 떨어진 선분, 예를 들어 edge 3인 선분 EF의 경우 거리 오차율은 이 보다 작아질 것이다. 최단 거리의 선분을 대상으로 한 것이므로 입력 선분들 모두를 대상으로 한 전체 오차율은 이보다 작아질 것이다. 어떤 선분과의 거리가 모든 입력 선분들 사이의 거리 중 최소인 거리의 k배라고 가정했을 때, 오차율은 다음과 같다.

$$\sqrt{1+(\frac{1}{2\times k})^2}\ -1$$

예를 들어 최소 거리의 3배 되는 선분은 k=3이 되므로 이때의 오차율은 약 1.4%가 될 것이다. 따라서 k=1인 즉, 포탈간격이 최소 선분간의 거리와 같다면 오차율은 최대 11.8%가 될 것이지만, 선분간의 거리가 최소 거리 이상일 것이므로 오차율은 11.8% 미만이 될 것이다. 포탈 간격을 선분간 최소 거리 *dis*(CQ) 보다 작게 하면 *min*{*dis*(CR), *dis*(CP)}값이 작아질 것이므로 오차는 더 줄게 될 것이고 반대로 포탈 간격을 크게 하면, 오차는 더 커질 것이다. 따라서 주어진 선분들의 최소 거리를 기준으로 포탈간격을 정하면 최대 오차율에 대한 추정이 가능하다.

최적의 선분 최소 신장 트리를 구하기 위해서는 각 선분당 양 끝점에서의 다른 선분과의 연결이 필요하다. 입력 선분의 개수를 e라고 했을 때, 한 선분의 다른 선분과의 연결 개수는 $2\times(e-1)$이고, 따라서 총 연결의 수는 $2\times(e-1)\times e$ 이다. 이것의 실행시간은 트리 구조의 연결 수에 비례하므로 $O(e^2)$ 이다.

한 입력 선분 i에 위치하는 포탈의 개수는 $(\frac{l_i}{d}+\varphi)$이다. 여기서 l_i는 선분 i의 길이이고 d는 외부 인자로 주어지는 포탈 간격이다. φ는 포탈 포기 비율에 따라 포탈 배정 대상에 대한 포탈 할당을 결정 하는 함수이다. 따라서 모든 입력 선분에 대해 연결의 개수의 총 합을 계산하면 다음과 같다.

$$\sum_{i=1}^{e}\sum_{j=1,j\neq i}^{e}(\frac{l_i}{d}+\varphi)\times(\frac{l_j}{d}+\varphi)$$

본 논문에서 제안하는 메커니즘에서 가장 많은 연결 개수의 경우는 각 선분의 길이가 동일하여 각 선분의 포탈의 수가 같을 때이다. 즉 최악의 실행시간은 모든 입력 선분의 길이가 같을 때이다. 따라서 다음과 같은 식이 성립한다.

$$\sum_{i=1}^{e}\sum_{j=1,j\neq i}^{e}\left(\frac{l_i}{d}+\varphi(r)\right)\times\left(\frac{l_j}{d}+\varphi(r)\right)\le\sum_{i=1}^{e}\sum_{j=1,j\neq i}^{e}\left(\frac{\gamma}{d}+\varphi(r)\right)^2$$

$$=\frac{1}{2}\times\left(\frac{\gamma}{d}+\varphi(r)\right)^2 e^2=O(e^2)$$

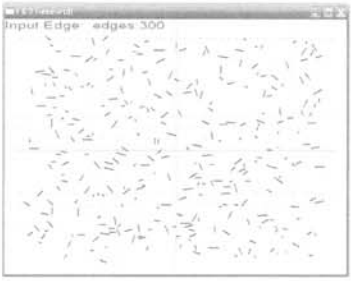
$$\varphi(r)=\begin{cases}0, & r<d\times p\\ 1, & r\ge d\times p\end{cases}$$

위의 식에서 γ은 전체 입력 선분에 대하여 평균 길이이다. r은 포탈 배정 여부를 결정해야 하는 대상 구간의 길이이다. p는 외부 인수로 입력되는 포탈 포기 비율로, 0과 1사이의 실수 값이다. 본 논문에서 제안하는 메커니즘도 최적치인 최적 선분 최소 신장 트리와 동일한 $O(e^2)$의 시간 복잡도를 갖는다. 따라서 외부 인수인 포탈 간격 d와 포탈 포기 비율 p를 적절히 선택하여 생성 메커니즘을 구성한다면, 최대 오차율을 고려하면서, 추정 가능한 빠른 시간 안에 근사 선분 최소 신장 트리를 생성할 수 있다.

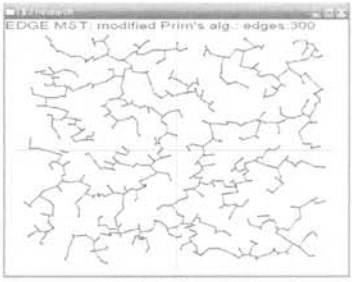
(그림 9)는 본 논문에서 제안하는 근사 선분 최소 신장 트리를 생성하는 알고리즘이다. 이 알고리즘을 이용하여 300개의 입력 선분에 대하여 각 포탈 간격을 달리하여 생성한근사 선분 최소 신장 트리의 결과가 (그림 10)에 있다. (그림 10)-(a)는 300개의 무작위로 생성된 입력 선분을 보이고 있고 (그림 10)-(b)는 이들 선분들에 대하여 Prim의 최소 신장 트리 알고리즘을 변형하여 생성한 최적 선분 최소 신장 트리의 모습이다. (그림 10)-(c),(d)는 포탈 간격이 각각 0.05, 0.85일 경우의 최종적으로 생성되는 근사 선분 최소 신장 트리들이다. 최적 선분 최소 신장 트리의 길이가 308.45인데, 포탈 간격이 0.05인 경우가 이 길이와 가장 근사한 308.48이고, 포탈 간격이 0.85인 경우에는 311.93이다.

단계 1: 입력된 선분들의 길이를 각각 계산
단계 2: 길이가 포탈 간격보다 작은 입력 선분들에 대해 포탈간격과 포탈 포기비율의 곱과 길이를 비교하여 그림 6과 같이 포탈 생성 및 배치
단계 3: 길이가 포탈 간격보다 크거나 같은 입력 선분들에 대해 그림 7과 같이 각 구간에 대해 차례로 포탈의 생성 및 배치
단계 4: 각 선분들의 포탈과 다른 선분들의 포탈을 연결하는 선 생성
단계 5: 각 입력 선분을 정점으로, 단계 4에서 생성한 연결 선을 선분으로 변형하여 최소 신장 트리 생성.
단계 6: 단계 6에서 생성한 최소 신장 트리 결과의 선분을 이용하여 입력선분을 연결
단계 7: 단계 6에서 구축된 연결구조를 근사 선분 최소 신장 트리의 결과로 출력

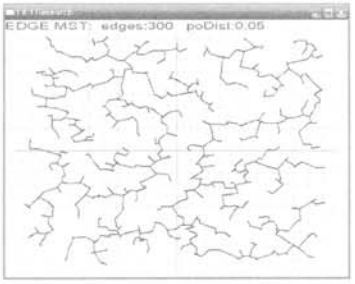
(그림 9) 근사 선분 최소 신장 트리 생성 알고리즘



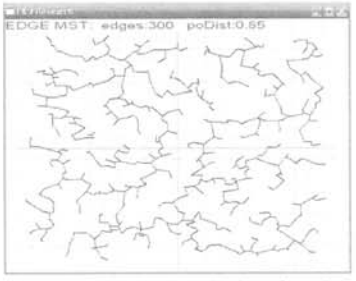
(a) 입력 선분 (300개)



(b) 최적 선분 최소 신장 트리
(길이=308.45)



(c) 포탈_간격=0.05의 근사 선분 최소
신장 트리 (트리 길이=308.48)

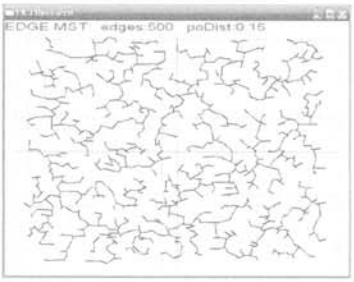


(d) 포탈_간격=0.85의 근사 선분 최소 신장
트리 (트리 길이= 311.93)

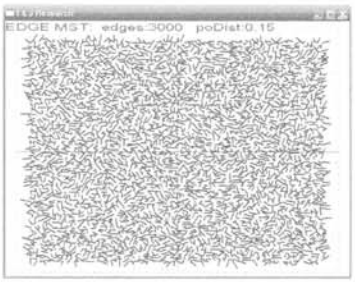
(그림 10) 포탈 간격의 변경에 따라 생성된 선분 최소 신장 트리

4. 실험 및 결과 분석

본 연구를 위해 사용된 실험 인자는 입력 선분의 수, 포탈 간격, 포탈 포기 비율 등이다. 관찰결과는 생성된 근사 선분 최소 신장 트리 길이의 근사도와 각 트리의 생성 시간이다. 근사도는 3장에서 기술한 변형된 Prim의 최소 신장 트리 알고리즘을 변형, 실행시켜 생성된 최적 선분 최소 신장 트리의 길이와 비교하여 얻는다. 실험을 위해 무작위로 생성된 선분의 수는 500, 1000, 1500, 2000, 2500, 3000개이



(a) 근사 선분 최소 신장 트리
(입력선분=500)



(b) 근사 선분 최소 신장 트리 (입력선분=3000)

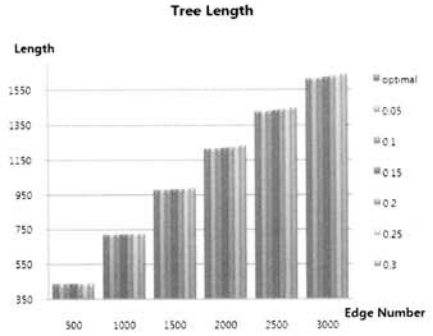
(그림 11) 입력선분 수의 따라 생성되는 근사 선분 최소 신장 트리(포탈 간격=0.15)

다. 각 선분들은 2차원 평면상에서, -10.0과 10.0 사이의 x, y 좌표 값을 무작위로 선택하고 이를 중심으로 하여 0과 360도 사이의 각도를 가지는, 0.2와 0.8 사이의 거리의 지점을 연결하여 생성된다. 생성된 각 선분들은 서로 교차 않고 또한 이들 각 선분들 사이는 거리가 0.15이상이 되도록 제한을 가하였다. 각 선분 위에 위치하는 가상의 점인 포탈 사이의 간격은 0.05과 1.05 사이의 값으로 지정하여 실험하였다. 실험 환경은 Intel 1.83 GHz (T5600) 프로세서와 1기가 메모리 램을 장착한 랩탑 컴퓨터이고, 본 논문에서 제안하는 알고리즘을 C++로 구현하였다.

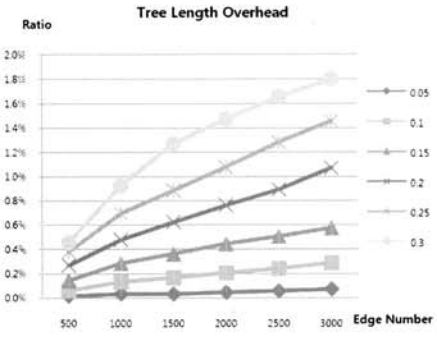
(그림 11)은 포탈 간격이 0.15이고, 입력 선분의 개수가 500, 3000개일 경우의 생성되는 근사 선분 최소 신장 트리의 결과를 보이고 있다.

4.1 근사 최소 신장 트리 길이 및 오차율

(그림 12)-(a),(b)에는 포탈 포기 비율이 0.0일 때, 입력 선분의 수와 포탈 간격의 변화에 따라 생성되는 각 근사 선분 최소 신장 트리의 길이와 최적 선분 최소 신장 트리와의 길이의 오차율을 비교한 결과가 나타나 있다. (그림 12)-(b)에서와 같이 각 포탈 간격을 크게 할수록 각 근사 선분 최소 신장 트리의 길이 오차율은 점차 커짐을 알 수 있다. 그러나 포탈의 간격이 어느 크기 이상이 되면 오차율은 더 이상 늘어나지 않을 것이다. 그 이유는 본 논문에서 제안된 방법에 의해 포탈 간격이 모든 입력 선분의 길이 이상이 되면 모두 선분의 양 끝점에만 포탈이 생성될 것이기 때문이다. 입력 선분의 수가 3000개인 경우에, 포탈 간격이 0.05인 경우에는 0.07%, 0.1인 경우에는 0.29%, 0.15인 경우에는 0.58%, 0.2인 경우에는 1.07%, 0.25인 경우에는 1.46%, 0.3인

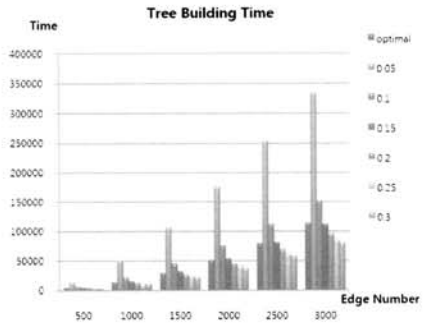


(a) Building Tree Length

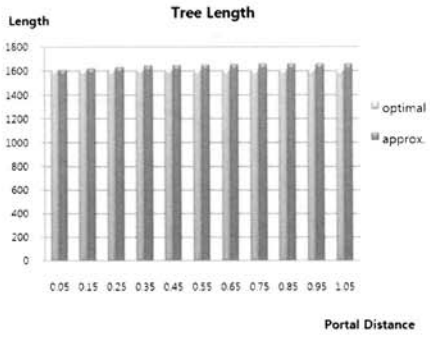


(b) Additional Length Ratio of Building Tree

(그림 12) 입력 선분 수, 포탈 간격의 변화에 따른 근사 선분 최소 신장 트리 길이



(a) Tree Building Time



(b) Time Overhead Ratio of Building Tree

(그림 13) 입력 선분 수, 포탈 간격의 변화에 따른 근사 선분 최소 신장 트리 생성 시간

경우에는 1.8%의 오차율을 보였다. 또한 입력 선분이 많을수록 근사 선분 최소 신장 트리의 오차율이 점차 증가하지만, 포탈 간격이 작으면 오차율의 차이는 거의 없어짐을 확인할 수 있다.

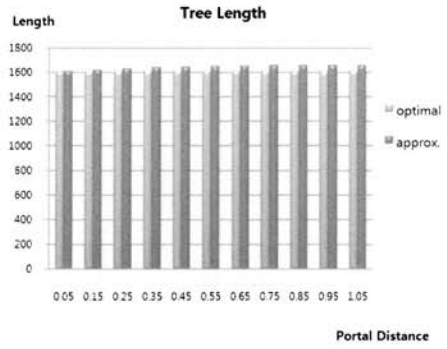
4.2 근사 최소 신장 트리 생성 시간 및 시간 절감율

(그림 13)에는 포탈 포기 비율이 0.0일 때, 입력 선분의 수와 포탈 간격의 변화에 따라 생성되는 각 근사 선분 최소 신장 트리의 생성 시간을 비교한 결과가 나타나 있다. 입력 선분의 수가 많은 트리일수록 절대적인 생성 시간은 더 많이 요구된다. 입력 선분의 수가 같은 경우에, 각 포탈 간의 간격을 크게 하면 할수록 생성시간은 줄어든다. 포탈의 간격을 0.15로 했을 때, 생성 시간은 최적 선분 최소 신장 트리를 생성하는 것과 유사해 짐을 확인할 수 있다. 이 포탈 간격의 길이는 입력 선분간의 거리들 중에서 최소인 값과 일치하는 수치이다. 입력 선분의 수가 3000개인 경우에 생성되는 트리는, 최적 선분 최소 신장 트리 생성 시간에 비해, 포탈 간격이 0.05인 경우에는 195%, 0.1인 경우에는 33%, 0.15인 경우에는 -2%, 0.2인 경우에는 -16%, 0.25인 경우에는 -25%, 0.3인 경우에는 -30%의 차이를 보였다. 이 결과에서 포탈의 간격을 크게 하면 최적 선분 최소 신장 트리에 비교해 생성 시간이 많이 절감되는 것으로 분석되지만 어느 포탈의 간격이 어느 크기 이상이 되면 절감율은 더 이상 커지지 않는다. 그 이유는 포탈 간격이 모든 선분의 길이 이상이라면 모든 입력 선분들이 생성하는 포탈의 수는 같아질 것이고, 이는 같은 개수의 정점들이 입력되는 최소 신장 트리의 생성이 되기 때문이다.

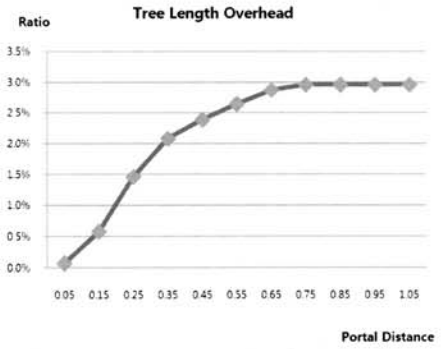
4.3 포탈 간격 변경

포탈 간격에 대한 세밀한 분석을 위해, 포탈 포기비율이 0.0일 때, 3000개의 입력 선분에 대해 포탈 간격을 달리하여 실험, 분석하였다. (그림 14)에는 3000개의 입력 선분에 대해, 포탈간의 간격을 변경하여 생성된 근사 선분 최소 신장 트리의 길이에 대한 결과가 나타나 있다. 포탈 간격이 짧으면 각 선분을 대표하는 포탈의 수가 많아 질 것이므로, 생성되는 근사 트리의 길이와 최적 트리의 길이와의 오차가 줄어들 것이다. 포탈 간격이 커지면 그 오차는 증가할 것이다. 포탈 간격이 0.05인 경우에는 최적 선분 신장 트리에 비해 0.07% 트리의 길이가 증가한다. 포탈 간격이 커짐에 따라 생성되는 근사 트리 길이의 증가는 계속되어, 포탈 간격이 0.65 인 경우 2.87% 증가하였다. 트리 길이는 포탈 간격이 0.75부터는 최적 선분 최소 신장 트리에 비해 2.96% 증가한 상태로 수렴된다. 이 포탈 간격은 입력된 선분의 길이들 중 최대 길이인 0.8에 근접한 값이다.

(그림 15)는 3000개의 입력 선분에 대해, 포탈 간격을 변경하여 생성된 근사 선분 최소 신장 트리의 생성 시간에 대한 결과이다. 포탈의 길이를 작게 하면 각 선분을 대표하는 포탈의 수가 많아질 것이고 이를 이용한 최소 신장 트리의 생성 시간은 커질 것이다. 포탈 간격이 0.05인 경우의 근사

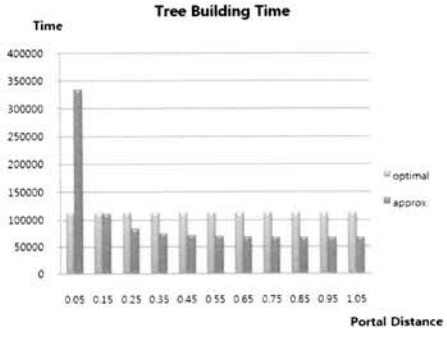


(a) Building Tree Length

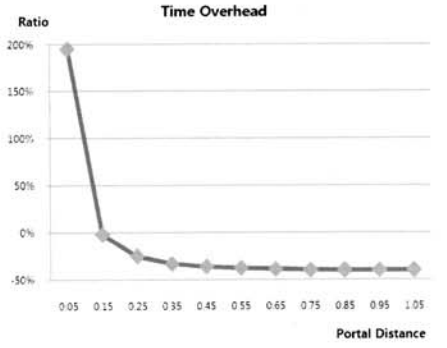


(b) Additional Length Ratio of Building Tree

(그림 14) 입력 선분 수가 3000일 때, 포탈 간격의 변화에 따른 근사 선분 최소 신장 트리 길이



(a) Tree Building Time



(b) Time Overhead Ratio of Building Tree

(그림 15) 입력 선분 수가 3000일 때, 포탈 간격의 변화에 따른 근사 선분 최소 신장 트리 생성 시간

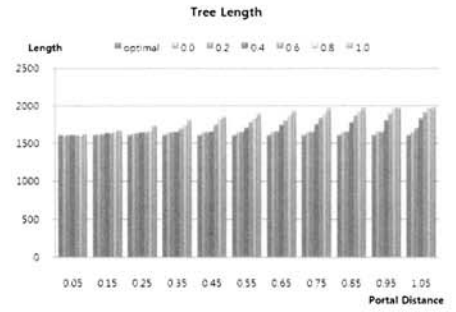
트리인 경우에는 최적 선분 최소 신장 트리의 생성 시간에 비해 약 194.8% 초과되었지만 0.15부터는 급격히 감소하여 약 2.2% 감소하였고 포탈의 간격이 증가함에 따라 근사 트리의 생성 시간은 더욱 감소하였다. 그러나 포탈 간격이 일정한 값에 도달하면 생성되는 포탈의 수는 일정해 질 것이다. 그 결과 해당 근사 트리의 생성하는 시간은 거의 일정해 진다. (그림 15)의 (b)에서와 같이, 포탈 간격이 0.65인 경우에 최적 선분 최소 신장 트리의 생성 시간에 비해 약 38.9% 감소되고, 0.75 이상부터는 약 40%에 수렴하는 값으로 절감된다.

4.4 포탈 포기비율의 변화

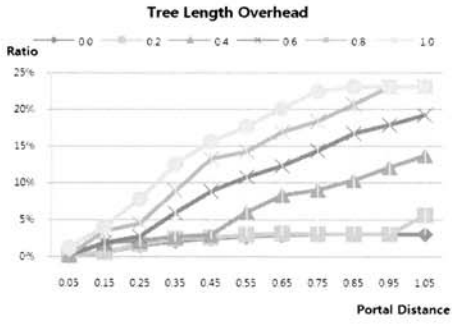
이전 실험까지는 포탈 포기 비율이 0.0인, 즉 입력 선분의 길이 또는 포탈 배정 후 선분의 남은 구간의 길이가 포탈 간격보다 작은 경우에 무조건 오른 쪽 끝점에 하나의 포탈을 자동적으로 배정하였다. 따라서 모든 입력 선분에 대해 선분 길이에 상관없이 최소한 두 개의 포탈이 자동적으로 생성되었다. 그러나 짧은 길이의 선분에 대해서는 하나의 포탈로도 충분할 수 있다. 포탈의 개수가 많아지면 근사 트리의 길이 근사도 측면에서는 긍정적이지만, 근사 트리의 생성 시간에는 부정적 영향을 미친다.

포탈 생성 대상의 선분 또는 구간 길이가 포탈 간격보다 작을 경우, 3장의 (그림 6)과 (그림 7)과 같이, 그 길이를 포탈 간격과 포탈 포기비율의 곱과 비교하여 결과에 따라 포탈을 양 끝점에 각각 생성하거나 적절한 위치에 하나 생성하게 한다. 즉, 포탈 간격과 주어진 포탈 포기비율의 곱 보다 포탈 생성 대상의 길이가 큰 경우, 오른쪽 끝점에 하나의 포탈이 생성될 것이다. 만약에 그 곱의 결과보다 작다면, 그 대상을 대표할 할 수 있는 적절한 위치에 하나의 포탈을 생성한다. 예를 들어 포탈 포기 비율이 0.0이면 포탈 생성 대상의 길이에 상관없이 오른쪽 끝점에 하나의 포탈을 생성하고 1.0이면 그 길이가 포탈 간격보다 크다면 오른쪽 끝점에 하나의 포탈이 생성될 것이고, 그 길이가 포탈 간격보다 작다면 중간 위치에 포탈이 하나 생성 될 것이다. 이 때 바로 왼쪽 위치에 생성된 포탈은 버려지게 된다. (그림 16)에는 3000개의 입력 선분에 대해, 포탈 포기비율을 변경하여 생성된 근사 선분 최소 신장 트리의 길이에 대한 결과가 나타나 있다. 포탈 포기 비율이 낮을수록 최적 치에 가까워짐을 알 수 있다. 또한 포탈 간격이 큰 경우에는, 포탈 포기 비율의 증가가 어느 값까지는 오차율이 증가함을 확인할 수 있다. 포탈 간격이 0.05인 경우, 포탈 포기 비율이 0.0인 경우에는 0.07%의 오차를 보임에 반해 비율이 1.0인 경우에는 0.8%의 오차율을 보였다. 포탈간격이 1.05인 경우에, 포탈 포기 비율이 0.0인 경우에는 2.96%의 오차를 보임에 반해 비율이 1.0인 경우에는 23.08%의 오차율을 보였다. 그러나 비율이 1.0인 경우에는 포탈간격이 0.85부터는 오차율이 변화가 없는데, 이는 3000개 모든 선분에 대해 하나의 포탈이 생성되어 동일한 포탈분포를 나타내기 때문이다.

(그림 17)은 3000개의 입력 선분에 대해, 포탈 포기 비율

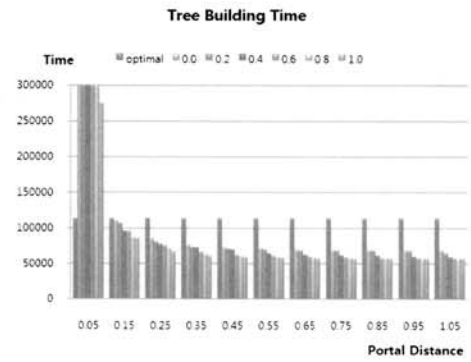


(a) Building Tree Length

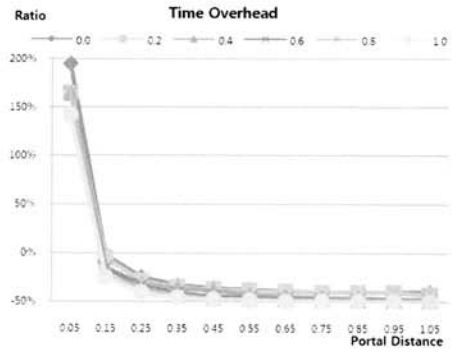


(b) Additional Length Ratio of Building Tree

(그림 16) 포탈 포기 비율의 변화에 따른 근사 선분 최소 신장 트리 길이



(a) Tree Building Time



(b) Time Overhead Ratio of Building Tree

(그림 17) 포탈 포기 비율의 변화에 따른 근사 선분 최소 신장 트리 생성 시간

을 변경하여 생성된 근사 선분 최소 신장 트리의 생성 시간에 대한 결과이다. 포탈 포기 비율을 작게 하면 각 선분을 대표하는 포탈의 수가 많아지므로 최소 신장 트리를 생성하는 시간은 많이 필요할 것이고, 포탈 포기비율을 크게 하면 각 선분을 대표하는 포탈의 수가 적어지므로 최소 신장 트리를 생성하는 시간은 많이 단축될 것이다. 포탈 간격이 0.05인 경우에는 비율이 0.0인 경우, 최적 선분 최소 신장 트리에 비해 약 194.8% 초과된 시간이 소요되었고, 비율이 1.0인 경우에는 약 142.0% 초과되었다. 포탈간격이 0.15부터는 급격히 감소하여 각각 약 2.2%, 24.5% 감소하였고 포탈 간격이 증가함에 따라 트리 생성 시간은 더욱 감소하였다. 그러나 포탈 간격이 일정한 길이에 도달하면 소요되는 시간은 일정해 진다. 포탈 간격이 0.75이상이면 포탈 포기 비율이 0.0인 경우에는 트리의 생성시간이 약 40.1% 감소하고, 1.0인 경우에는 약 49.8% 감소하는 것으로 수렴된다.

5. 결 론

본 논문은 주어진 입력 선분들을 직선 연결 선분을 이용해서 최소 비용으로 연결하는, 선분 신장 트리를 신속히 구성하는 방안을 제안한 것이다. 이 방법에 의해 생성되는 근사 선분 최소 신장 트리의 길이는, 비교 대상인 최적 선분 최소 신장 트리에 비해 증가되었지만, 외부에서 포탈 간격 및 포탈 포기 비율을 적절히 설정함으로 인해, 증가분의 크기와 트리의 생성 시간을 조절할 수 있다.

입력 선분은 통신 네트워크에서의 통신선 또는 동적객체의 정해진 이동궤적으로 간주될 수 있다. 따라서 선분 신장 트리 문제는 네트워크에서의 라우팅, ad-hoc 네트워크, 회로 설계, 항로 결정, 도로 연결 등에 적용될 수 있다. 그런데 이러한 응용들의 어떤 경우에는 주어진 선분들을 최소 비용 또는 최소 길이로 연결하는 것보다, 허용 가능한 비용의 추가 또는 길이의 증가 이내에서 빠르고 신속하게 연결 방법을 찾는 것이 더 중요한 요인이 될 수 있다.

이 문제에 대한 최적의 최소 비용의 트리를 구하는 방법은 각 선분의 양 끝점에서 다른 선분들과의 최단 거리의 연결 선들을 모두 구하고, 이 중 최소 길이의 연결선을 선택하여 최소 신장 트리를 구하는 것이다. 그러나 이 방법은 복잡한 수학적 과정을 거치기 때문에 입력 선분의 수가 매우 많다면, 상대적으로 많은 시간을 필요로 한다. 따라서 본 논문에서는 비록 최적의 트리와 비교하여 어느 정도의 길이의 증가는 허용하면서 빠른 시간 내에 근사 선분 최소 신장 트리를 형성하는 방법을 제안하였다. 이를 위해 각 입력 선분 위에 포탈이라는 가상 노드를 도입하여 이용하였다. 본론에서 기술한 것과 같이, 포탈 간격이 입력 선분들 사이의 최소 거리인 경우, 수학적 계산으로는 생성되는 트리의 증가분이 최대 11.8% 이다. 실행 시간은 최적 선분 최소 신장 트리의 생성 방법과 본 논문에서 제안하는 메커니즘 모두 $O(e^2)$로 동일하지만, 본 논문에서 제안된 방법의 경우는, 외부 인수인 포탈 간격과 포탈 포기 비율에 따라 계산식의 상

수항 조정이 가능하므로 적용하려는 응용의 목적과 요구에 따라 실행시간과 트리 길이 증가분의 조절이 가능하다. 본 논문의 실험 결과는 포탈 간격을 크게 할수록, 또한 포탈 포기 비율을 크게 할수록 실행시간이 단축되었다. 그러나 이 경우 생성되는 근사 선분 최소 신장 트리들은 상대적으로 높은 트리 길이의 증가분을 보였다.

향 후 연구는, 생성되는 근사 선분 연결 트리의 길이를 단축시키기 위한 방안으로, 최소 신장 트리보다 더 최단의 트리를 구성할 수 있는 최소 스타이너(Steiner) 트리 문제를 고려하여 적용하고자 한다. 그러나 최소 스타이너 트리 문제는 NP 문제이므로 적절한 휴리스틱을 고안해야 한다[13]. 또한 무수히 많은 입력 선분을 연결해야 하는 경우, 이를 위한 근사 선분 최소 신장 트리의 생성 시간의 단축을 위해, 다이나믹 프로그래밍 기법을 이용한 PTAS(Polynomial Time Approximation Scheme)의 적용 방안을 연구하려 한다[14]. 이 PTAS는 NP 문제와 같은 실 세계에서 계산하기 힘든 문제에 대하여, 그 문제를 작은 기본 단위로 세분화시키고, 이를 Brute-force 방법으로 해결한 후, 그 결과를 상위 단계에서 통합하는 방식으로 원래 문제를 해결하는 방법이다. 이러한 시도를 통해 네트워크에서의 통신선 연결 등에 유용하게 사용될 수 있는 선분 연결 문제에 대하여 트리 길이 혹은 생성시간 면에서, 좀 더 개선된 방법을 제안할 수 있을 것이다.

참 고 문 헌

[1] 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.

[2] http://en.wikipedia.org/wiki/Minimum_spanning_tree, August, 2009.

[3] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, Introduction to Algorithm, 2nd Ed., MIT Press, 2001.

[4] F.P. Preparata and M.I. Shamos, Computational Geometry: an Introduction, Springer-Verlag, New York, 1985.

[5] K.W. Chong, Y. Han, and T.W. Lam, "Concurrent threads and optimal parallel minimum spanning trees algorithm", Journal of the ACM, Vol.48, pp.297-323, March, 2001.

[6] S. Pettie and V. Ramachandran, "An Optimal Minimum Spanning Tree Algorithm", Journal of the ACM, Vol.49, pp.16-34, January, 2002.

[7] D.R. Karger, P.N. Klein and R.E. Tarjan, "A randomized linear-time algorithm to find minimum spanning trees", Journal of the ACM, Vol.42, pp.321-328, March, 1995.

[8] M. Ahuja and Y. Zhu, "A distributed algorithm for minimum weight spanning trees based on echo algorithms", 9th International Conference on Distributed Computing Systems, pp.2-8, June, 1989.

[9] J.M. Ho, D.T. Lee, C.H. Chang, and C.K. Wong, "Minimum Diameter Spanning Trees and Related Problems", SIAM Journal on Computing Vol.20, No.5, pp.987-997, 1991.

[10] J. Gudmundsson, H. Haverkort, S.M. Park, C.S. Shin and A. Wolff, "Approximating the Geometric Minimum-Diameter Spanning Tree", APPROX 2002, Springer LNCS 2462, pp.146-160, 2002.

[11] C. Monmna and S. Suri, "Transitions in Geometric Spanning Trees", Transitions in Geometric Spanning Trees Vol.8, No.1, pp.265-293, Dec 1992.

[12] K.M. Chandy and T. Lo, "The Capacitated minimum spanning tree", Networks Vol.3, pp.173-182, 1973.

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

[14] 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.

김 인 범

e-mail : ibkim@kimpo.ac.kr

1989년 서울대학교 컴퓨터공학과(공학사)

1991년 서울대학교 컴퓨터공학과(공학석사)

2007년 University of Wisconsin-Milwaukee, Department of Computer Science (PhD)

1991년~1996년 대우통신 종합연구소, Oracle 코리아

1996년~현 재 김포대학 IT학부 부교수

관심분야 : 그래프 및 네트워크 알고리즘, 컴퓨터 이론, 데이터베이스 등

김 수 인

e-mail : sikim@kimpo.ac.kr

1984년 광운대학교 전자공학과(공학사)

1991년 광운대학교 전자계산기공학과(공학석사)

2004년 광운대학교 컴퓨터공학과(공학박사)

1984년~1996년 갑일전자, 유캔컴퓨터

1996년~현 재 김포대학 IT학부 부교수

관심분야 : 영상인식, 스테레오비전, 로봇비전, 알고리즘 등