논문 2010-47CI-5-4

PTAS를 이용한 대형 스타이너 트리의 효과적인 구성

(Efficient Construction of Large Scale Steiner Tree using Polynomial-Time Approximation Scheme)

김 인 범*

(Inbum Kim)

요 약

스타이너 포인트들을 추가하여 모든 입력 노드들을 최단 길이로 연결하는 스타이너 최소 트리는 최소 신장 트리에 비해 전체 길이는 짧으나, 그것을 생성하는 문제는 NP-Complete 영역에 속한다. 이 문제를 위한 휴리스틱들은, 입력 노드의 수가 매우 큰 경우에는 많은 시간과 계산을 요구한다. 본 논문에서는 많은 입력 노드에 대해, 최하위 계층에서 포탈을 이용한 모든 가능한 단위 스타이너 트리들을 생성하고 각 상위 계층에서 이들을 계층별 병합 처리하여 최상위 계층에서 최소 비용의 트리를 선택하는 효과적인 PTAS 기법을 제안한다. 16,000개의 입력 노드와 최하위 계층에서 16개의 단위 영역으로 설계된 실험에서 생성된 PTAS 스타이너 트리는, pure 스타이너 트리의 길이에 비해 길이가 0.24% 증가되었으나, 생성 시간은 직렬 처리는 85.4%, 병렬처리는 98.9% 개선되었다. 따라서 제안하는 PTAS 스타이너 트리 생성 기법은 많은 입력 노드들에 대해 근사 스타이너 트리를 신속히 생성하는 응용에 잘 적용될 수 있을 것이다.

Abstract

By introducing additional nodes called Steiner points, the problem of Steiner Minimum Tree whose length can be shorter than Minimum Spanning Tree and which connects all input terminal nodes belongs to Non-Polynomial Complete domain. Though diverse heuristic methods can be applied to the problem, most of them may meet serious pains in computing and waiting for a solution of the problem with numerous input nodes. For numerous input nodes, an efficient PTAS approximation method producing candidate unit steiner trees with portals in most bottom layer, merging them hierarchically to construct their parent steiner trees in upper layer and building swiftly final approximation Steiner tree in most top layer is suggested in this paper. The experiment with 16,000 input nodes and designed 16 unit areas in most bottom layer shows 85.4% execution time improvement in serial processing and 98.9% in parallel processing comparing with pure Steiner heuristic method, though 0.24% overhead of tree length. Therefore, the suggested PTAS Steiner tree method can have a wide range applications to build a large scale approximation Steiner tree quickly.

Keywords: PTAS, Minimum Spanning Tree, Steiner Minimum Tree, Unit Area, Portal

I. 서 론

스타이너 최소 트리는 주어진 입력 노드에 대하여, 스타이너 포인트라는 새로운 노드들을 적절히 도입하여, 입력 노드를 모두 연결하는 최소 길이의 트리이다. 이것은 일반적으로 다항 적(Polynomial) 문제 영역에서의 최적화 트리인 유클리드 최소 신장 트리에 비해 더 단축된 트리 길이를 갖는다[1~2]. 스타이너 트리는 최소 신장 트리와 마찬가지로 네트워크의 라우팅, 송, 배전 전력선의 설계, 전자 회로 설계, 도로, 철도건설 및 선박이나 항공기의 항로 개발 등에 활용될 수 있으나 최소 신장 트리보다 더 좋은 결과를 기대할 수 있다. 그러나 스타이너 최소 트리를 생성하는 것은 비-다항 적(Non-Polynomial) 문제 영역에 속하는 것이므로, 적절한 휴리스틱을 개발하여 사용해야 할 것이고, 이 경우 노드의 수가 매우 클 때는 막대한 시간과 계산 양이 요구된다. 또한 매우 큰 전체 영역이 몇 개의 독립적인 영

* 정회원, 김포대학 IT학부
(School of Information Technology, Kimpo College)
접수일자: 2010년5월4일, 수정완료일: 2010년8월31일

역으로 분리되어 개별적으로 운용되고, 필요에 따라 다른 영역에 위치하는 특정 노드와 통신을 해야 할 경우, 전체의 노드의 위치 및 연결 정보가 모두 공개되지 않고 일부 공개되었을 경우가 있을 수 있다. 매우 많은 단말 노드들에 대해 분산처리를 하여 독립적으로 이들을 처리할 수 있다면, 전체 노드를 한꺼번에 다루는 것 보다는 트리 생성시간의 이점이 있다. 그러나 이 경우는 전체를 대상으로 결과를 얻은 가상의 최적 스타이너 최소 트리와 비교하여 트리의 길이가 커짐을 피할 수 없을 것이다. 따라서 전체 단말 노드를 각 단위 영역으로 적절히 분류하여 각 영역에서 독립적으로 단위 후보 스타이너 트리를 생성한 후, 상위 계층에서 이를 계층별 병합하여 최상위 스타이너 트리를 생성하며, 생성된 트리는 최적의 스타이너 트리와의 오차를 줄이는 방안이 필요하다.

본 논문에서 비-다항 적 문제인 Euclidean Traveler Salesman 문제 해결을 위해 고안된 PTAS(Polynomial Time Approximation Scheme)[3~4]를 응용하여, 많은 입력 노드들에 대하여 역시 비-다항 적 문제인 근사 스타이너 최소 트리를 생성하는 방법을 제안하고자 한다. 이 방법은 가상의 최적 스타이너 최소 트리와의 약간의 오차를 허용하면서, 포탈이라는 영역 분리선 상의 가상 노드들과 계층 병합하는 동적 프로그래밍 기법을 이용한다. 이 방법의 장점 중의 하나는 PTAS의 설계 방법에 따라 대형 근사 스타이너 최소 트리 생성시간의 단축에 초점을 맞출 것인지, 혹은 최적 트리와의 오차 감소에 초점을 맞출 수 있을 것인지를 선택할 수 있다는 점이다. 여기에 현재 다양하게 연구되고 있는 분산처리 혹은 병렬처리 기법을 활용한다면 더 효과적인 성능을 기대 할 수 있다.

본 논문은 Ⅱ장에서 제안하는 방법과 관련된 기존의 연구 내용을 기술하고, Ⅲ장은 PTAS 스타이너 트리 생성 방법에 대한 제안 내용이고, 본 논문의 제안 방법에 관한 구현 및 그에 따른 실험과 결과분석을 Ⅳ장에서 기술하고, Ⅴ장에서 결론을 맺는다.

Ⅱ. 관련 연구

입력 단말 노드가 아닌 스타이너 포인트라는 추가적인 노드들을 활용하여 입력 단말 노드들을 모두 연결하는 트리가 일반적인 스타이너 트리이고, 이들 중에서 가장 적은 비용의 트리가 스타이너 최소 트리이다. 이 문제는 비-다항 적 시간문제로 알려져 있다[1~2, 4]. 따라서 스타이너 최소 트리를 현실적인 문제에 활용하기 위해서는 적절한 휴리스틱을 이용한 근사 최소 스타이너 트리가 필요하다. 이러한 근사 최소 스타이너 트리는 네트워크의 라우팅, 송, 배전 전력선의 설계, 전자회로의 설계, 도로, 철도건설 및 선박이나 항공기의 항로 등 여러 분야에 활용될 수 있다.

최소 스타이너 트리와 관련된 주된 연구는 근사 스타이너 최소 트리를 생성하는 휴리스틱과 이를 현실적인 문제에 응용하는 방법, 그리고 생성된 트리의 근사 비율을 높이는 분야들이다. 원격 검침 시스템을 구성하는 검침기, 중계기, 집중기의 배치 및 연결을 근사 스타이너 트리를 이용에 관한 제안과, 그 방법에 의해 생성된 트리를 최소 신장 트리를 이용한 방법과 비교한 연구가 있다[5]. 스타이너 트리를 활용하여 멀티 캐스트 통신에서의 QoS(Quality of Service) 처리를 위한 다중 제약 멀티 캐스트 처리 알고리즘에 관한 연구가 있다[6]. 스타이너 트리 이론을 적용하여, 현재의 대전을 중심으로 한 우편물 중심 교환센터의 위치보다 효율적인 위치를 선정하는 방법에 관한 연구가 있었다[7]. 센서 네트워크의 효율적인 라우팅을 위하여 센서 노드들을 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 근사비율을 연구가 있다[8]. 직선 Steiner 최소 트리를 생성하는 연구로는 Polygonal Contraction 기법을 이용하여 직선 스타이너 최소 트리를 직선 최소 신장 트리로의 변환에 관한 연구가 있다[9]. 스타이너 트리를 VLSI 라우팅 및 설계에 응용한 연구로는 IC 설계 시 고려해야 하는 Obstacle-Avoiding Rectilinear Steiner Minimal Tree 문제 해결을 위한 라우팅 그래프와 알고리즘에 관한 연구이다[10].

PTAS(Polynomial Time Approximation Scheme)는 최적화 문제를 위한 근사 알고리즘 형태로, Polynomial Time 알고리즘의 모임이다. 즉, PTAS는 특정 문제에 대하여, 최적 해의 $\varepsilon > 0$의 오차를 허용하는 해를 찾는 알고리즘이다. PTAS를 이용하여 유클리드 Traveling Salesman Problem(TSP) 문제의, 최적의, 즉 최단 거리 여정의 $\varepsilon > 0$ 의 거리를 추가하는 여정을 찾는 것이 대표적인 예이다. Arora는 이 문제를 PTAS를 이용하여 평면상의 N개 노드에 대해, TSP 최적값의 $1+\varepsilon$ 근사를

$N^{O(1/\varepsilon)}$시간 내에 얻을 수 있음을 보였다[3]. 또한 스타이너 최소 트리 문제의 변형인 GOSST(Grade of Services Steiner Minimum Tree) 문제에 대해, J.Kim 등은 PTAS 기법을 적용하여, $1+\varepsilon$ 근사 트리를 다항적 시간 내에 얻을 수 있음을 증명하였다[4]. PTAS의 실행시간은 모든 고정된 ε와 노드 N에 대하여 다항 적이다. 그런데 실행 시간이 $O(N^{1/\varepsilon})$인 경우에, ε이 매우 작은 경우, 이 실행 시간은 지수 적으로 커질 수 있다.

EPTAS(Efficient Polynomial-Time Approximation Scheme)은 이 문제를 해결하기 위해 제안된 것으로, 실행시간이 $O(N^c)$가 되어야 한다. 여기서 c는 ε에 독립적인 상수이다. 이것은 문제의 크기 N의 증가가, ε과는 상관없이 실행시간에 상대적으로 같은 영향을 미치게 한다. 그러나 big-O 영향 하에서 상수 c는 ε에 임의로 의존할 수 있다. 이에 비해서, 실세계에서 유용하고, 제한적인 것이 FPTAS(Fully Polynomial-Time Approximation Scheme)이다. 이것은 알고리즘에 대해 문제의 크기인 N과 $1/\varepsilon$ 모두에게 다항 적이어야 함을 요구한다. 즉 FPTAS는 ε을 입력으로 받아 최적의 해와 $(1+\varepsilon)$의 근사 해를 N과 $1/\varepsilon$의 다항 함수 내에서 얻게 한다.

현재까지 발표된 근사 최소 스타이너 트리 구성을 위한 휴리스틱들 중에서, 외접 삼각형을 이용한 방법이 유명하다[11]. 이 방법은 트리에 존재하는 인접한 두 간선에 존재하는 세 정점 A, B, C를 각각 연결하는 선분을 생성하고, 이 가운데 가장 큰 선분을 $\overline{AB}$라고 할 때, 이것을 한 변으로 하는 정삼각형 $\triangle ABQ$를 생성한다. 직선 $\overleftrightarrow{CQ}$와 $\triangle ABQ$의 외접원과의 교점을 S라고 할 때, 이 정점을 스타이너 포인트로 정의한다. 입력 트리를 구성하는 모든 간선의 인접한 쌍에 대해 이와 같은 방법으로 스타이너 포인트들을 생성하고 이를 활용하여 적절한 처리를 통해 스타이너 트리를 생성한다.

본 논문에서 제안하는 방법은 관련 연구[3~4]에서 이용한 정의와 개념을 일부 이용한다. PTAS를 적용하기 위해, 주어진 입력 노드가 위치한 평면을 문제 영역으로 간주하고, 이를 직사각형 형태로 분할한다. 분할된 각 직사각형의 모든 변은 좌표축과 수직이거나 평행하다. 찾고자 하는 최적의 트리 구조를 최적 연결 구조라 하고 최적이 아닌 구조는 연결 구조라고 정의한다. 비용은 연결구조가 갖는 선분 길이의 총합이다. 직사각형 R의 분할 선은 R을 $m:n$의 비율로 두개의 하위 직사각형으로 분할하는 선분으로 영역 분리선이라 정의한다. 이 분할을 기반으로 본 논문에서는 동적(Dynamic) 프로그래밍을 수행하게 된다. 본 논문의 실험에서는 간단하게 이 분할 비율을 1:1로 정의하였다. 이 분할과정은 하위 직사각형의 크기가 미리 정의된 크기에 도달하면 종료된다. PTAS에서 중요한 역할을 담당하는 포탈(Portal)은 직사각형 R을 하위 직사각형들로 분할 할 때, 영역 분리선 상의 특정 위치에 존재하는 점으로 정의한다. 그 위치는 외부에서 조절할 수 있으나, 문제 영역을 구성하는 모든 계층의 사각형에서의 인접한 두 포털 사이의 간격은 동일하다. 또한 각 분할 과정에서 생성되는 포탈의 수도 일정하다. 인접한 영역의 연결 구조를 병합할 때 사용되는 연결선은 이 포탈에서만 만날 수 있다.

III. 본 론

본 논문에서 제안하는 방법은 각 영역의 연결구조가 이웃하는 영역의 연결 구조와 병합할 때, 반드시 영역 분리선에 위치하는 포털을 통해서만 연결된다. 최적 해와의 근사도를 높이기 위해 최하위 계층에서 한 영역을 둘러싸는 영역 분리선의 모든 포털들의 모든 조합과 해당 영역에 포함된 입력 노드들을 이용해서 부분 연결 구조, 즉 단위 스타이너 트리를 생성한다. 각 계층의 모든 포털들의 조합으로 생성된 연결 구조과 이들의 병합으로 생성된 부모 연결 구조들 중에서 최상위 계층에서 최소 길이의 연결 구조를 구성하는 하위 부분 연결 구조들이 최종 선택된다.

대형 근사 스타이너 최소 트리 문제에, PTAS를 적용하기 위해 분할 설계된 계층구조는, 본 논문에서는 최하위 계층의 16개 기본 영역이 존재하고, 그것의 차상위 계층에서는 최하위 계층의 인접한 2개의 셀이 병합되어 8개의 부분 영역이 생성된다. 같은 방법으로 상위 계층의 각 영역은 그것의 하위계층의 인접한 2개의 영역을 병합하여 생성된다. 최상위 계층은 각 하위계층에서의 라인 분리선과 포털들의 모든 조합에 의해 생성되는 모든 연결구조에 관한 정보가 생성되고, 이 가운데 최소 비용의 연결구조가 선택된다.

본 논문에서 제안하는 방법은 후처리 과정이 따르는데, 이는 특정 포털을 통해 영역 분리선 좌, 우 영역의 인접한 연결 구조가 병합이 될 때, 삼각형 정리에 따라 포털을 경유해서 연결하는 다른 영역의 두 노드들을 직접 연결하는 것으로, 이는 생성되는 트리의 최종 연결

표 1. PTAS 근사 스타이너 최소 트리 생성 단계
Table 1. Steps for building PTAS approximation Steiner minimum tree.

단계	처리내용
1	각 라인 분리선에 대해 포털을 생성
2	최하위 계층의 각 기본 영역에 대해, [단계 1]에서 생성한 포탈의 조합과 소속 입력 노드에 대해 적절한 휴리스틱을 이용하여 스타이너 포인트를 생성하여, 모든 가능한 단위 스타이너 트리를 생성
3	각 영역의 생성된 단위 스타이너 트리에서 연결 차수가 1인 스타이너 포인트와 이와 관련된 연결을 삭제
4	[단계 3]에서 생성된 인접한 두 영역의 단위 스타이너 트리를 영역 분리선상의 포털의 모든 조합을 이용하여 서로 연결하여 상위 계층의 스타이너 트리를 생성하고 각 트리와 관련된 정보들을 해당 테이블 엔트리에 저장
5	최상위 계층의 스타이너 트리가 완성될 때까지 차 상위 계층별로 [단계 4]를 반복
6	최상위 계층의 테이블 정보를 이용하여 최소 비용의 스타이너 트리를 찾고, 이것의 하위 계층을 구성하는 스타이너 트리 찾음
7	포털과 연결된 인접한 두 영역의 노드들을 직접 연결하는 후-처리 작업을 시행하여, 최종 PTAS 스타이너 최소 트리를 생성

길이를 감소시킬 수 있다.

표 1은 본 논문에서 제안하는 PTAS를 이용한 대형 근사 스타이너 최소 트리를 생성하는 과정이다. 주어진 입력 노드에 대하여 완전 연결을 생성하고, 이에 대한 1차 최소 신장 트리를 생성하며, 이 트리에 존재하는 모든 인접한 두 간선에 대하여 2장 관련 연구에서 기술한 방법을 활용하여 스타이너 포인트들을 생성한다[11]. 생성된 스타이너 포인트들을 입력 노드에 연결한 후 인접한 두 해당 간선은 삭제하여 스타이너 그래프를 생성한다. 생성된 스타이너 그래프의 노드와 간선에 대하여 2차 최소 신장 트리를 생성한 후에 연결차수가 1인 스타이너 포인트들은 삭제하는 등의 후처리 작업을 시행하면 일반적인 스타이너 최소 트리가 결정된다. 이 휴리스틱의 실행시간에서 입력 노드 n에 대하여, 유클리드 최소 신장 트리 생성을 위한, 완전 연결 생성 시간 $O(n^2)$과 Prim의 알고리즘을 이용한 1차 최소 신장 트리의 생성시간은 $O(n\log n)$ 그리고 생성된 최소 신장 트리의 E개의 간선에서 두 개의 인접한 연결 간선에 대한 스타이너 포인트 생성시간 ${}_EC_2 = O(n^2)$이다. 또한 생성 가능한 스타이너 포인트의 수가 $O(n^2)$이므로 이것과 기존의 입력 노드에 의해 생성되는 2차 최소 신장 트리의 생성 시간은 $O(n^2\log n^2)=O(n^2\log n)$이 된다. 후처리 작업 등을 감안하더라도 스타이너 최소 트리의 생성 시간은 $O(n^2\log n)$이다.

본 논문에서 제안하는 PTAS 최소 신장 트리의 생성 시간은 최하위 계층에서의 각 기본 영역에서의 단위 스타이너 트리의 생성 시간과 각 계층의 연결구조의 테이블 엔트리 생성 및 테이블 검색시간이 포함된다. 최하위 계층의 각 단위 영역에서의 모든 단위 스타이너 트리의 생성 시간은, 직렬 처리의 경우:

$$\sum_{i=1}^{k}(m\times O(n_i^2\times\log n_i)) = m\times\sum_{i=1}^{k}O(n_i^2\times\log n_i) \approx O(n^2\times\log n) \quad (1)$$

병렬처리의 경우:

$$Max_{0\le i\le k}(m\times O(n_i^2\times\log n_i)) = m\times Max_{0\le i\le k}O(n_i^2\times\log n_i) \approx O(n^2\times\log n) \quad (2)$$

여기서 k는 최하위 계층을 구성하는 기본 영역의 수이다. n_i는 기본 영역 i에 포함된 입력 노드의 수이고, 전체 N개의 입력노드의 영역별 분포가 거의 균등하다고 가정할 때, n은 약 N/k이다. r은 영역 분리선 상에 존재하는 포탈의 모든 수이고, m은 영역 분리선에서의 선택 가능한 포털들의 조합, 즉 $\sum_{i=1}^{r}{}_rC_i$이다. 본 논문에서는 단순화하기 위해 최하위 계층에서 r은 4, 그 밖의 계층에서는 2로 설정하였으나, 문제 영역에 따라 높은 근사도를 필요로 하면 r을 크게 하여 높은 정밀도로 결과를 얻을 수 있다. 그러나 이 경우 실행시간의 증가는 필연적이다. 최하위 계층에서의 생성된 두 인접 영역의 단위 스타이너 트리가 다음 상위 계층에서 병합이 될 때, 최소 거리로 연결 가능한 포털은 4개의 영역 분리선 가운데 인접한 영역 분리선 위에 있는 두 개의 포털이므로 최하위 계층에서 실제로 후보 포탈로 고려해야 할 것은 다른 상위 계층과 마찬가지로 두 개 이고, 이중에서 한 개가 될 것이다. 따라서 본 논문에서 고려할 m은 2가 된다. 즉 각 계층의 연결 구조 정보를 저장하는 테이블들의 모든 엔트리의 합 E는 영역 분리 선에서의 총 포털 수가 2이고 이 가운데에서 1개를 선택하는 경우, 다음 식으로 표현이 될 수 있다.

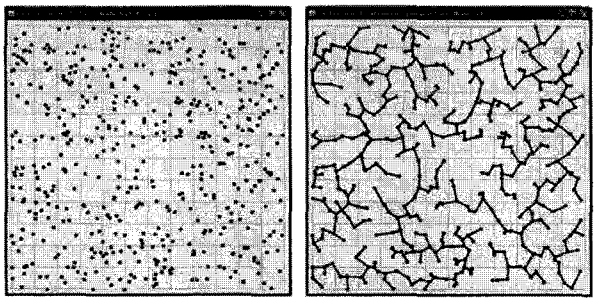
$$E=\sum_{p=1}^{d}(2^{2^p+d-(p-1)}) \tag{3}$$

여기서 d는 PTAS를 구성하는 계층 수를 의미한다. 이것은 입력 노드의 수에 독립적이고, PTAS 설계 방법에 따라서 결정될 수 있으므로, 본 논문에서 제안하는 방법의 실행시간 T 는 다음과 같이 정리된다.

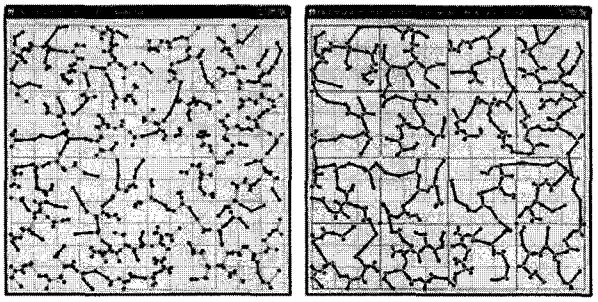
$$\begin{aligned} T &= O(n^2\times\log n)+t_1\times E+t_2\times E \\ &\approx O(n^2\times\log n)\le O(N^2\times\log N) \end{aligned} \tag{4}$$

위의 식에서 t_1과 t_2는 각 테이블의 엔트리 생성 시간과 그것들의 검색 시간이다. 그러나 테이블의 부분 최소 신장 트리들의 정보를 위한 엔트리의 생성 및 검색 시간은 최하위 계층에서의 연결 구조를 생성하는 시간에 비해 매우 작은 시간이므로 무시될 수 있다.

그림 1의 (a), (b), (c), (d)는 무작위로 생성된 500개의 입력 단말 노드, 이에 대한 생성된 최소 신장 트리와 pure 스타이너 트리, 그리고 본 논문에서 제안한 방법에 의해 생성된 트리이다. 그림 2는 표 1에서 기술한 본 논문에서 제안된 방법의 예로 최하위 계층의 기본 영역의 수가 16개이고, 최하위 계층의 영역 분리선 상의 최대 포탈은 4개, 그 외 계층에서는 2개 이며, 이 중 선택되는 포탈의 수는 한 개로 설계된 PTAS의 계층 구조를 설명한다. 그림 (a)와 (b)에는 최하위 계층의 16개의 기본 영역 중에서 영역 1과 영역 8에 관한 것이다. 영역을 의미하는 사각형의 네 꼭짓점에 후보 포탈이 존재하는 것으로 그림에서 원으로 표현되었다. 이 포탈은 다음 계층에서의 인접한 기본 영역과 병합하기 위해 사용되는 가상 노드이다. 최하위 계층에서는 다른 계층과는 달리 고려되는 영역 분리선이 최대 4이고, 총 4개의 포탈이 생성된다. 선택되는 포탈은 이 가운데서 한 개가 선택되어 빨간색의 채워진 원으로 표시되었는데, 그 선택은 최상위 계층에서 결정된다. 그림 2의 (c)는 계층 1의 영역1과 영역 8이 병합되는 계층 2를 표현한 것으로 영역 분리선에 있는 두개의 포탈 가운데 하나를 사용하여 병합된 스타이너 트리들을 생성하게 되는데, 이 중 최상위 단계에서 선택 연결 구조가 그림 3의 (c)에 표현되었다. 같은 방법으로 생성된 계층 2의 영역 2의 스타이너 트리 구조가 그림 3의 (d)에 표현되어 있다. 이들을 계층 3에서 병합하기 위한 영역 분리선 및 포탈이 그림 2의 (d)에 있고, 이것에 의해 병합 생성된 계층 3에서의 스타이너 트리 구조가 그림 3의 (d)에 있다. 계층 4와 최상위 계층인 계층 5를 위한 라인 분리선 및 포탈 위치가 그림 2의 (e)와 (f)에 표현되어 있으며, 이것에 따라 인접한 두 하위 계층인 영역 1과 영역2의 스타이너 트리와 이들이 병합되어 생성된 스타이너 트리



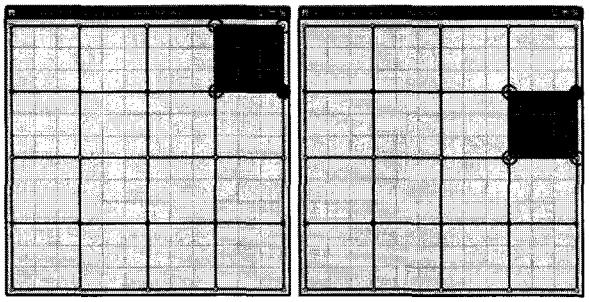
(a) 입력 단말 노드 (b) 최소 신장 트리



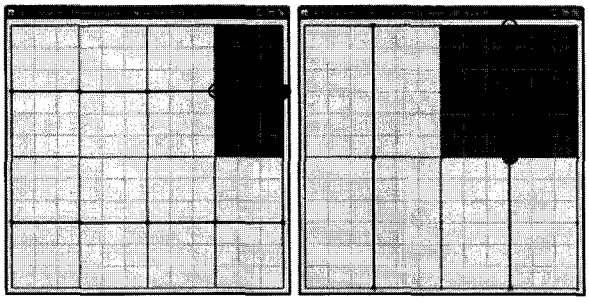
(c) pure 스타이너 최소 트리 (d) ptas 스타이너 최소 트리

그림 1. 500개 단말 노드 및 각 방법에 따라 생성된 트리

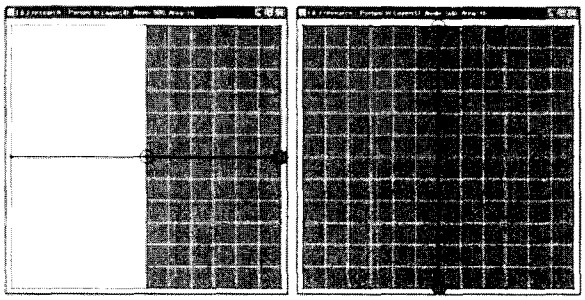
Fig. 1. 500 input terminal nodes and built three trees.



(a) 계층 1의 영역 1 및 포털 (b) 계층 1의 영역 2 및 포털



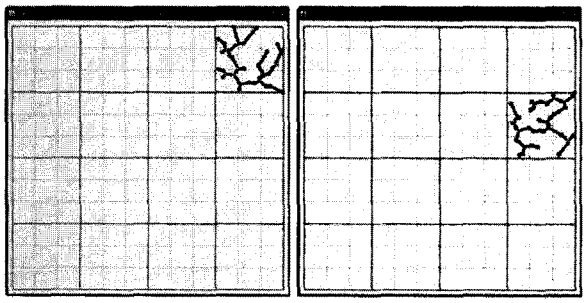
(c) 계층 2에서의 병합 및 포털 (d) 계층 3에서의 병합 및 포털



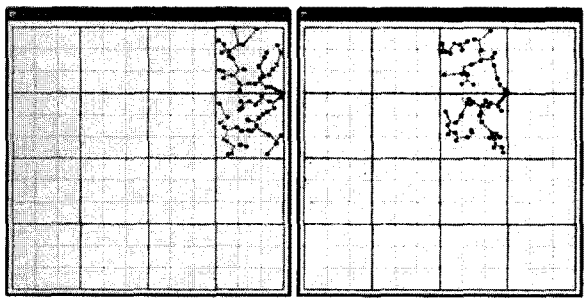
(e) 계층 4에서의 병합 및 포털 (f) 계층 5에서의 병합 및 포털

그림 2. 제안된 방법에서 각 계층 영역 및 사용되는 분리선상의 포탈

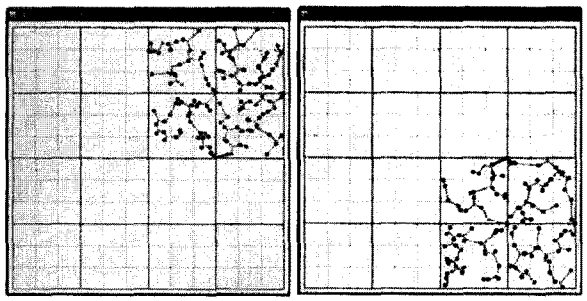
Fig. 2. Layers and portals on separating lines.



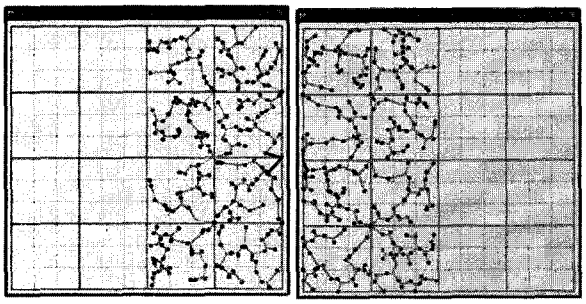
(a) 계층 1의 영역 1의 연결 구조 (b) 계층 1의 영역 8의 연결 구조



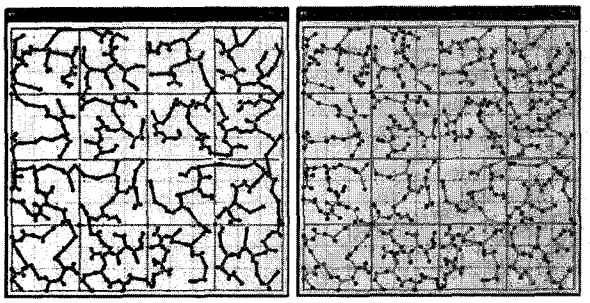
(c) 계층 1의 1, 8을 병합, 계층 2의 영역 1 생성 (d) 계층 2의 영역 2의 연결 구조



(e) 계층 2의 1과 2를 병합, 계층 3의 영역 1 생성 (f) 계층 3의 영역 4



(g) 계층 3의 1과 4를 병합, 계층 4의 영역 1 생성 (h) 계층 4의 영역 2



(j) 계층 5를 후처리한 최종 스타이너 최소 트리 (i) 계층 4의 1과 2를 병합, 계층 5영역 생성

그림 3. 500개 입력노드에 대한 PTAS 스타이너 최소 트리 생성 과정

Fig. 3. A process of building PTAS Steiner minimum tree for 500 input terminal nodes.

가 계층 3의 경우에는 그림 3의 (e), (f), (g)에 표현되었고, 계층 4의 경우에는 그림 3의 (g), (h), (i)에 나타나 있다. 그림 3의 (j)는 그림 (i)의 최상위 계층에서 선택된 스타이너 트리에서 각 계층의 영역 분리선의 포탈을 경유하는 연결을 직선으로 연결하여 최종 생성되는 트리의 길이를 단축시키기 위한 후처리 작업의 결과를 보인다.

Ⅳ. 실 험 및 분 석

실험에 사용된 인자는 입력 노드의 수와 트리 생성 방법이고, 관찰 결과는 각 방법에 의해 생성되는 트리의 길이와 실행시간이다. 실험의 목적은 본 논문에서 제안하는 PTAS 스타이너 트리 방법이, 최적 트리의 길이와 근사한 트리를 얼마나 짧은 실행시간 내에 생성하는 것을 검증하기 위함이다. 본 논문에서 제안하는 방법에 의해 생성된 트리는 최소 신장 트리와 pure 스타이너 트리와 비교된다. 최소 신장 트리는 다항 적(Polynomial) 방법에 의해 최적의 트리임이 증명되었고,

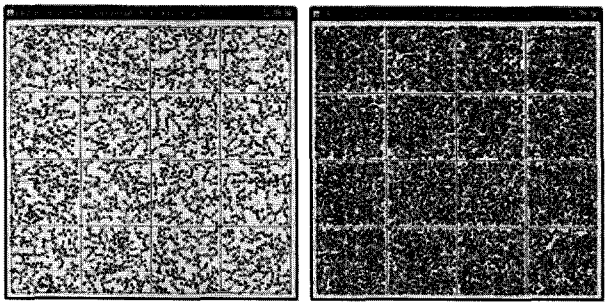


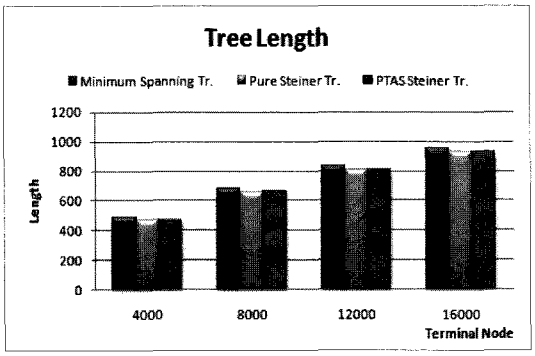
그림 4. 입력노드가 4000개(좌)와 16000개(우) 경우 생성된 PTAS 스타이너 트리

Fig. 4. Built PTAS Steiner minimum tree for 4000 (left) and 16000(right) input terminal nodes.

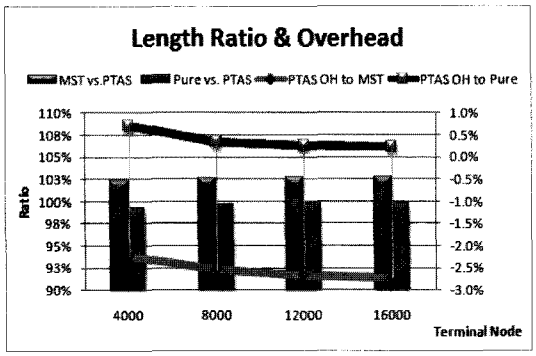
pure 스타이너 트리 방법은 비-다항 적(Non-Polynomial) 문제로 이에 대한 적절한 휴리스틱으로 다항식 방법에 비해 최단의 트리를 생성할 수 있다. 제안된 방법의 실제 구현을 보다 쉽게 하기 위해 영역 분리선에 위치하는 전체 포탈의 수는 2 또는 4이고 이 중에서 한 개가 사용되어 각 계층에서 연결 구조를 병합한다. 실험을 위해 무작위로 생성된 입력 노드의 수는 4000, 8000, 12000, 16000 이고 각 노드의 위치는 2차원 평면에서 중복되지 않고 각 계층에서 사용될 영역 분리선에서 어느 정도 떨어진 곳에 생성된다. 최하위 계층을 분할하는 기본 영역의 수는 16으로 한정하였으나, 이는 응용에 따라 변경이 가능하다. 각 영역에 속한 입력 노드의 분포가 거의 균등하다고 가정할 때, 기본 영역의 수가 많을수록 각 영역에 포함되는 입력 노드의 수는 적어질 것이다. 또한 최하위 계층의 기본 영역의 수가 적을수록, 전체 노드를 대상으로 하는 pure 스타이너 최소 트리의 특징을 많이 보이게 될 것이다. 그림 4는 입력 노드의 수가 4000과 16000인 환경에서, 본 논문에서 제안하는 방법으로 생성된 PTAS 스타이너 최소 트리의 결과이다. 최소 신장 트리는 Prim의 최소 신장 트리 생성 알고리즘으로 생성하였고[12], pure 스타이너 트리는 기존의 스타이너 포인트 생성 휴리스틱을 활용하여 생성하였다[11]. 실험 환경은 Intel 1.83 GHz (T5600) 프로세서와 4기가 램의 랩탑 컴퓨터이고, 알고리즘은 C++와 Java로 구현하였다.

1. PTAS 스타이너 트리의 길이 및 분석

그림 5의 (a)와 (b)는 입력 노드의 수에 따라 생성된 최소 신장 트리, pure 스타이너 트리, 그리고 제안된 방법에 의한 PTAS 스타이너 트리의 길이를 비교한 결과이다. 입력 노드가 많을수록 PTAS 스타이너 트리는



(a) 입력 노드 수에 따른 생성된 각 트리의 길이



(b) 입력 노드 수에 따라 생성된 PTAS 스타이너 트리의 길이 비교 및 분석

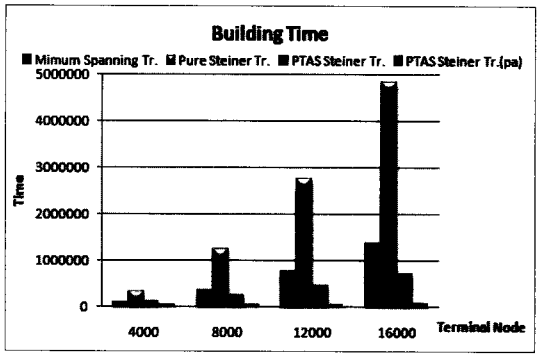
그림 5. 단말 노드수의 변화에 따른 PTAS 스타이너 트리의 길이 비교

Fig. 5. Length comparison of built PTAS Steiner minimum trees for 4 types of input terminal node number.

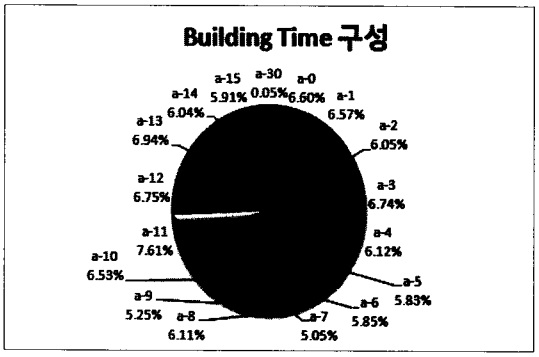
pure 스타이너 트리의 길이와 거의 근접함을 보이고 있으며, 반면에 최소 신장 트리와 비교된 길이의 절감율은 점차 커짐을 확인할 수 있다. 입력 노드 수가 4000개인 경우, PTAS 스타이너 트리는 pure 스타이너 트리에 비해 트리의 길이가 0.71% 증가했으나, 1600개인 경우에는 증가된 트리의 길이가 0.24%로 증가 폭이 대폭 감소됨을 알 수 있다. 최소 신장 트리에 비해서는 입력 노드가 4000인 경우에는 PTAS 스타이너 트리의 길이가 2.25%의 감소가 발생하였으나, 16000개인 경우에는 2.73% 감소되었다. 따라서 본 논문에서 제안하는 방법은 입력 노드의 수가 매우 많을 때, 가장 짧은 길이의 트리를 생성하는 pure 스타이너 트리 방법과 거의 근접한 결과를 얻을 수 있을 것이다.

2. 실행 시간

그림 6의 (a)는 단말노드의 수에 따라, 각 트리를 생성하는데 소요되는 실행시간을 나타낸다. 다른 방법에

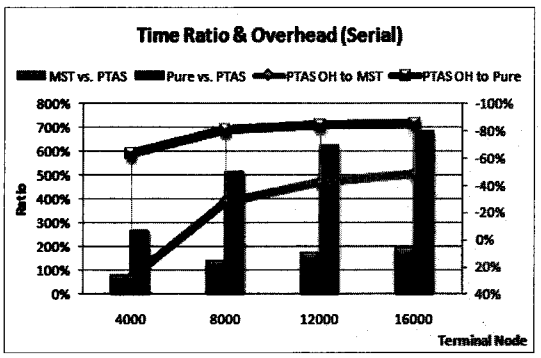


(a) 입력 노드 수에 따른 각 트리의 생성 시간

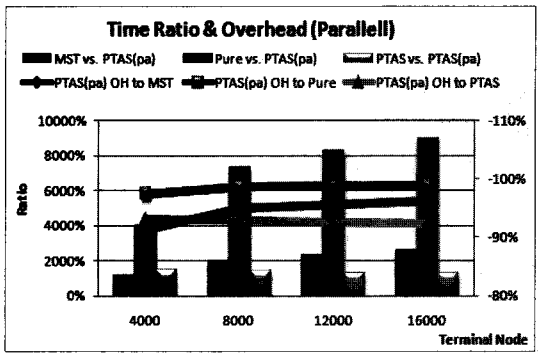


(b) 생성된 PTAS 스타이너 트리의 생성 시간 비교 및 분석

그림 6. 단말 노드수의 변화에 따른 트리 생성 시간
Fig. 6. Execution time for building PTAS Steiner minimum trees with 4 types of input terminal node number.



(a) 입력 노드 수에 따른 생성된 직렬 PTAS 스타이너 트리의 생성 시간 비교 및 분석



(b) 입력 노드 수에 따른 생성된 병렬 PTAS 스타이너 트리의 생성 시간 비교 및 분석

그림 7. 각 트리 생성 방법별 단말 노드수의 변화에 따른 트리 생성 시간 분석
Fig. 7. Execution time comparison and analysis of built PTAS Steiner minimum tree for 4 types of input terminal node number.

비해 제안된 방법에 의한 트리 생성 시간이 매우 작음을 확인할 수 있다. 그림 6의 (b)는 16000개의 입력 노드에 대해, PTAS 스타이너 트리의 생성 시간을 분석한 것이다. 최하위 계층에서 각 영역 단위로 입력 노드와 선택된 포탈을 대상으로 단위 스타이너 트리를 생성하는 시간이 전체 시간의 99.95%를 차지하고 있다. 최하위 계층에서 각 영역별 생성 시간은 전체 시간의 최소 5.05%부터 최대 7.61%를 차지하고, 이중 영역 a-11이 가장 많은 시간을 사용한다. 최하위 계층 생성 시간과 비교할 때, 다른 상위 계층에서의 사용 시간은 상대적으로 매우 적음을 확인할 수 있다. 최상위 계층에서의 소모되는 시간은 전체시간의 0.05% 이었다.

그림 7은 처리 방식에 따른 PTAS 스타이너 트리의 생성시간에 대한 분석이다. 그림 7 (a)의 직렬처리 방법은 각 계층에 속한 모든 영역을 정해진 순서대로 하나씩 처리하는 방법이다. 입력 노드 수가 4000이고 16개의 기본 영역으로 설계된 직렬 PTAS 스타이너 트리는 최소 신장 트리의 생성 시간보다 28.39%의 시간이 더 필요했으나, 입력 노드가 16000인 경우에는 48.54%의 시간을 절감할 수 있었다. 또한 pure 스타이너 트리의 생성 시간과 비교해서는, 입력 노드수가 4000인 경우에는 63.36%, 16000인 경우에는 85.41%의 생성 시간을 개선할 수 있었다. 그림 7의 (b)는 각 PTAS 스타이너 트리를 병렬 처리 방법으로 생성했을 때 소요된 시간과 그에 대한 분석결과이다. 병렬 처리에서는 최하위 계층의 기본 영역 별로 독립적인 프로세서를 각각 할당하여 각 단위 스타이너 트리를 생성한다. 이 때, 최하위 계층에서의 소요되는 시간은, 가장 많은 시간이 필요한 기본 영역의 단위 스타이너 생성 시간이다. 그림 6의 (b)에서 입력 노드가 16000개인 PTAS 스타이너 트리를 병렬 처리로 생성할 때, 최하위 계층의 각 영역 중에서 생성 시간을 최대로 소모하는 영역은 a-11로 전체 7.61%를 소모한다. 따라서 이 시간을 최하위 계층에서의 생성시간으로 정의한다. 병렬 처리 방법에서 최하위

계층을 제외한 상위 계층에서의 연결구조는 직렬 처리 방법과 유사하게 생성된다. 실행시간의 대부분을 최하위 계층의 각 영역에서 소비됨을 분석을 통해 확인할 수 있었으므로, 그 외의 계층에서의 처리는 어떤 방식을 이용하더라도 큰 차이는 없을 것이다. 그림 7의 (b)에서, 직렬 방식과 동일하게, 입력 노드가 많아질수록 병렬 처리 방식은 PTAS 스타이너 트리를 생성하는 시간을 많이 절감할 수 있었다. 입력 노드의 수가 16000인 경우에는 최소 신장 트리에 비해 96.09%의 생성 시간의 개선을 보였고, pure 스타이너 트리 생성 시간에 비해 98.39%의 개선을 보였다. 또한 그림 7의 (b)에는 직렬 처리 방식과 병렬 처리 방식의 PTAS 스타이너 트리의 생성 시간을 비교가 있는데, 입력 노드가 16000인 경우에는 병렬 처리 방식이 직렬처리 방식에 비해 92.39%%의 개선이 있었다. 따라서 병렬처리가 가능한 환경에서는 최하위 계층의 단위 영역에서 스타이너 트리를 생성하는 과정을 여러 개의 프로세서로 각기 실행할 수 있도록 설계한다면, 최소 신장 트리 방법, pure 스타이너 트리 방법, 그리고 직렬 PTAS 스타이너 트리 방법에 비해 매우 짧은 시간에, 근사 최적화된 트리를 생성할 수 있을 것이다.

V. 결 론

본 논문에서는 대형 스타이너 트리의 생성 문제에 대한 방안을 제안하였다. 이것은 일반적인 스타이너 트리를 생성하는 기존의 방법과는 달리, 전체 입력 노드를 한 번에 처리하지 않고, 문제 영역을 계층별로 구분하고 최하위 계층을 몇 개의 단위 영역으로 나누어 각 영역에 위치하는 노드들과 경계선 상에 위치하는 가상의 노드인 포털의 모든 조합을 이용하여 단위 스타이너 트리 구조를 생성한 후에, 적절한 포탈을 이용하여 각 인접한 두 영역의 단위 스타이너 트리들을 계층적으로 병합하여 차 상위 계층의 스타이너 트리를 형성한다. 최상위 계층에서는 하위 스타이너 트리들의 병합으로 형성된 모든 최상위 계층에서의 스타이너 트리들 중에서 최소 비용의 것을 선택하고, 적절한 처리 작업을 수행하여 최종 PTAS 스타이너 트리를 생성한다.

본 논문에서 제안된 방법은 실생활에서 발생하는 여러 형태의 대용량 문제들을 매우 신속하게 해결할 수 있을 것이다. 본 논문의 실험 결과는 16,000개의 입력 노드와 16 개의 기본 영역 환경에서 PTAS 스타이너 트리를 생성했을 때, 입력 노드 전체를 대상으로 한 pure 스타이너 트리에 비해 0.24% 길이의 증가를 보였으나, 실행 시간은 직렬 처리 경우에는 85.41%, 병렬처리 경우에는 98.89%의 개선을 보였다. 또한 입력 노드가 많을수록, 트리의 길이나 생성 시간 측면에서 효과적임을 보였다. 이것은 본 논문에서 제안하는 PTAS 스타이너 트리 생성 방법이 많은 입력 노드들을 거의 최적 트리와 비슷한 길이의 트리를 매우 빠른 시간에 생성하여 활용하는 응용에 적합함을 보인다. 병렬 처리가 가능한 환경이라면 더 좋은 실행시간의 개선을 보일 수 있을 것이다. 그러나 본 논문에서 제안하는 방법은 문제 처리 과정을 단순화하기 위해 기존의 PTAS에 상당한 제한을 부여했다. 이것은 분명 최적 해와의 근사도에 영향을 미칠 것이나, 만약 기존의 PTAS 방법을 그대로 사용한다면 매우 큰 실행시간과 많은 저장 공간이 필요할 것이고 이는 실용성에 문제를 발생할 것이다.

향 후 연구는, 다른 대규모 비-다항식적 문제에 대하여 본 논문에서 제시한 기법을 적용한 방안에 대한 연구가 될 것이다. 이를 통해 실행시간과 기억장소의 절약을 얻게 될 것이다. 또한 본 논문에서 제안된 방법은 노드들이 비교적 균등하게 분포될 때, 효과적이므로, 비균등 분포인 문제에 대한 연구가 더 필요하다.

참 고 문 헌

[1] 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.

[2] http://en.wikipedia.org/wiki/Steiner_tree, 2010.

[3] S. Arora, "Polynomial-time Approximation Schemes for Euclidean TSP and Other Geometric Problems," Journal of ACM, Vol.45, issue 5, pp.753-782, 1998.

[4] X. Cheng, J.M. Kim and B. Lu, "A Polynomial Time Approximation Scheme for the Problem of Interconnecting Highways," Journal of Combinatorial Optimization, Vol. 5, issue 3, pp.327-343, 2001.

[5] 김재각, 김인범, 김수인, "원격 검침 시스템에서 근사 최소 스타이너 트리를 이용한 집중기 및 중계기의 효율적인 배치와 연결", 한국통신학회 논문지 B, Vol.34 No.10, pp.994-1003, 2009.

[6] 이성근, 한치근, "다중 제약이 있는 멀티캐스트 트

리 문제에 관한 연구", 한국 인터넷정보학회 논문지, Vol.5, No.5, pp.129-138, 2004.
[7] 양성덕, 유용규, 이상중, "Steiner Tree 이론을 이용한 우편물 교환센터의 최적 위치 선정", 한국 조명 전기설비학회 논문지. Vol.22, No.9 pp.82-87, 2008.
[8] 김준모, "센서 네트워크 구축에서의 Combinatorial 기법 적용", 대한 전자공학회 논문지 TC, Vol.45, No.7, pp.9-16, 2008.
[9] C. Liu, S. Yuan, S. Kuo, S. Wang, "High-performance Obstacle-avoiding Rectilinear Steiner Tree Construction," ACM Transactions on Design Automation of Electronic Systems, Article 45, Vol.14, No.3, 2009.
[10] Y. Wang, X. Hong, T. Jing, Y. Yang, X. Hu, G. Yan, "The Polygonal Contraction Heuristic for Rectilinear Steiner Tree Construction," Proceedings of the 2005 Asia and South Pacific Design Automation Conference, Shanghai, China, pp.1-6, 2005.
[11] B. Bell, "Steiner Minimal Tree Problem," http://www.css.taylor.edu/~bbell/steiner/, 1999.
[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, Introduction to Algorithm, 2nd ed., MITPress, pp.525-579, 2001.

저 자 소 개

김 인 범(정회원)
1989년 서울대학교 컴퓨터공학과 학사 졸업
1991년 서울대학교 컴퓨터공학과 석사 졸업
2007년 위스컨신 주립대-밀워키 전산학과 박사 졸업
1996년~현재 김포대학 IT학부
<주관심분야 : 네트워크 알고리즘, 데이터베이스, 컴퓨터 이론>