韓國經營科學會誌
第25卷 第3號
2000年 9月

열 생성 기법을 이용한 스타이너 나무 분할 문제에 관한 연구

정규웅* · 이경식** · 박성수* · 박경철*

Column Generation Approach to the Steiner Tree Packing Problem

Guewoong Jung* · Kyungsik Lee** · Sungsoo Park* · Kyungchul Park***

Abstract

We consider the Steiner tree packing problem. For a given undirected graph $G = (V, E)$ with positive integer capacities and non-negative weights on its edges, and a list of node sets (nets), the problem is to find a connection of nets which satisfies the edge capacity limits and minimizes the total weights. We focus on the switchbox routing problem in knock-knee model and formulate this problem as an integer programming using Steiner tree variables. The model contains exponential number of variables, but the problem can be solved using a polynomial time column generation procedure. We develop a branch-and-price algorithm based on this polynomial time column generation procedure. We test the algorithm on some standard test instances and compare the performances with the results using cutting plane approach.

Computational results show that our algorithm is competitive to the cutting plane algorithm presented by Grotschel et al. and can be used to solve practically sized problems.

1. 서 론

VLSI (Very Large Scale Integration)는 실리콘에 큰 회로들을 구현하는 기술로 마이크로 프로세서, 신호 프로세서, 대용량 메모리, 입출력 컨트롤러, 상호연결망 등을 구성하는데 성공적으로 사용

* 한국과학기술원 산업공학과
** 한국전자통신연구원
*** 한국통신 통신망연구소

되어져 왔고 오늘날 거의 모든 전자기기들의 핵심 기술이라 할 수 있다 [20]. 이러한 VLSI 회로는 수천 또는 수만 개의 트랜지스터로 구성되어져 있기 때문에, VLSI 회로 설계는 매우 복잡하고 어렵다. 이러한 어려움을 극복하기 위해 VLSI 회로는 보통 두 단계 (논리적 설계, logical design와 물리적 설계, physical design)를 거쳐 설계된다. 논리적 설계 단계에서는 어떤 논리 단위장치(셀)를 사용하고 어떤 셀들을 연결하여 회로를 구현할 것인가를 결정한다. 셀에는 접촉점(터미널)들이 있는데, 이 점들을 거쳐서 전선으로 셀들은 연결된다. 연결해야 하는 터미널들의 집합을 네트(net)라고 부른다. 논리적 설계 단계를 통해 셀과 네트들의 리스트가 결정되면, 물리적 설계 단계에서는 셀들의 위치를 결정하고 정해진 설계 규칙아래 주어진 목적식을 최소화하는 네트들의 연결 경로를 찾는다. 이러한 물리적 설계 문제는 *NP-hard* 이고, 더군다나 대부분의 실제 문제에 있어서 셀과 네트의 수가 상당히 많기 때문에 최적까지 푸는 것은 불가능하다. 따라서, 이런 내재된 복잡도를 줄이기 위해 일반적으로 몇 개의 부문제로 나누어 해결하게 된다 먼저 셀들의 적절한 위치를 결정한다(위치 선정 문제, placement problem). 그 다음에는 네트들을 연결할 전선의 대략적인 경로를 결정한다(전체 배선 문제, global-routing problem). 대략적인 경로가 구해지면, 배선 영역 내에서 전선들의 정확한 경로를 결정한다(세부 배선 문제, detailed-routing problem). 마지막으로, (필요하다면) 배선 영역을 줄이는 단계(압축, compaction)를 거치게 된다. 만약 얻어진 해가 만족스럽지 못하면, 위의 과정들을 반복한다.

본 연구에서는 위의 부문제 중에서 세부 배선 문제를 다룬다. 세부 배선 문제는 어떤 그래프 (배선 그래프, routing graph)위에서 어떤 연결 방법 (배선 방식, routing model)을 사용하는가에 의해서 정의된다. 이에 대해서는 다음 절에서 자세히 다룬다. 여러 가지 세부 문제 중 관심 있는 문제는 switchbox 배선 그래프 위에서 knock-knee 배선 방식으로 네트들을 연결하는 문제이다. 이 문제에 대해 자세히 설명하기 전에 그래프 이론에서 잘 알려진 스타이너 나무 (Steiner tree)를 소개하겠다. 어떤 그래프 $G=(V,E)$와 하나의 노드 집합 $N \subseteq V$ 이 주어졌을 때, 나무의 성질을 가지면서 집합 N의 모든 노드들을 연결하는 부분 그래프(subgraph)를 스타이너 나무라고 한다. 본 연구에서 고려하는 문제는 배선 그래프, $G=(V,E)$와 노드 집합의 리스트, $N=\{N_1, \ldots, N_n\}$가 주어졌을 때, n 개의 집합들을 각각 독립적으로 연결하는 n개의 스타이너 나무들을 구하는 문제이다. 그래프에서 하나의 호는 하나의 스타이너 나무에만 포함될 수 있다. 이러한 성질을 갖는 부분 그래프를 스타이너 나무 분할(Steiner Tree Packing)이라 부르겠다. 각각의 호에 할당된 가중치가 있을 때 가중치의 합을 최소로 하는 스타이너 나무 분할을 찾아야 할 때 이 문제를 스타이너 나무 분할 문제(Steiner Tree Packing Problem, STP)라고 부르겠다. 많은 세부 배선 문제는 위에서 정의한 나무 분할 문제나 그와 유사한 문제로 모형화할 수 있다. 스타이너 나무 분할 문제는 *NP-hard* 임이 알려져 있고 [8, 17, 18], 본 논문에서 다루는 switchbox routing graph에서 knock-knee 방식으로 연결하는 문제도 또한 *NP-hard*이다[21]. 이 문제에 대해서는 많은 발견적 기법(heuristic)들[4, 5, 15]이 개발되어있다. 이 발견적 기법들은 상당히 짧은 시간 안에 해를 찾을 수 있지만, 최적을 보장할 수 없을 뿐만 아니라, 최적해를 알 수 없기 때문에 알고리듬의 성능을 알 수 없는 단점이 있다. Grötschel et al. [10]은 이 문제를 정수계획 문제로 모형화하여 절단평면(cutting plane) 기법을 이용하여 풀었다. 하지만, 대형 문제에 대해서는 시간이 너무 많이 걸리는 단점이 있다. 우리는 정수계획 문제를 해결하는 또 다른 하나의 방법인 열 생성(column generation) 기법을 이용한 알고리듬을 개발하여 기존의 절단평면 알고리듬과 비교해 보았다.

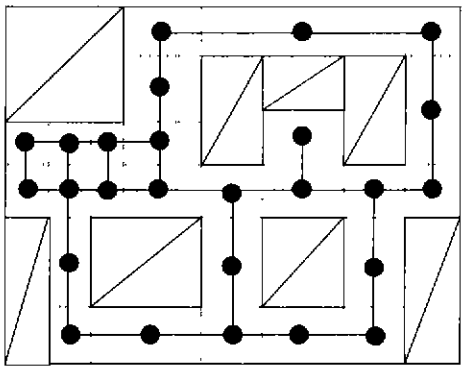
2. VLSI 설계의 배선 문제

배선 문제는 셀의 집합과 연결되어야 하는 터미널들의 집합인 네트들의 리스트, 그리고 배선 영역 위에서 셀들의 위치가 주어졌을 때, 주어진 몇 개의 제약들을 만족시키면서 주어진 목적 함수를 최소화 시키는 네트들의 연결 경로를 찾는 문제라고 할 수 있다. 제약들은 설계자에 의해서 직접 주어질 수도 있고, 구현 과정, 배치 전략, 그리고 설계 스타일에 의해서 발생할 수도 있다. 예를 들어, 다른 네트를 연결하는 인접한 전선들간의 최소 간격, 배선 전선의 최소 너비, 가용한 배선 층(layer)의 수, 시간 제약 등이 제약이 될 수 있다. 목적식의 예로는 전체 사용한 전선의 길이, 배선 영역의 넓이, 상호연결로 인한 시간 지연 등을 들 수 있다.

앞에서 언급했듯이, 배선 문제는 일반적으로 셀의 대략적인 연결 경로를 결정하는 전체 배선 문제와 전선들의 정확한 경로를 결정하는 세부 배선 문제로 나누어 해결하게 된다. 배선 문제를 위와 같이 두개의 부문제로 나누어 푸는 이유는 다음과 같다. 하나는 배선 문제가 한 번에 풀기 어렵기 때문에 좀 더 다루기 쉬운 부문제로 나누어 해결하는 것이다. 또 다른 이유는 설계 및 구현 기술에 따라서 몇 개의 다른 세부 배선 모형(detailed routing model)들이 존재하고, 이 세부 배선 모형은 필요한 배선 영역의 크기에 영향을 주는 데 반해, 전체 배선에는 별 영향을 주지 않는다는 것이다. 배선 문제에 관한 좀 더 자세한 설명은 [3, 13, 20]을 참조하기 바란다.

2.1 전체배선

전체 배선에서는 배선 영역을 부분 영역 (sub-area)으로 나누어 다음과 같이 그래프로 표현한다. 노드는 부분 영역을 나타내고, 서로 인접한 두 부분 영역들을 나타내는 노드들 사이에 호를 넣는다 ([그림 1] 참조). 이렇게 얻어진 그래프를 $G=(V, E)$라 하자. 한 호 $uv \in E$에 대해서, 호의 용량, c_{uv}를 두 노드 u, v에 해당하는 두 부분 영역들 사이를 지나갈 수 있는 네트들의 수로 정의하고, 호의 가중치, w_{uv}를 그 두 영역의 중심점들간의 거리로 정의한다. 네트들의 각 터미널은 터미널을 포함하거나 터미널의 위치에 가장 가까운 부분 영역에 해당하는 노드에 할당 된다. 이와 같이 그래프를 구성하고 나면, 전체 배선 문제는 호의 용량 제약을 만족시키면서 각 네트에 대한 스타이너 나무를 구함으로써 해결할 수 있다.



[그림 1] 전체 배선 그래프

2.2 세부 배선

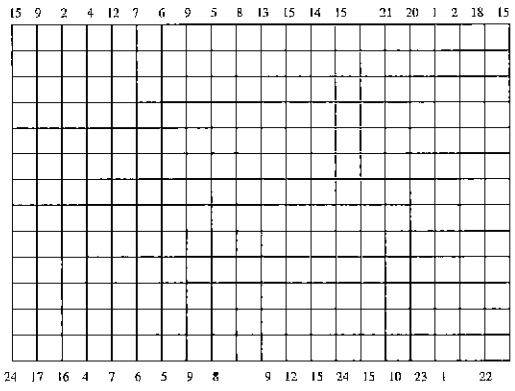
세부 배선 문제는 전체배선에서 결정된 경로(전체 경로)에 부합되는 세부 배선 그래프상에서 몇 개의 제약들을 만족시키는 세부 경로를 찾는 문제이다. 제약은 바로 세부 배선 모형(detailed routing model)이다. 이 문제는 대부분 격자 그래프로 모형화 되므로, 여기서는 이러한 경우만을 고려하기로 한다. 세부 배선 문제는 어떤 그래프상에서 어떤 배선 모형을 사용하느냐에 의해서 결정된다. 따라서, 이 두 가지 기준으로 세부 배선 문제를 분류할 수 있다. 이 분류 방법에 대해서 자세히 설명하기로 하자.

2.2.1 세부 배선 그래프

세부 배선 그래프는 배선 영역의 모양과 터미널

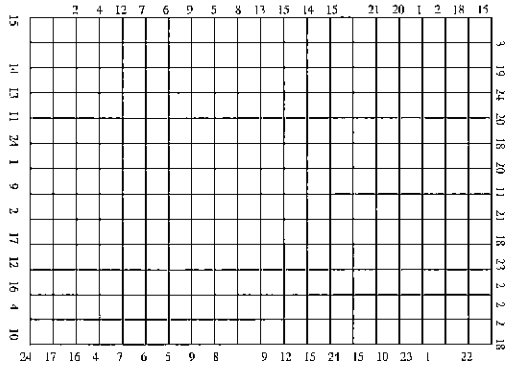
들의 위치에 의해 결정된다. 부분 영역 (세부 배선 영역)의 모양은 전체 배선에서 배선 영역을 나누는 방법에 의해 결정되어지는데, 보통 사각형, L자 모양, T자 모양 등을 갖는다. 전체 배선 문제를 해결하고 나면, 어떤 네트들이 어떤 부분 영역을 지나갈 것인가를 알 수 있다. 두 개의 인접한 부분 영역의 경계를 생각해 보자. 불행히도 전체 배선의 정보만으로는 네트들이 경계선의 어떤 점을 지나갈 것인가를 알 수 없다. 세부 배선을 하기 위해서는 이런 점들 (모조 터미널, pseudo-terminal)의 위치를 알아야 한다. 이런 모조 터미널들의 위치는 발견적 기법으로 결정하게 된다 [10]. 세부 배선 그래프는 다음과 같이 크게 3가지로 분류한다.

(a) *Channel Routing* : 완전 격자 사각형 그래프 위에 모든 터미널들이 그래프의 상단과 하단에만 존재한다 ([그림 2] 참조). [그림 2]에서 숫자들은 네트의 번호를 나타낸다. Channel의 높이, 즉 수평 트랙의 수가 정해져 있지 않기 때문에, 이 문제에서는 그 높이를 최소화하는 것이 목적이다.



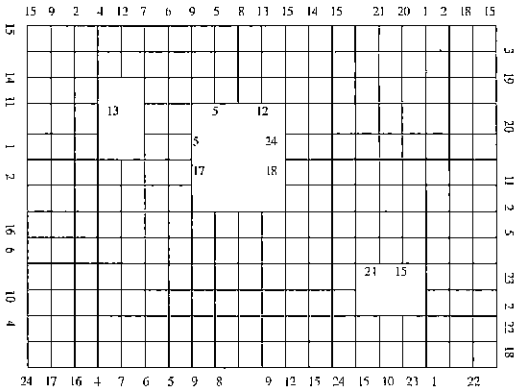
[그림 2] Channel routing graph [10]

(b) *Switchbox Routing* : 터미널들이 그래프의 네 개의 바깥면에 존재할 수 있다는 것이 channel routing과 다른 점이다 ([그림 3] 참조). Channel routing과 달리 switchbox routing에서는 모양이 결정되어져 있기 때문에 모양을 최적화할 수 없다. 따라서, 여기서는 전체 사용하는 전선의 길이를 최소화하게 된다. 이렇게 함으로써, 배선 다음 단계인 compaction에서 전체 배선 영역의 넓이를 줄일 수 있는 가능성을 높일 수 있다.



[그림 3] Switchbox routing graph[10]

(c) *General Routing* : 임의의 격자 그래프에 터미널들이 바깥면에만 존재하지 않고 내부에도 존재할 수 있다 ([그림 4] 참조).



[그림 4] General routing graph [10]

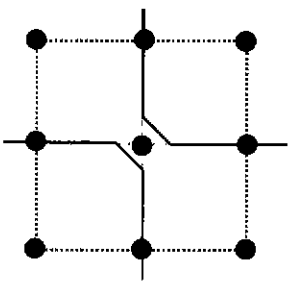
2.2.2 세부 배선 모형

4가지 기본적인 세부 배선 모형들을 소개하겠다. 복수계층 (multiplayer) 배선 모형을 제외한 나머지 모형들은 주어진 그래프가 평면 격자 그래프

(planar grid graph)라고 가정한다.

(a) *Planar routing model* : 이 모형에서는 서로 다른 네트의 연결 경로는 같은 노드를 지날 수 없다 (node-disjoint). 이런 가정을 만족시키는 배선은 하나의 층만을 사용하여 구현할 수 있지만, 전선이 서로 교차할 수 없기 때문에 해가 존재할 가능성이 매우 적다. 따라서, 실제적으로 평면 배선 모형으로 모형화 할 수 있는 배선 문제는 거의 없다.

(b) *Knock-knee routing model* : 이 모형은 edge-disjoint한 네트들의 연결 경로를 가정한다. 배선이 node-disjoint할 필요가 없기 때문에 그림 5와 같은 knock-knee가 가능하다. 따라서, 이 모형은 배선 가능성은 높지만, 층 할당을 고려하지 않았다는 단점이 있다. 하지만, Brady and Brown [2]이 knock-knee 모형의 배선을 4개 이하의 층에 할당할 수 있는 알고리듬을 개발하였다. 그리고, Lipski [14]가 knock-knee 배선이 3개의 층만을 사용할 수 있는가를 결정하는 문제가 *NP-complete*임을 증명하였다.



[그림 5] knock-knee routing

(c) *Manhattan routing model* : 이 모형에서는 edge-disjoint하게 연결하되 그림 5의 knock-knee를 허용하지 않는다. 이러한 제약으로 인해 2개의 층에 할당이 가능하다.

(d) *Multilayer routing model* : 이 모형은 평면 배선 모형을 일반화 시킨 것으로, k-차원 격자 그래프상에서 node-disjoint한 연결을 찾는다 (단, k는 층의 수). 이 모형은 PCB(Princted-Circuit-Board) 배선과 같은 형태의 문제의 실제성을 잘 반영하는 장점이 있지만, 그래프가 너무 커서 다루기가 어렵다는 단점이 있다.

세부 배선 문제는 지금까지 설명한 두 가지 기준에 의해서 정의될 수 있다. 예를 들어 첫번째 기준에서 switchbox 배선 그래프를 두 번째 기준에서 knock-knee 배선 모형을 선택하면 모든 터미널들이 바깥면에만 존재하는 완전 격자 그래프상에서 edge-disjoint한 스타이너 나무들을 찾는 문제가 된다. 바로 이 문제가 본 연구에서 고려하는 문제이다.

3. 문제 정의 및 정식화

이 장에서는 스타이너 나무 분할 문제에 대해 자세히 설명하고 2개의 가능한 정수 계획 모형을 제시한 후 이 모형들을 비교해 보겠다.

먼저 스타이너 나무 분할 문제는 다음과 같이 말할 수 있다.

스타이너 나무 분할 문제
(Steiner Tree Packing Problem, STP)

그래프 $G=(V,E)$와 노드 집합들의 리스트 $N=\{N_1, \ldots, N_n\}$가 주어졌을 때, 다음과 같은 3가지 성질을 갖는 호의 집합 $S_1, \ldots, S_n \subseteq E$를 찾는다.

(i) $(V(S_k), S_k)$는 노드집합 N_k를 연결하는 스타이너 나무이다.

(ii) $\sum_{k=1}^{n} |S_k \cap \{e\}| \leq 1$ for all $e \in E$

(iii) $\sum_{k=1}^{n} |S_k|$ 이 최소값을 가진다.

단, $V(S_k)$는 S_k에 속하는 호와 인접한 노드들의 집합이다.

Grötschel et al. [10]은 아래와 같은 정수 계획 모형을 제시하였다.

(STP1)

$$\min \quad \sum_{k=1}^{n} \sum_{e \in E} x_e^k$$

s.t

$$\sum_{e \in \delta(W)} x_e^k \geq 1, \text{ for all } W \subset V,\ W \cap N_k \neq \emptyset,$$

$$(V \setminus W) \cap N_k \neq \emptyset,\ k = 1, \ldots, n, \quad (1)$$

$$\sum_{k=1}^{n} x_e^k \leq 1, \text{ for all } e \in E, \quad (2)$$

$$x_e^k \in \{0,1\}, \text{ for all } e \in E \text{ and } k = 1, \ldots, n, \quad (3)$$

단, $\delta(W)$는 노드 집합 W에 의한 cut이다.

결정변수 x_e^k는 호 e가 네트 k의 연결에 사용되면 1, 그렇지 않으면 0인 이진 변수이다. 제약식 (1)은 Steiner cut inequality라고 부르며, 스타이너 나무를 나타내기 위한 것이다. 제약식 (2)는 하나의 호는 하나의 스타이너 나무에만 포함될 수 있다는 것을 의미한다. Grötschel et al. [10]은 Steiner partition inequality, alternating cycle inequality와 같은 절단 평면을 이용한 알고리듬을 제시하였다.

본 연구에서 고려하는 문제는 배선 그래프가 격자 그래프이고, 터미널 노드가 모두 그래프의 바깥면에 존재한다(switchbox routing). 일반적으로 스타이너 나무 문제는 *NP-hard*로 알려져 있다 [8, 17]. 그래프가 평면적(planar)이고 모든 터미널 노드가 그래프의 바깥변에 존재하는 경우에 대해서 Erickson et al. [7]이 Dreyfus and Wagner [6]가 제시한 동적계획법을 발전시킨 다항 시간 알고리듬을 제시하였다. 이 알고리듬의 시간 복잡도는 $O(vt^3 + (v \log v)t^2)$이고, t는 네트의 터미널의 수이고 v는 그래프의 전체 노드 수이다. 이러한 사실을 볼 때, 열 생성기법을 이용하여 다음과 같은 정수 계획 모형을 푸는 것이 타당하다.

(STP2)

$$\min \quad \sum_{k=1}^{n} \sum_{i \in T_k} d_i y_i$$

s.t.

$$\sum_{i \in T_k} y_i = 1, \text{ for all } k = 1, \ldots, n \quad (4)$$

$$\sum_{k=1}^{n} \sum_{i \in T_k(e)} y_i \leq 1, \text{for all } e \in E, \quad (5)$$

$$y_i \in \{0,1\}, \text{for all } i \in T_k \text{ and } k = 1, \ldots, n, \quad (6)$$

단, T_k는 네트 k의 모든 가능한 스타이너 나무들의 집합이고, $T_k(e)$는 T_k에 속하는 스타이너 나무들 중에서 호 e를 사용하는 것들이며. $T = \cup_{k=1}^{n} T_k$이고, $d_i = |S_i|$이다.

결정 변수, y_i는 스타이너 나무 i가 연결에 사용되면 1이고, 그렇지 않으면 0인 이진 변수이다. 제약식 (4)는 각 네트는 하나의 스타이너 나무로 연결되어야 한다는 것을 의미하고, 제약식 (5)는 제약식 (2)와 같은 의미를 갖는다. 위의 모형은 변수의 수가 너무 많다는 단점이 있다. 이러한 형태의 문제에 가장 적합한 방법은 열 생성 기법이다. STP2의 열 생성 문제는 각 네트에 대해서 갱신되는 호의 가중치에 대해서 최소가중치를 갖는 스타이너 나무를 구하는 문제가 된다. 열 생성기법에 대해서는 4.3절에서 자세히 설명하기로 하자. 일반적으로 변수의 수가 너무 많다는 단점에도 불구하고, 이 모형을 사용하는 이유는 STP2의 선형완화 문제가 최적값에 대한 좋은 하한을 주기 때문이다. 정수계획 모형의 선형완화 문제는 정수 제약을 비음 제약으로 대체한 것으로, 선형완화 문제의 최적값은 정수계획 문제의 최적값에 대한 하한이 된다. 위의 두 정수계획 문제의 선형완화 문제를 각각 LSTP1, LSTP2라고 하자. 네트 k의 스타이너 나무를 나타내는 이진 벡터들의 집합을 ST_k라 하고, ST_k의 볼록 집합(convex hull)을 $conv(ST_k)$라 하자. 그리고, 아래와 같은 선형계획문제를 LP1이라 하자.

(LP1)

$$\min \sum_{k=1}^{n} \sum_{e \in E} x_e^k$$

$$\text{s.t.} \sum_{k=1}^{n} x_e^k \le 1, \text{ for all } e \in E \text{ and}$$

$$x^k \in conv(ST_k), \text{ for all } k = 1, \ldots, n.$$

정리1 STP2의 선형완화 문제 LSTP2와 LP1은 동일한 최적값을 갖는다.

(증명) LP1의 실행 가능한 하나의 해를 $\bar{x} = (\bar{x}^1, \bar{x}^2, \ldots, \bar{x}^n)$라 하면, 네트 k에 대한 $|E|$-차원 벡터인 $\bar{x}^k$는 아래와 같이 $conv(ST_k)$의 정점들(extreme points)의 볼록 결합(convex combination)으로 표현 가능하다.

$$\bar{x}^k = \sum_{i=1}^{m_k} \bar{\lambda}_k^i q_k^i, \text{for all } k = 1, \ldots, n.$$

단, $\bar{\lambda}_k^i \ge 0$, $\sum_{i=1}^{m_k} \bar{\lambda}_k^i = 1$이고 $q_k^1, \ldots, q_k^{m_k}$는 $conv(ST_k)$의 정점들이다. $conv(ST_k)$의 하나의 정점, q_k^i는 네트 k의 하나의 스타이너 나무를 나타내는데, 이 스타이너 나무에 해당하는 LSTP2의 변수를 y_{k_i}라 하자. 이러한 변수 y_{k_i}의 값을 위의 볼록 결합에서 정점 q_k^i의 계수, $\bar{\lambda}_k^i$라고 하고, 다른 y_{k_i}들의 값을 0이라고 하면, LSTP2의 실행 가능한 해가 된다. LP1의 임의의 실행 가능한 해로부터 LSTP2의 실행 가능한 해를 얻을 수 있으므로, LSTP2의 최적값은 LP1의 최적값보다 작거나 같다.

위와 같은 방법으로 LSTP2의 임의의 실행 가능한 해로부터 LP1의 실행 가능한 해를 얻을 수 있으므로, LP1의 최적값은 LSTP2의 최적값보다 작거나 같다. 따라서, LP1과 LSTP2는 동일한 최적값을 갖는다. □

Z_{LP1}, Z_{L1}, Z_{L2}를 각각 LP1, LSTP1, LSTP2의 최적값이라고 하자. LP1의 모든 실행 가능한 해는 LSTP1의 실행 가능한 해인데, LSTP1의 실행 가능한 해 x는 모든 k에 대해 x^k가 T_k의 스타이너 나무 특성벡터의 볼록조합 다면체(convex hull)에 속하는 경우만 LP1의 해가 되므로, $Z_{L1} \le Z_{LP1}$이다. 그리고, $Z_{LP1} = Z_{L2}$이므로, $Z_{L1} \le Z_{L2} = Z_{LP1}$이다. 그러므로, LSTP2는 적어도 LSTP1보다 크거나 같은 하한을 준다. 따라서, 분지한계법(branch-and-bound)으로 최적해를 구하는 경우에 선형완화 문제가 주는 하한을 고려해 볼 때 STP2가 STP1보다 좀 더 유리하다고 할 수 있다.

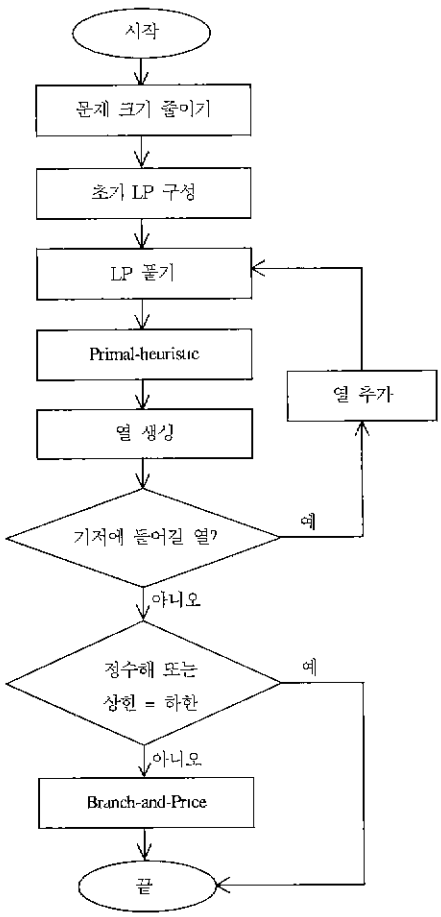
4. 알고리듬

4.1 전체 알고리듬

이 장에서는 우리가 사용한 알고리듬에 대해 설명하겠다. 본 논문에서 제시한 모형은 매우 많은 변수를 가지고 있으므로, 열 생성 기법과 분지한계법을 결합한 분지평가법 (branch-and-price)을 적용하였다. 이 기법은 분지한계법을 수행하면서 변수 (열)를 추가하는 방법으로 그 동안 큰 크기의 혼합 정수계획문제에 성공적으로 적용되어져 왔다 [12, 22]. 일부분의 변수만을 포함한 선형완화문제에서 시작하므로, 현재의 해가 최적이 아닐 수 있다. 따라서, 현재의 해가 최적인가를 확인하기 위해 기저 (basis)에 들어올 변수 (열)를 찾기 위해 열 생성 문제 또는 평가 문제 (pricing problem)라고 부르는 부문제 (subproblem)를 푼다. 만약 기저에 들어올 변수가 발견되면 이 변수를 선형완화문제에 추가한 후에 다시 푼다. 이 과정은 기저에 들어올 변수가 더 이상 없을 때까지 반복되는데, 이 때의 해가 선형완화문제의 최적해가 된다. 만약 최적해가 정수가 아니면, 분지 (branching)를 하게 된다. 분지 나무 전체에 걸쳐 유사한 방법으로 새로운 변수들이 추가 된다. 이 기법에 대한 좀 더 자세한 설명은 [1]을 참조하기 바란다. 이 기법은 비정수해를 제외시키는 제약식을 찾아 추가하는 절단평면기법과도 같이 사용할 수도 있다. 그 예로

Park et al. [19]는 이러한 방법으로 bandwidth packing 문제를 매우 효과적으로 해결하였다.

이제 좀 더 자세히 본 논문의 알고리듬을 설명하겠다. 초기 LP를 구성하기 전에 critical cut [10]을 이용하여 문제의 크기를 줄인다. 그 다음에, 기존 연구의 발견적 기법 해[4, 5, 15]를 이용하여 초기 LP를 구성하고, STP2의 선형완화문제 (LSTP2)를 열 생성 기법을 이용하여 최적까지 푼다. 열 생성 기법을 한번 실행할 때마다 현재 LP의 해를 이용하여 상한을 개선하는 primal-heuristic을 실행한다. LSTP2의 최적해를 구했을 때, primal-heuristic으로 구한 상한과 LSTP2가 주는 하한이



[그림 6] 전체 알고리듬의 순서도

같거나 LP의 해가 정수해이면 멈추고, 그렇지 않으면 분지평가법을 적용하여 STP2의 최적해를 구하게 된다. 분지평가법은 최적을 보장하기 위해 분지한 후에도 열을 생성한다는 점을 제외하면 분지한계법과 다를 바가 없다. 한 노드를 다 풀고 난후 다음 풀 노드를 선택하는 기준으로는 가장 낮은 하한을 주는 노드를 선택하는 최적한계규칙 (best bound rule) [13]을 이용하였다. 전체 알고리듬은 [그림 6]에 있다.

4.2 문제 크기 줄이기

Grötschel et al. [10]은 critical cut을 이용하여 어느 네트의 연결에 사용될 수 없는 호를 찾아내었다. 이와 동일한 방법으로 각 네트의 그래프에서 이러한 호를 제거할 수 있다. 이렇게 하면 가능한 스타이너 나무의 수가 줄어들게 되므로, 문제의 크기를 줄일 수 있다. Critical cut에 대한 좀 더 자세한 설명은 [9, 10]을 참조하기 바란다.

4.3 열 생성 기법

앞 절에서 언급한 바와 같이, 초기 선형완화문제는 모든 변수를 포함하지 않고 일부분의 변수만을 포함한다. 이러한 STP2의 제한된 선형완화문제를 LSTP2′라 하자. LSTP2′의 최적해가 LSTP2에 대해서도 최적인지 조사하기 위해 기저에 들어올 열(변수)를 찾기 위해 열 생성 문제를 풀어야 한다. 주어진 LSTP2′의 최적기저에서 얻어진 제약식 (4), (5)의 쌍대 변수 벡터를 각각 λ, π이라 하자. 그러면, 최적 조건은 다음과 같다.

$$\lambda_k + \sum_{e \in S_i} \pi_e \le d_i, \text{ for all } i \in T_k \text{ and}$$
$$\pi_e \le 0, \text{ for all } e \in E \text{ and } k = 1, \ldots, n. \quad (7)$$

단, S_i는 네트 k에 대한 스타이너 나무 i의 호들의 집합이고 $d_i = |S_i|$이다.

최적 조건 (7)은 $\sum_{e \in S_i} (1 - \pi_e) \ge \lambda_k$, and $\pi_e \le 0$, for all $e \in E$ and $k = 1, \ldots, n$와 동일하

다. 따라서, 열 생성 문제는 다음과 같다.

For all $k=1, K, n$,

(CGP) $\min \sum_{e \in E} (1-\pi_e) z_e$

s.t. $z \in S_k^{|E|}$.

단, $S_k^{|E|}$는 네트 k의 스타이너 나무를 나타내는 이진 벡터들의 집합이다.

다시 말하면, 열 생성 문제(CGP)는 호의 가중치가 $w_e = 1-\pi_e$로 주어진 그래프에서 각 네트 k에 대한 최소 가중치의 스타이너 나무를 찾는 문제가 된다. 구해진 스타이너 나무의 가중치가 λ_k보다 작으면, 이에 해당하는 변수 (열)가 기저에 들어갈 수 있다. 그렇지 않으면, 현재의 해가 최적이다. 스타이너 나무 문제에 대해서는 많은 발견적 기법과 몇 개의 최적 알고리듬이 개발되어져 있다. 스타이너 나무 문제에 대한 자세한 내용은 [11, 23]을 참조하길 바란다. Dreyfus and Wagner [6]는 동적계획법을 이용하여 시간 복잡도가 $O(3^t v + 2^t v \log v + v^2)$인 알고리듬을 개발하였다. Erickson et al. [7]은 연결해야 할 터미널들이 그래프의 바깥면에 있는 경우에 대해서 이 알고리듬을 개선하여 시간 복잡도를 $O(vt^3 + (v \log v)t^2)$으로 줄여 다항 시간 안에 스타이너 나무를 구할 수 있게 하였다. 열 생성 문제의 경우도 모든 터미널이 그래프의 바깥면에 존재하기 때문에, Erickson et al. [7]이 개발한 알고리듬을 이용할 수 있었다. 일반적으로 분지평가법을 적용할 때 어려운 점은 분지 (branching) 후에 열 생성 문제의 구조가 변하는 데에 있다. 따라서, 열 생성 문제의 구조에 심각한 영향을 미치지 않는 분지 방법 (branching rule)을 개발하는 것이 중요하다. 본 연구에서는 열 생성 문제의 구조에 전혀 영향을 주지 않는 분지방법을 개발하여 분지 후에도 아무런 문제없이 열 생성 문제를 다항 시간 안에 해결할 수 있었다. 분지 방법에 대해서는 다음 절에서 자세히 다루기로 한다.

4.4 분지 방법

이분법에 근거한 전통적인 분지 방법은 분지한계법에 부적합하다. 본 논문에서 다루는 문제의 경우에도 마찬가지이다. 변수 y_i의 값이 정수가 아니어서 y_i를 분지하여 0으로 고정한 경우를 생각해 보자. 그러면, 열 생성 문제의 최적해가 y_i에 해당하는 스타이너 나무와 동일할 가능성이 있다. 이러한 경우에는 두번째로 낮은 가중치를 갖는 스타이너 나무를 찾아야 한다. 최악의 경우 분지 나무의 깊이가 n인 노드에서는 n번째로 낮은 가중치를 갖는 스타이너 나무를 찾아야만 할 수도 있다. 따라서 열 생성 문제와 잘 부합할 수 있는 분지 방법이 필요하다. 즉, 분지에 의한 제약에 위배되는 열이 생성되지 않도록 열 생성 문제를 수정해야 한다. 또한, 이러한 수정에 의한 열 생성 문제의 변화가 심각하지 않아야 한다. 이에 대한 자세한 내용은 [22]를 참조하기 바란다.

본 연구에서는 STP에 대해서 두 가지 분지 방법을 생각할 수 있었다. 하나는 호 e가 네트 k의 연결에 사용되는지의 여부를 나타내는 이진 변수 x_e^k를 이용하는 것이다. 변수 x_e^k는 아래의 식을 이용하여 계산할 수 있다.

$$x_e^k = \sum_{t \in T_k(e)} y_t \quad \text{for all } e \in E,\ k=1, K, n.$$

단, 는 호 e를 사용하는 네트 k의 스타이너 나무들의 집합이다.

x_e^k가 이진 변수이므로 y_t들도 모두 이진 변수가 된다는 것을 쉽게 알 수 있다. 따라서, y_t 대신에 x_e^k를 이용하여 분지할 수 있다. 만약 x_e^k가 0으로 분지 되면, 네트 k에 대한 스타이너 나무를 구할 때 그래프에서 호 e를 없앤다. 이 경우에는 여전히 모든 터미널들이 그래프의 바깥면에 있음으로 다항 시간 내에 최적의 스타이너 나무를 구할 수 있다. 그러나, 만약 x_e^k가 1로 분지 되면 호 e의 양 끝 노드를 터미널로 선언하고 호 e의 가중치를

0으로 놓는다. 이렇게 되면 새로 선언된 터미널들이 그래프의 내부에도 존재할 수 있으므로, 다항 시간 내에 최적의 스타이너 나무를 구할 수 없다. 다시 말하면 분지에 의한 제약으로 P에 있던 열생성 문제가 어려워져 *NP-hard*가 된다. 이런 문제점은 한 네트가 반드시 사용해야 하는 호가 존재하기 때문에 발생하므로, 반드시 사용해야 하는 호가 나타나지 않도록 분지방법을 고안함으로써 해결할 수 있다. 두번째 분지방법은 분지 후 각 자식 노드에서 사용하지 못하는 호들을 지정하는 방식이다. 사용하지 않는 호만을 결정하면 단순히 그래프에서 해당하는 호들만을 제거하면 되므로, 여전히 모든 터미널들이 그래프의 바깥면에 존재하게 된다 따라서, 열 생성 문제의 난이도는 변하지 않고 우리는 다항 시간 내에 최적의 스타이너 나무를 구할 수 있다. 주어진 LSTP2′의 최적해 y^*에 대해, 호 e를 사용하는 네트들의 집합을 아래와 같이 정의하자.

$$U(y^*, e) = \{k \mid |N_k(y^*, e)| \geq 1\}, \text{ for all } e \in E.$$

단, $N_k(y^*, e) = \{i \in T_k(e) : y_i^* > 0\}$

y^*가 정수이면, 명확히 $|U(y^*, e)| \leq 1$ for all $e \in E$이다. 그 역은 성립하지 않지만, 아래와 같은 긍정적인 결과를 얻을 수 있다.

정리 2 LSTP2′의 최적해를 y^*라 하고, 그 값을 Z^*라 하자.

만약 $|U(y^*, e)| \leq 1$, for all $e \in E$,이면, y^*는 정수해이거나 동일한 값을 갖는 정수해가 존재한다.

증명) $|U(y^*, e)| \leq 1$ for all $e \in E$이고, y^*가 정수해가 아니라고 가정하자. 양의 값을 갖는 네트 k의 변수들의 집합을 T_k^*라 하자. 즉, $T_k^* = \{i \in T_k : y_i^* > 0\}$.

모든 네트에 대해 $|T_k^*| = 1$이면, 각 네트에 대한 변수 중 양인 것은 하나뿐이므로 y^*는 정수해이다. 그렇지 않으면, T_k^*의 원소 중 하나를 선택하여 1로 고정하고 다른 것은 모두 0으로 고정한다. 이렇게 얻은 해는 $|U(y^*, e)| \leq 1$이므로 두개 이상의 네트가 동일한 호를 사용할 수 없으므로 실행 가능하다. 또한, T_k^*안의 변수들은 목적 함수의 계수가 모두 동일하다 그렇지 않으면, y^*가 최적해라는 가정에 모순이다. 따라서, 최적해 y^*의 값과 동일한 값을 갖는 정수해를 얻을 수 있다. □

위의 정리에 의해 LSTP2′의 최적해 y^*가 정수인지 직접 조사하지 않고 단순히 $|U(y^*, e)| \leq 1$ for all $e \in E$인가를 조사하면 된다.

이제 이 분지 방법에 대해 자세히 설명하겠다. 주어진 LSTP2′의 최적해, y^*에 대해서, 가장 많은 네트가 사용하는 호, e^*를 찾는다($e^* = \arg\max_{e \in E} |U(y^*, e)|$). 만약 두개 이상의 호들이 발견되면 $\sum_{k=1}^{n} x_{e^*}^k$의 값이 가장 큰 호를 선택한다. 즉, 스타이너 나무들에 의해 가장 많이 사용된 변수를 선택한다. 이렇게 선택된 호 e^*에 대해서 구해진 집합 $U(y^*, e^*)$를 $\{k_1, \ldots, k_m\}$라 하자. 만약 $|U(y^*, e^*)| = 1$이면, 정리 2에 의해 분지할 필요가 없다. 그렇지 않으면, 집합 를 크기가 같은 두개의 부분 집합으로 다음과 같이 나눈다.

$$U_1(y^*, e^*) = \{k_1, \ldots, k_{\lfloor m/2 \rfloor}\},$$
$$U_2(y^*, e^*) = U(y^*, e^*) \setminus U_1(y^*, e^*)$$

그리고 나서, 두개의 자식 노드를 만든다. 첫번째 노드에서는 $k \in U_1(y^*, e^*)$에 속한 네트들의 스타이너 나무는 호 e^*를 사용하지 않도록 하고, 두번째 노드에서는 $U_2(y^*, e^*)$에 속한 네트들의 스타이너 나무가 호 e^*를 사용하지 않도록 한다. 즉, 첫번째 노드에서는

$$y_i = 0, \text{ for all } i \in T_k(e^*) \text{ and } k \in U_1(y^*, e^*),$$

이 되도록 하고, 두번째 노드에서는

$y_i = 0$, for all $i \in T_k(e^*)$ and $k \in U_2(y^*, e^*)$.

이 되도록 한다. 위의 두 조건을 만족하도록 하기 위해서, LSTP2′가 이미 포함하고 있는 스타이너 나무 변수 y_i에 대해서, 첫번째 노드에서는 $i \in T_k(e^*)$이고 $k \in U_1(y^*, e^*)$이면 그 변수의 상한을 0으로 고정하고, 두번째 노드에서는 $i \in T_k(e^*)$이고 $k \in U_2(y^*, e^*)$이면 그 변수의 상한을 0으로 고정한다. 이렇게 하면 현재 해 y^*는 위의 두 조건 중 어느 것도 만족하지 않으므로 제외되며 분지 후에도 실행가능한 정수해들은 모두 고려된다. 게다가, 첫번째 노드에서, $k \in U_1(y^*, e^*)$에 속한 네트에 대한 열 생성 문제는 호를 제거한 그래프상에서 해결하면 된다. 두번째 노드에서도 동일한 방식으로 열 생성 문제를 해결하면 된다. 따라서, 분지한 후에도 Erickson et al. [7]의 다항 시간 알고리듬을 적용할 수 있다.

4.4 초기해

열 생성 기법을 사용하기 위해서는 실행 가능한 초기해가 필요하다. 본 연구에서 7절에서 언급될 시험 문제들에 대해서는 발견적 기법들이 개발되어 있고, 그 해가 이미 있다 [4, 5, 15]. 이 발견적 기법으로 얻은 해를 초기해로 사용하였다. 어떤 문제에 대해서는 발견적 기법이 실행 가능한 해를 찾지 못했다. 이런 경우에는 초기 LP를 실행 가능하게 만들기 위해서 다음 절에서 설명할 인공변수를 도입하였다. 실행 가능한 초기해는 분지평가법에서 최적해의 상한으로 사용될 수 있다.

4.5 인공변수

STP2의 선형완화문제, LSTP2가 현재 어떤 스타이너 나무 변수를 포함하고 있는가에 관계없이 항상 실행 가능하게 만들기 위해서 인공변수를 도입하였다. 각 네트 k에 대해서, 목적식에는 충분히 큰 계수 M을, (4)의 k번째 제약식에는 계수 1을 나머지 제약식에는 계수 0을 도입한다. 그러면 제약식 (4)는 다음과 같이 된다.

$$\sum_{i \in T_k} y_i + a_k = 1, \text{for all } k = 1, \ldots, n$$

물론 초기에 인공 변수들만을 가지고 시작할 수 있지만, 이 방법은 앞에서 설명한 방법보다 최적까지 수렴하는 속도가 떨어진다.

4.6 불필요한 열 제거

STP2는 지수적으로 많은 변수를 가지고 있기 때문에 선형완화문제를 효과적으로 풀기 위해서는 불필요한 변수를 제거하는 것이 좋다. 분지 나무의 노드 0에서 LSTP2′를 최적까지 풀고 나면, 다음 정리에 의해 불필요한 변수 (열)를 제거할 수 있다.

정리 3 노드 0에서 STP2의 선형완화문제, LSTP2를 최적까지 풀었을 때, Z_{IN}를 최적해의 상한(incumbent solution value), Z_{LP}를 LSTP2의 최적값이라고 하고, 변수 y_i의 수정 비용을 r_i라고 하자. 변수 y_i가 비기저 변수이고 그 값이 하한에 있고 $r_i > Z_{IN} - Z_{LP}$이면, y_i의 값이 하한에 있는 정수계획문제, STP2의 최적해가 적어도 하나 존재한다.

증명) y_i가 이진 변수이므로 y_i는 STP2의 최적해에서는 0 또는 1이어야 한다. Z^*가 y_i를 1로 고정시켰을 때 LSTP2의 최적값이라고 하면, $Z^* \geq Z_{LP} + r_i$이다. $r_i > Z_{IN} - Z_{LP}$이므로 $Z^* > Z_{IN}$이다. 따라서, $y_i = 1$인 STP2의 최적해는 존재하지 않는다. 그러므로, $y_i = 0$인 STP2의 해가 적어도 하나 존재한다.

LSTP2를 노드 0에서 최적까지 풀고 나면, 정리 4의 조건을 만족하는 변수들은 모두 제거할 수 있다. 많은 변수를 제거하기 위해서는 좋은 상한을 가지고 있어야 한다. 이것이 상한을 개선하는 primal-heuristic이 필요한 하나의 이유이다.

4.6 Primal-heuristic

Primal-heuristic은 상한을 개선하기 위한 목적으로 사용된다. 이 primal-heuristic은 현재 LP의 해가 가지고 있는 정보를 활용하여 실행 가능한 좋은 해를 얻고자 하는 것이다. x_e^k의 값이 크다는 것은 호 e가 네트 k의 연결에 쓰일 가능성이 높다는 것을 의미한다. 이 사실을 이용하여 두 단계로 이루어진 primal-heuristic을 개발하였다

(초기화) Set $S=\emptyset$ and for $k=1, \dots, n$, set $S_k=\emptyset$

[Phase 1] 아래의 과정을 수행한다.

Step 1) 네트들을 $\bar{y}_{t'}^k$ ($=\max_{t\in T_k} \bar{y}_t^k$)을 기준으로 내림차순으로 정리한다. 정리된 순서를 $(1', \dots, n')$라 하자.

Step 2) $\bar{y}_{t'}^{p'} \geq 0.9$인 가장 큰 정수 p'를 찾는다.

Step 3) For $k=1', \dots, p'$, set $S_k=$ {변수 $\bar{y}_{t'}^k$에 해당하는 스타이너 나무의 호들} and $S=S\cup S_k$.

[Phase 2] For $k=(p+1)', \dots, n'$, 아래의 과정을 수행한다.

Step 1) $R_k=N_k$라 놓고, 호의 가중치 $w_e=1-x_e^k$인 새로운 그래프 $G_k=(V, E_k)$를 구성한다. 단, $E_k=E\setminus S$.

Step 2) 만약 $S_k=\emptyset$이면, $Sh_k(s^*, t^*)=\min_{s,t\in N_k} Sh_k(s, t)$인 s^*, t^*을 결정한다. 단, $Sh_k(s, t)$는 그래프 G_k에서 s와 t사이의 최단경로의 길이이다.

만약 $f_k(s^*, t^*)=\infty$이면,

go to TERMINATION.

그렇지 않으면, $S_k=S_k\cup$ {edges in the s^*-t^* path } & $R_k=R_k\setminus\{s^*, t^*\}$라 놓는다.

그렇지 않으면, $Sh_k(s^*, t^*)=\min_{s\in R_k, t\in V(S_k)} Sh_k(s, t)$인 $s^*\in R_k$, $t^*\in V(S_k)$를 결정한다.

만약 $f_k(s^*, t^*)=\infty$이면,

go to TERMINATION.

그렇지 않으면, $S_k=S_k\cup$ {edges in the s^*-t^* path } & $R_k=R_k\setminus\{s^*\}$라 놓는다.

$G_k=(V, E_k)$에서 s^*-t^* path를 이루는 호들을 제거한다.

만약 $R_k\neq\emptyset$이면, repeat Step 2.

그렇지 않으면, $S_k=S_k\cup S_k$라 놓고 go to Step 3.

Step 3) 만약 $k=n'$이면, 실행가능한 해 S를 출력하고 STOP.

그렇지 않으면, $k=(k+1)'$라 놓고 go to Step1.

TERMINATION) "No feasible solution found" 출력하고 STOP

Phase 2에서 스타이너 나무를 찾기 위해 최적 알고리듬을 이용하지 않고, 최단경로문제의 알고리듬을 이용한 간단한 발견적 기법을 사용하였다. 이에는 두 가지 이유가 있다. 하나는 primal-heuristic이 실행가능한 해를 찾을 가능성이 적기 때문이다. 따라서, 가능한 많이 primal-heuristic을 적용해 봐야 한다 본 알고리듬에서는 새로운 열이 추가된 후 선형완화문제를 풀 때마다 primal-heuristic을 적용한다. 따라서, 매우 빨리 primal-heuristic을 수행해야 한다. 다른 하나의 이유는 primal-heuristic은 발견적 기법일 뿐이기 때문이다. 다시 말하면, phase 2에서 최적의 스타이너 나무를 찾는다고 할지라도 실행 가능한 해를 찾는다는 보장은 없기 때문이다. Primal-heuristic이 실행 가능한 해를 찾는가의 여부는 최적 스타이너 나무를 구하느냐에 영향을 받는 것이 아니라 스타이너 나무를 구하는 네트들의 순서에 영향을 받는 것처럼 보인다.

Primal-heuristic으로 얻어진 해의 값이 현재 상

한보다 더 작으면, 현재의 상한을 이 값으로 바꾸고 primal-heuristic 해의 열들을 현재의 선형완화문제에 첨가한다. 물론, 우리가 사용한 primal-heuristic이 좋은 해를 줄 거라는 확실한 보장은 없다. 하지만, 본 알고리듬을 적용해 본 문제들에서는 primal-heuristic으로 좋은 상한을 얻을 수 있었다.

4.7 선형완화문제의 퇴화성 (degeneracy)

열 생성 기법을 LSTP2의 선형완화문제에 적용할 때 새로운 열을 LP에 첨가하고 다시 LP를 풀게 되는데 (reoptimization), 새로이 첨가된 열들은 primal feasibility는 여전히 만족하지만 dual feasibility은 만족하지 않으므로, primal 심플렉스 방법이 dual 심플렉스 방법보다 더 적절하다. 하지만, 사전 시험을 통해서 우리 문제에는 dual 심플렉스 방법이 primal 심플렉스 방법보다 더 나은 것으로 나타났다. STP의 다른 formulation (STP1)도 유사한 문제점을 가지고 있다. STP1의 선형완화문제, LSTP1은 primal과 dual에서 모두 퇴화성 (degeneracy)이 매우 크다 [10]. 퇴화성을 해결하기 위해 Grötschel et al. [10]은 pertubation method를 사용하였다. 하지만, 우리의 경우는 선형완화문제가 primal에서만 퇴화성이 크기 때문에 perturbation method 대신에 dual 심플렉스 방법을 사용하였다. Dual 심플렉스 방법은 매번 LP를 처음부터 다시 풀어야 하기 때문에 dual feasible한 해를 빨리 찾을 수 있어야 한다. 다행히 목적식의 계수가 모두 양수이기 때문에 CPLEX (본 연구에서 사용한 LP solver)가 dual feasible한 해를 쉽게 찾을 수 있었다. 그리고, 사전 실험을 통해서, dual 심플렉스 방법이 primal 심플렉스 방법보다 더 나은 것으로 나타났다.

5. 실험 결과 및 토의

지금까지 설명한 알고리듬을 Grötschel et al. [10]이 실험해 본 switchbox routing 문제들에 적용해 보았다. 문제들에 대한 정보는 <표 1>에 정리되어 있다. 첫번째 열은 문헌에서 사용되는 문제들의 이름이고, 열 2, 3은 각각 사각 격자 그래프의 높이와 너비를 나타낸다. 열 4는 네트들의 수이고, 열 5에서 9까지는 네트들의 분포를 나타낸다. 열 5는 터미널 수가 2개인 네트의 수를, 열 6은 터미널 수가 3개인 네트들의 수를 나타내고, 나머지 열도 이와 유사한 정보를 나타낸다. 마지막 열은 문제를 가져온 참고문헌을 나타낸다. 이 문제들에 대한 좀 더 자세한 설명은 [10]을 참조하기 바란다. 네트들의 분포를 보면 각 문제가 얼마나 어려운가를 대충 짐작할 수 있다. 터미널의 수가 큰 네트가 많아질수록 문제는 어려워진다. Dense problem (Dense, Augmented dense, Modified dense)가 다른 문제들보다 더 어려울 것으로 보인다.

〈표 1〉 시험 문제 [10]

이 름	H	B	N	네트의 분포					참고 문헌
				2	3	4	5	6	
Difficult	15	23	24	15	3	4	1	1	[4]
More difficult	15	22	24	15	3	5	0	1	[5]
Terminal intensive	16	23	24	8	7	5	4	0	[15]
Dense	17	15	19	3	11	5	0	0	[15]
Augmented dense	18	16	19	3	11	5	0	0	[15]
Modified dense	17	16	19	3	11	5	0	0	[5]
Pedagogical	16	15	22	14	4	4	0	0	[5]

4.2절에서 설명했듯이 문제를 풀기 전에 문제의 크기를 줄이는데 이에 대한 결과는 <표 2>에 정리되어 있다. 두 번째 열은 전체 x_e^k들의 수를, 세 번째 열에는 0으로 고정된 즉 제거할 수 있는 호의 수를 나타낸다. 열 4와 5는 각각 문제 크기 줄이기 과정을 하지 않고 노드 0까지 풀었을 때 발생된 열의 수와 그 과정을 거쳤을 때의 열의 수를 나타낸다. 결과를 보면 문제 크기 줄이기 과정이 가능한 스타이너 나무 수를 줄임에도 불구하고 실제적으로 발생되는 열의 수를 줄이는 데는 별 영향을 주지 못함을 알 수 있다.

〈표 2〉 문제 크기 줄이기 결과 (노드 0)

문 제	전체 호	제거된 호	생성된 열	
			호 제거 안함	호 제거
Difficult	15648	2048	696	935
More difficult	14952	2176	745	816
Terminal intensive	16728	4599	1637	1836
Dense	9082	4593	5668	3819
Augmented dense	10298	2428	1997	2382
Modified dense	9709	3789	3945	3089
Pedagogical	9878	1883	365	295

〈표 3〉 불필요한 열 제거한 결과

문 제	최적 상한	LP value (하한)	불필요한 열
Difficult	464	464	*
More difficult	452	452	*
Terminal intensive	538	535	689(37.53%)
Dense	None	438	0(0.00%)
Augmented dense	469	467	613(34.65%)
Modified dense	452	452	*
Pedagogical	336	331	5(1.70%)

*는 노드 0에서 최적해를 얻었기 때문에 불필요한 열을 제거할 필요가 없는 경우를 말함.

선형완화문제를 노드 0에서 최적까지 푼 후에 불필요한 열을 제거하게 된다. <표 3>은 이에 대한 결과를 정리한 것이다. 열 2와 3은 각각 primal-heuristic에 의해서 발견된 상한과 LSTP2의 목적식값(하한)을 나타낸다. 마지막 열은 제거된 불필요한 열의 수를 나타낸다. (*)는 상한과 하한이 동일하여 불필요한 열을 제거할 필요가 없었던 경우를 나타낸다. 상한과 하한이 가까울수록 더 많은 열들이 제거됨을 알 수 있다.

<표 4>와 <표 5>에는 각각 분지평가법과 분지절단법 (branch-and-cut)의 적용결과가 제시되어 있다. 열 2와 3은 분지 나무의 노드 0에서 각각의 알고리듬이 주는 상한과 하한을 나타내고, 열 4는 상한과 하한의 상대적인 차이, Gap(=100(상한-하한)/하한)를 나타낸다. 열 5는 <표 4>에서는 열 생성 기법의 반복횟수를 <표 5>에서는 절단평면기법의 반복횟수를 나타낸다. 열 6에는 전체 변수의 수가 제시되어 있다. 열 7은 분지 나무의 노드 수를 나타내고, 마지막 열에는 알고리듬의 수행 시간

〈표 4〉 분지평가법 적용 결과 (HP 9000 715/50)

문 제	상한	하한	Gap(%)	반복수	전체열수	노드수	CPU-time(m:s)
Difficult	464	464	0.0	68	935	1	32:26
More difficult	452	452	0.0	58	816	1	25:20
Terminal intensive	536	535	0.19	241	2324	15	222:35
Dense	441	438	0.68	1042	10000*	10	1604:01
Augmented dense	469	467	0.43	1730	10000*	70	1670:25
Modified dense	452	452	0.0	201	3089	1	131:49
Pedagogical	331	331	0.0	124	493	54	22:37

*최적해를 찾지 못한 문제

〈표 5〉 분지절단법 적용 결과 (Grotschel et al. [10], SUN 4/50)

문 제	상한	하한	Gap(%)	반복수	전체변수	노드수	CPU-time(m:s)
Difficult	464	464	0.0	69	13424	3	1564:15
More difficult	452	452	0.0	53	12502	1	983:23
Terminal intensive	537	536	0.19	163	11815	13	3755:44
Dense*	441	438	0.68	119	4251	4	1017:43
Augmented dense*	469	467	0.43	105	7620	1	4561:41
Modified dense	452	452	0.0	51	5652	1	387:03
Pedagogical	331	331	0.0	77	7839	5	251:58

*최적해를 찾지 못한 문제

이 제시되어 있다.

LSTP2는 예상했던 대로 매우 좋은 하한을 준다(가장 큰 Gap이 0.68%). Terminal intensive problem을 제외한 다른 문제들에 LSTP2'의 하한은 Grötschel et al. [10]이 얻은 하한과 동일하다. Grötschel et al. [10]은 Steiner partition inequality, alternating cycle inequality 등의 절단평면을 사용하였다. Steiner partition inequality는 스타이너 나무의 볼록집합의 절단평면이다. 이에 관련된 separation problem은 *NP-hard*이기 때문에, Grötschel et al.[10]은 발견적 기법을 사용하였다. Alternat-ing cycle inequality는 두개의 네트로 이루어진 스타이너 나무 분할의 볼록 집합의 절단평면이고, 이에 관련된 separation problem도 마찬가지로 발견적 기법을 이용하였다. Critical cut은 변수를 0으로 고정시키는데 사용하였다. 이제 두 가지 사실에 주목하자; 하나는 LSTP2가 스타이너 나무의 볼록 집합을 완전히 묘사한다는 것이고, 다른 하나는 terminal intensive problem을 제외한 경우에 대해서 우리의 알고리듬으로 Grötschel et al [10]과 동일한 하한을 얻었다는 것이다. 이러한 사실에 근거하여 선형완화문제에서 얻은 하한의 관점에서 몇 가지를 논의할 수 있다. 첫째로, 스타이너 나무의 볼록 집합은 Steiner cut inequality만으로는 완전히 묘사할 수 없다는 점을 감안해 볼 때 Grötschel et al. [10]이 사용한 절단평면은 하한을 얻는 데 큰 역할을 했다. 둘째로, Steiner partition inequality의 separation problem을 최적까지 풀지는 않았어도 스타이너 나무의 볼록 집합을 잘 묘사한다면, alternating cycle inequality는 하한을 얻는데 큰 역할을 하지 못했을 것이다.

우리는 모든 문제에 대해서 Grötschel et al. [10]과 동일한 값의 해를 얻었다. 특히, "Terminal intensive switchbox"에 대해서는 Grötschel et al. [10]이 537이 최적이라고 발표했으나, 우리 알고리듬을 적용해 본 결과 536이 최적임이 밝혀졌다. 이에 대해서는 [9, 10]의 저자 중의 하나인 Dr. Martin과 토론을 거쳐 그가 다시 이 문제를 풀어본 결과 우리와 동일한 값을 얻었다. 하지만, 메모리의 한계로 인해 최대 변수의 수를 10,000개로 한정하였기 때문에, dense와 augmented dense problem에 대해서는 최적해를 얻지 못했다. Grötschel et al.[10]도 또한 이 문제에 대해서는 최적해를 얻지 못했다. 우리가 사용한 컴퓨터는 HP 9000 715/50이고 Grötschel et al.[10]이 사용한 컴퓨터는 SUN 4/50이다. 각 컴퓨터에 대한 정보는 <표 6>과 같다. <표 6>의 정보만으로 두 컴퓨터의 계산속도를 논하는 것은 무리지만, HP 9000 715/50이 3배 빠르다고 가정하더라도 최적해를 구하지 못한 두 문제의 경우를 제외하고는 우리의 알고리듬이 Grötschel et al.[10]이 제시한 알고리듬에 비해 결코 나쁘지 않다.

〈표 6〉 두 컴퓨터에 대한 정보

	CPU clock(MHz)	MIPS	MFLOPS
HP 9000 715/50	50	62	13
SUN 4/50	40	28.5	4.2

MIPS : Million Instructions Per Second
MFLOPS : Mega FLoating point Operations Per Second

6. 결 론

본 논문에서는 스타이너 나무 분할 문제(Steiner Tree Packing Problem, STP)를 해결하기 위한 열 생성 기법을 제안하고, 이 기법을 이용하여 어떻게 지수적으로 많은 변수를 가지고 있는 선형완화문제를 해결할 것인가를 설명하였다. 또한, 분지 후에도 열 생성 문제가 다루기 쉽도록 하는 분지 방법을 개발하였다.

계산 결과를 통해 Grötschel et al.[10]이 제안한 절단평면 기법에 비해 본 논문에서 제시한 알고리듬이 결코 뒤지지 않음을 알 수 있었다. 또한, 본 알고리듬이 실제 문제에도 적용가능하리라고 예상된다.

앞에서 언급했듯이 VLSI 디자인에는 많은 종류의 배선 문제가 있다. 본 논문에서 고려한 모형은 그 중의 일부에 지나지 않는다. 앞으로 다른 여러 가지 모형은 좋은 연구 대상이 될 수 있을 것이다.

참 고 문 헌

[1] Barnhart, C., E.L., Johnson, G.L., Nemhauser, M.W.P., Savelsbergh and P.H., Vance, "Branch-and-price : Column generation for solving huge integer programs", *Operations Research*, Vol.46(1998), pp.316-327.

[2] Brady, M.L. and D.J., Brown, *VLSI routing : Four layers suffice In Advances in Computing Research. Vol.2 : VLSI theory*, ed F. P. Preparata. JAI Press, London, 1984, pp. 245-258.

[3] Burstein, M., *Channel routing. In Advances in CAD for VLSI, Vol. 4 : Layout Design and Verification*, ed T. Ohtsuki. North-Holland, Amsterdam, 1986, pp. 133-167.

[4] Burstein, M. and R., Pelavin, "Hierarchical wire routing", *IEEE Transactions on Computer-Aided-Design*, Vol.CAD-2(1983), pp.223-234.

[5] Cohoon, J.P. and P.L., Heck, "BEAVER : A computational-geometry-based tool for switchbox routing", *IEEE Transactions on Computer-Aided-Design*, Vol.CAD-7(1988), pp.684-697.

[6] Dreyfus, S.E. and R.A., Wagner, "The Steiner problem in graphs, *Networks*", Vol.1(1972), pp.195-207.

[7] Erickson, R.E., C.L., Monma and A.F.JR., Veinott, "Send-and-split method for minimum-concave-cost network flows", *Mathematics of Operation Research*, Vol.12(1987), pp.634-665.

[8] Garey, M.R. and D.S., Johnson, "The rectilinear Steiner tree problem is NP-Complete", *SIAM Journal on Applied Mathematics*, Vol.32 (1972), pp.826-834.

[9] Grötschel, M., A., Martin and R., Weismantel, "Packing Steiner trees : polyhedral investigations", *Mathematical Programming*, Vol.72 (1996), pp.101-123.

[10] Grötschel, M., A., Martin and R., Weismantel, "Packing Steiner trees : a cutting plane algorithm and computational results", *Mathematical Programming*, Vol.72(1996), pp.125-145.

[11] Hwang, F.K. and D.S., Richards, "Steiner tree problems", *Networks*, Vol.22(1992), pp.55-89.

[12] Johnson, E.L., A., Mehrotra and G.L., Nemhauser, "Min-cut clustering", *Mathematical Programming*, Vol.62(1993), pp.133-152

[13] Lengauer, T., *Combinatorial Algorithms for Integrated Circuit Layout*, John Wiley & Sons, New York, 1990.

[14] Lipski, W., *On the Structure of Three-Layer Wireable Layouts. In Advances in Computing Research, Vol.2 - VLSI theory*, ed F. P. Preparata. JAI Press, London, 1984, pp.231-244.

[15] Luk, W. K., "A Greedy Switchbox Router", *Integration*, Vol.3(1985), pp.129-149.

[16] Nemhauser, G.L. and L.A., Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, 1988.

[17] Karp, R.M, *Reducibility among Combinatorial Problems. In Complexity of Computer Computations*, eds R.E. Miller and J.W. Thatcher. Plenum Press, New York, 1972, pp.85-103.

[18] Kramer, M.R. and J.V., Leeuwen, *The Complexity of Wire-Routing and Finding Minimum Area Layouts for Arbitrary VLSI Circuits. In Advances in Computing Research, Vol. 2 : VLSI theory*, ed F.P. Preparata. JAI. Press, London, 1984, pp.129-146.

[19] Park K., S., Kang and S., Park, "An integer programming approach to the bandwidth packing problem", *Management Science*, Vol 42 (1996), pp.1277-1291.

[20] Sait, M. and H., Youssef, *VLSI Physical Design Automation*, McGraw-Hill, London, 1995.

[21] Sarrafzadeh, M., "Channel-routing problem in the knock-knee mode is NP-Complete", *IEEE Transactions on Computer-Aided-Design*, Vol. CAD-6(1987), pp.503-506.

[22] Savelsbergh, M.W.P., "A branch-and-price algorithm for the generalized assignment problem", *Operations Research*, Vol.45(1997), pp. 831-841.

[23] Winter, P., "The Steiner problem in networks : A survey", *Networks*, Vol.17(1987), pp.129-167