整数計画を用いたシュタイナー木詰め込み問題の解法とその実験的評価

Integer Programming Approaches for the Steiner Tree Packing Problem and Their Experimental Evaluation

大月仁志 * 森本尚之 † 宮崎修一 ‡ 岡部寿男 ‡
Hitoshi Otsuki Naoyuki Morimoto Shuichi Miyazaki Yasuo Okabe

概要

本研究では、帯域制限のあるネットワークに複数のシュタイナー木を詰め込む問題を考察する。整数計画問題への定式化を用いた解法を提案及び実装し、その性能を実験により評価する。

1 はじめに

VLAN(Virtual LAN) はスイッチなどのネットワーク機器の機能によって、物理的な接続の制約を受けることなく、仮想的なサブネットワークを実現する技術である。通常ならばスイッチ 1 つにつき 1 つのサブネットワークしか構成することができないため、このネットワークを複数のサブネットに更に分割する場合、スイッチを複数台用意する必要がありコストがかかってしまう。VLAN ではスイッチ内部でネットワークの分割を実現することができるため、より柔軟性のあるネットワーク構成が可能となる。

ポート VLAN と呼ばれる技術ではそれぞれのポートに対して、そのポートに接続されている端末がどの VLAN に属するかを指定することができる。そのため、例えばある端末を利用するユーザの所属部署が変わったとしてもケーブルをつなぎ替える必要はないというメリットがある。IEEE 802.1Q は VLAN の実装を標準化し、Ethernet フレームのヘッダに付加され、そのフレームがどの VLAN に所属しているかを示す情報を含んでいる “VLAN タグ” を導入することによって複数のスイッチをまたぐ VLAN を実現した。複数の VLAN で同じネットワークを共有する際に、VLAN の要求する帯域幅の合計が、スイッチとスイッチをつなぐワイヤの帯域幅の制約を超えてしまうと、遅延が発生してしまうため、できるだけそれぞれのワイヤの帯域幅の制約を超えないようにそれぞれの VLAN の設計をしなければならない。しかし、この設計作業は殆どの場合未だに手動で行われているのが現実である。最適な構成を見つけることは、比較的小さいネットワークであれば手動でできるが、大規模なネットワークになると非常に困難である。この問題は、NP 完全であるシュタイナー木詰め込み問題 (Steiner Tree Packing Problem) [16] として定式化できる（定義は 2 節で与える）。

シュタイナー木詰め込み問題は NP 完全であるが、VLAN の設計の自動化を実現するためにできるだけ大規模な問題をより効率的に解く必要がある。本研究では、シュタイナー木詰め込み問題を整数計画で定式化し、既存の IP ソルバを使って解を求めることを試みる。整数計画への定式化には、2 つの方法を使った。1 つ目の定式化は既存のもので、ターミナル集合の全てのカット集合に対して制約式を作らなければならないので制約式の数が指数個になる。それに対して 2 つ目の定式化は、通常のシュタイナー木問題の定式化を参考に本論文で提案したものである。変数の数は倍以上になるものの、制約式を多項式個で抑えることができるため 1 つ目の定式化よりも大規模な問題が解けることが期待される。実験は 6 種類のトポロジ、複数のノード数と要求数の組み合わせに対して定められたメモリ、時間の制約下で解が求められるかを検証した。我々の定式化は、メッシュ型のグラフでノード数が増えても解ける問題が見られたが、それらの問題の多くは解が存在しない問題であることが分かった。そのため、追加実験としてメッシュ型のグラフで解が存在する問題としない問題を解くのにかかる時間の平均を求めたところ、解が存在しない問題の方が平均して 70 倍ほど早いことがわかった。

* 京都大学 情報学研究科 知能情報学専攻
Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University.
otsuki@nlp.ist.i.kyoto-u.ac.jp
† 京都大学 物質－細胞統合システム拠点
Institute for Integrated Cell-Material Sciences, Kyoto University.
nmorimoto@icems.kyoto-u.ac.jp
‡ 京都大学 学術情報メディアセンター
Acedemic Center for Computing and Media Studies, Kyoto University.
{shuichi,okabe}@media.kyoto-u.ac.jp

2 シュタイナー木詰め込み問題

2.1 問題の定義

まずシュタイナー木詰め込み問題を定義するために必要な記号を定義する。V をノードの集合、E を辺の集合とするとき、無向グラフを $G(V,E)$ で定義する。辺集合の部分集合 $E' \subseteq E$ に対して $V(E')$ を E' の各辺を構成するノードの集合とする。また、c_e をグラフ G の辺 e が持つ正の容量とする。シュタイナー木は次のように定義される。

定義 1 (シュタイナー木)
$G(V,E)$ を無向グラフ、$T \subseteq V$ を G のノードの部分集合（ターミナル集合という）とする。部分グラフ $(V(S), S)$ が木であり、$T \subseteq V(S)$ を満たすとき、辺集合 S は T のシュタイナー木という。

問題 1 (シュタイナー木詰め込み問題)

- インスタンス:
  - 正の容量 $c_e (\in \mathbb{N}, e \in E)$ を持った無向グラフ $G(V,E)$
  - N 個のタプル $\{(T_1, bw_1), \cdots, (T_N, bw_N)\}$, $N \geq 1$, $bw_k > 0, T_k \subseteq V, \forall k \in \{1,2,\cdots N\}$
- 問題:
  1. G で S_k が $T_k (k \in \{1,\cdots,N\})$ のシュタイナー木
  2. $\sum_{k=1}^{N} bw_k |S_k \cap \{e\}| \leq c_e, \forall e \in E$

  を満たすような $S_1, \cdots, S_N \subseteq E$ は存在するか.

2.2 関連研究

シュタイナー木詰め込み問題は VLSI 設計やネットワーク上のブロードキャスティングへの応用を持つため、古くから研究されてきた。本問題は Korte ら [16] により $N = 2$、全ての $e \in E$ に対して $c_e = 1$ かつ $bw_1 = bw_2 = 1$ でも NP 完全であることが示された。また、Kaski [14] は上記の制限に加えて $T_1 = T_2$ という制限を加えても NP 完全であることを示した。また、[14] は、全ての $e \in E$ に対して $c_e = 1$、$bw_1 = bw_2 = \ldots = bw_N = 1$、$T_1 = T_2 = \ldots = T_N$、$|T_1| = 7$ でも NP 完全であることを示した。Grötschel らは一連の論文 [10, 11, 7, 9, 8] で、シュタイナー木詰め込み問題を整数計画により定式化し、様々なアルゴリズムを使って VLSI 設計用のベンチマーク例題を解いている。

与えられた頂点集合 S をターミナルとするシュタイナー木をできるだけ多く埋め込む問題については、Menger の定理の拡張の一貫として近似可能性に関する多くの研究がなされている。Jain ら [12] は本問題が APX-困難であることを示し、Cheriyan ら [2] は $|S| = 4$ でも APX-困難であることを示した。Lau [18] は本問題に対する 26-近似アルゴリズムを示した。さらに West と Wu [21] は近似度を 6.5 に、DeVos ら [4] は $5 + \epsilon$ に改良した。また、$|S| = 4$ の場合に 1.5 近似が可能なことが示されている [17]。その他の関連研究は文献 [3, 20] に見られる。

与えられた全てのターミナル点をつなぐシュタイナー木のうち辺の重みの総和が最小になるものを見つける最小シュタイナー木問題は、Karp [13] によって NP 完全性が示された有名な問題である。Garey と Johnson [6] は、最小シュタイナー木問題が格子型グラフの場合でも NP 完全であることを示した。

2.3 整数計画による定式化

本節ではシュタイナー木詰め込み問題の整数計画による定式化を 2 つ紹介するが、その前に定式化に必要となるいくつかの記号を定義しておく。2 つのノード集合 $U, W \subseteq V$ に対して、$[U:W]$ を U, W の点を始点と終点に持つ辺の集合とする。辺の両端が u, v となる辺を uv と表す。有向グラフでは uv は u が始点、v が終点と考えるが、無向グラフでは uv と vu は同じ辺である。$r_k \in T_k$ を k 番目のターミナル集合の任意の 1 つのノードとし、T_k のルートと呼ぶことにする。また、$\overrightarrow{G}$ を無向グラフ G の各辺 uv (容量を c とする) を容量が c の 2 つの有向辺 uv と vu に置き換えることで得られる有向グラフとする。ただし、k 番目のターミナル集合を考える際には r_k に入ってくる辺はなく、出て行く辺しかないものとする。有向グラフにおいて、$V^-(j)$ をノード j に入ってくる有向辺の始点ノードの集合とする。

2.4 IP1

IP1 は文献 [1, 10, 15, 11] を参考にしており、以下に示す通りである。

$$\text{min} \quad \sum_{k=1}^{N} \sum_{e \in E} x_{ke} \tag{1}$$

$$\text{subject to} \quad x(\delta(W)) \geq 1, \forall W \subset V,$$
$$W \cap T_k \neq \phi, (V \setminus W) \cap T_k \neq \phi \tag{2}$$
$$x_{ke} = \{0,1\}, \forall e \in E \tag{3}$$
$$k = \{1,\cdots,N\} \tag{4}$$
$$0 \leq \sum_{k=1}^{N} bw_k x_{ke} \leq c_e, \forall e \in E \tag{5}$$

ここで、x_{ke} は k 番目のシュタイナー木において $e \in E$ が使われるとき 1 となり、そうでない時は 0 となる変数である。目的関数は式 (1) で表されるが、今は判定問題を考えているため、目的関数に意味はない。式 (2) において $\delta(W)$ は $\{x_{ke} \mid e \in [W : V \setminus W]\}$、$x(\delta(W))$ は $\delta(W)$ に含まれる変数 x の値の総和と定義する。つまり、式 (2) は $x(\delta(W))$ はターミナル集合を少なくとも 1 つは含むようにノード集合を 2 つに分けた時、この 2 つの集合のカッ

トセットに対応する辺が少なくとも 1 つは使われなければならないということを示している。式 (5) は各辺の容量をシュタイナー木の帯域幅要求が超えてはいけないという制約である。

2.5 IP2

IP2 は文献 [5] における最小シュタイナー木問題の整数計画への定式化を参考にしている。我々はこの定式化を次に示す通り容量制限付きの複数のシュタイナー木に関する定式化に拡張した。

$$\min \quad C \tag{6}$$

subject to

$$r_k \in T_k \tag{7}$$

$$\sum_{i \in V^-(j)} x(k)_{ij} \leq 1, \forall j \in V(\overrightarrow{G}) - \{r_k\} \tag{8}$$

$$n \sum_{i \in V^-(j)} x(k)_{ij} \geq u(k)_j + 1, \forall j \in V(\overrightarrow{G}) - \{r_k\} \tag{9}$$

$$(n+1) \sum_{i \in V^-(j)} x(k)_{ij} \leq n(u(k)_j + 1),$$
$$\forall j \in V(\overrightarrow{G}) - \{r_k\} \tag{10}$$

$$1 - n(1 - x(k)_{ij}) \leq u(k)_j - u(k)_i \leq 1 + n(1 - x(k)_{ij}),$$
$$\forall ij \in E(\overrightarrow{G}) \tag{11}$$

$$x(k)_{ij} \in \{0, 1\}, \forall ij \in E(\overrightarrow{G}) \tag{12}$$

$$u(k)_{r_k} = 0, u(k)_i \geq 0, \forall i \in T_k - \{r_k\} \tag{13}$$

$$k \in \{1, \cdots, N\} \tag{14}$$

$$0 \leq \sum_{k=1}^{T} bw_k (x(k)_{ij} + x(k)_{ji}) \leq c_{ij}, \forall ij \in E(\overrightarrow{G}), i < j \tag{15}$$

ここで、$x(k)_{ij}$ は 1 の時、有向辺 ij が k 番目のシュタイナー木で使われ、0 の時使われないことを示す変数である。また、$u(k)_j$ は k 番目のシュタイナー木において r_k からノード j までのホップ数を示しており -1 の時はノード j がそのシュタイナー木には含まれていないことを示す変数である。制約条件を満たす変数値の組み合わせさえ見つければよいので目的関数は適当な定数 C とした (式 (6))。式 (8) は k 番目のシュタイナー木において、ルートを除き各ノードに入る辺が最大 1 つであることを示している。式 (9) では、式 (8) の条件を用いると、ノード j が k 番目のシュタイナー木に含まれる場合 $n - 1 \geq u(k)_j$ となり、そうでない時 $-1 \geq u(k)_j$ となる。前者はルートからのホップ数が最大で $n-1$ となることを示しており、これは分岐のない木で j が葉である場合である。式 (9) は式 (8) と同様に考えると、j が k 番目のシュタイナー木に含まれる場合 $1 \leq u(k)_j$ となり、そうでない場合 $-1 \geq u(k)_j$ となる。式 (8) 及び式 (9) を組み合わせると、ノード j が k 番目のシュタイナー木に含まれる場合 $1 \leq u(k)_j \leq n-1$ が成り立ち、そうでなければ $u(k)_j = -1$ が成り立つ。前者はシュタイナー木に含まれるルート以外のノードはルートから 1 から $n-1$ のホップ数の距離になければならないことを示しており、後者はシュタイナー木に含まれないノードはホップ数が -1 でなければならないことを示している。式 (11) はノード j が k 番目のシュタイナー木にある場合全ての有向辺 ij に対して $u(k)_j - u(k)_i = 1$ となり、そうでなければ、$1 - n \leq u(k)_j - u(k)_i \leq 1 + n$ となる。前者は $ij \in E(\overrightarrow{G})$ に対して i から j までのホップ数はちょうど 1 になることを示している。式 (13) はルートからルートへのホップ数は 0 になり、ルートを除くシュタイナー木に含まれるノードまでのホップ数は 0 以上になることを示している。式 (15) は各辺の容量制約を示している。

IP2 の制約条件を $\{S_1, \cdots, S_N\}$ が満たすのは自明であるが、IP2 の解が 1 つのシュタイナー木を定めることを証明するのは自明ではない。証明については文献 [5] を参照していただきたい。

2.6 IP1 と IP2 の比較

まず IP1 の変数と制約の個数を見てみる。IP1 の変数の個数は明らからに $N|E|$ である。式 (2) の制約式の個数は W の可能な数に等しい。T_k のノードそれぞれについてを W に分類するか $V \setminus W$ に分類するかの 2 通りを考えるので $2^{|T_k|}$ となるが、どちらの集合も少なくとも 1 つのターミナルを含んでいる必要があり、また対称性を考えると $\left(2^{|T_k|} - 2\right)/2$ 通りの T_k の分割が考えられる。またターミナルでない $|V| - |T_k|$ 個のノードに関しても同様に考えると $\left(2^{(|V|-|T_k|)}\right)/2$ 通りの分割が考えられる。これらを全ての k に対して考えるので全部で $\sum_{k=1}^{N} \left(2^{|T_k|} - 2\right)/2 \cdot \left(2^{(|V|-|T_k|)}\right)/2$ 通りの W があることになる。式 (3), 式 (5) はそれぞれ $N|E|$ 個, $|E|$ 個の制約があることは自明である。次に IP2 の変数の個数と制約の個数を見てみる。変数の個数は各ターミナル集合につき $2|E(G)| - 1 + |V|$ 個あるので全部で $(2|E(G)| - 1 + |V|) \cdot N$ となる。各ターミナル集合について式 (8), (9), (10) はそれぞれ $|V| - 1$ 個の制約を持つ。式 (11), (12) は有向グラフを考えるのでそれぞれのターミナル集合についてそれぞれ $2|E(G)| - 1$ 個となる。式 (13) はそれぞれのターミナル集合について $|T_k|$ 個の制約を持つ。式 (15) は $|E(G)|$ 個の制約を持つ。よって IP2 は合計で $\sum_{k=1}^{N} (3 \cdot (|V| - 1) + (2|E(G)| - 1) \cdot 2 + |T_k|) + |E(G)|$ 個の制約を持つ。

IP1 のほうが変数の数は少ないものの、カットを列挙しなければならないので指数個の制約を要するという欠点がある。一方で IP2 は有向グラフを考え、またホップ数を変数として組み込んでいるために変数の数は IP1 の倍以上になってしまうという欠点があるが、一方で制約数は

表 1　実験に用いた 6 種類のネットワークトポロジ

エイリアス	ネットワークの種類	辺の容量
cr	完全グラフ	ランダム
cu	完全グラフ	一意
mr	格子型グラフ	ランダム
mu	格子型グラフ	一意
r0.5r	ランダムグラフ	ランダム
r0.5u	ランダムグラフ	一意

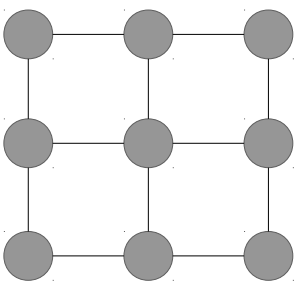

図 1　格子型グラフの例

多項式個で抑えられている。

3　実験

3.1　IP1 と IP2 の比較実験

3.1.1　実験設定

今回の実験では IP1 と IP2 がどの程度の大きさのシュタイナー木詰め込み問題を解くことができるかを検証した。実験環境は以下の通りである。

- OS: Linux 64bit
- メモリ: 24GB
- CPU: Intel Xeon E5530@2.40GHz
- コンパイラ: g++ (GCC) 4.9.1
- 整数計画ソルバ: SCIP[*1]
- 使用プログラミング言語: Python 3(グラフのランダム生成用[*2]), C++(メインプログラム[*3*4])

文献 [19] に従い本実験では表 1 に示す通り 6 種類のネットワークトポロジを用いた。完全グラフは任意の 2 つのノード間に一本の辺があるグラフである。格子型グラフは各ノードが平面上の整数の座標 $(x, y = 1, 2, \cdots, \sqrt{V})$ に対応し、距離が 1 の 2 つのノードを結ぶ辺があるグラフである (図 1)。ランダムグラフとはノード u とノード v の間に辺を 0.5 の確率で張るグラフである。辺の容量がランダムの場合、それぞれの辺に対してランダムに生成した容量を割り当て、一意の場合、ランダムに値を生成し、全ての辺にその値を割り当てる。1 つのインスタンスにつきメモリを 4GB 以上使用するか、終了までに 3 分以上かかる場合は強制的にプログラムを終了し、この場合を失敗とみなした。ノード数 $|V|$ は $\{9, 64, 169, 324, 529, 784, 1089\}$ を、ターミナル集合の数 N は $\{2, 4, 6, 8, 10\}$ を用いた。各トポロジ、$|V|$、N の組み合わせに対して辺の容量を離散一様分布 $U(1, 1000)$ から、$|T_k|$ を離散一様分布 $U(1, |V|)$ から、各ターミナル集合の最小帯域幅要求を離散一様分布 $U(1, 1000/N)$ からサンプリングし 20 個のインスタンスを生成した。

3.1.2　実験結果

表 2, 3, 4, 5, 6, 7 にそれぞれ cr, cu, mr, mu, r0.5r, r0.5u に対する IP1 と IP2 の $|V|, N$ ごとの解けた問題数、解けた問題のうち解があった問題 (YES 問題) の数を示す。これらの表をみると IP2 の成功率は全ての場合において IP1 の成功率を上回っていることが分かる。表 4, 5 を見ると、格子型グラフに対して、IP2 はノードの数が増えても解ける問題が他のトポロジーの場合よりも多いことがわかる。完全グラフやランダムグラフでは辺の数が多いため YES 問題が多いが、格子型グラフでは辺の数が少ないため NO 問題の方が多いのではないかと考え、格子型グラフのインスタンスで NO 問題の個数を調べたところほとんどの問題が NO 問題であることが分かった。そこで、我々は、IP2 が NO 問題の方が解きやすいのではないかという仮説を立てた。次節ではそれを検証する。

3.2　IP2 による YES 問題と NO 問題の解きやすさの検証実験

メッシュ型のグラフではノード数が増えても解ける問題が他のトポロジに比べて多いことが分かったが YES 問題が非常に少なかった。そこで、YES 問題と NO 問題で IP2 の解きやすさを評価するために追加実験を行った。

3.2.1　実験設定

メッシュ型のグラフで $(|V|, |T|) = (25, 4)$ の組み合わせに対して、YES 問題と NO 問題をそれぞれ 50 個生成した。YES 問題の生成方法を Algorithm 1 に示す。まず各要求に対するシュタイナー木をランダムに作り、グラフに埋め込む。G の辺 e の容量は、その辺を使っている要求の帯域幅の合計に、余裕を持たせるための定数 (ここでは 5) を加えた。前に埋め込みを決めているため、答は必ず YES である。シュタイナー木の生成法は以下の通りである。まず、全ての要求点が繋がるまで辺をランダムに選ぶ。選ばれた辺によりサイクルができてしてしまう場合はその辺は使わない。最後に、得られた木でターミナルでない葉がなくなるまで、そのような葉とその葉をつなぐ辺を削除する。

[*1] http://scip.zib.de/
[*2] https://bitbucket.org/hitochan777/misc
[*3] https://bitbucket.org/hitochan777/steiner
[*4] https://bitbucket.org/hitochan777/steiner_naive

表 2　cr に対する IP1 と IP2 の $|V|, N$ ごとの解けた問題数、YES 問題の数

IP1																																		
$\|V\|$	9					64					169					324					529					784					1089				
N	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
解けた問題の数	17	7	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YES 問題の数	17	7	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IP2																																			
$\|V\|$	9					64					169					324					529					784					1089				
N	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
解けた問題の数	20	20	20	20	20	17	5	2	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YES 問題の数	20	20	20	20	20	17	5	2	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 3　cu に対する IP1 と IP2 の $|V|, N$ ごとの解けた問題数、YES 問題の数

IP1																																		
$\|V\|$	9					64					169					324					529					784					1089				
N	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
解けた問題の数	18	9	6	6	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YES 問題の数	7	7	4	4	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IP2																																			
$\|V\|$	9					64					169					324					529					784					1089				
N	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
解けた問題の数	20	20	20	20	20	19	18	7	12	9	12	10	4	3	4	12	2	3	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YES 問題の数	9	18	18	18	19	10	15	4	10	5	8	5	1	0	1	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 4　mr に対する IP1 と IP2 の $|V|, N$ ごとの解けた問題数、YES 問題の数

IP1																																		
$\|V\|$	9					64					169					324					529					784					1089				
N	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
解けた問題の数	20	20	20	19	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YES 問題の数	10	11	11	14	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IP2																																			
$\|V\|$	9					64					169					324					529					784					1089				
N	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
解けた問題の数	20	20	20	20	20	12	8	11	2	6	15	13	9	9	9	15	11	13	13	9	19	14	16	15	11	18	18	13	16	11	16	16	18	16	12
YES 問題の数	10	11	11	15	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 5　mu に対する IP1 と IP2 の $|V|, N$ ごとの解けた問題数、YES 問題の数

IP1																																		
$\|V\|$	9					64					169					324					529					784					1089				
N	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
解けた問題の数	20	20	20	17	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YES 問題の数	15	12	19	12	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IP2																																			
$\|V\|$	9					64					169					324					529					784					1089				
N	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
解けた問題の数	20	20	20	20	20	9	5	2	2	3	7	3	4	4	3	7	6	5	3	3	5	5	4	3	5	8	4	8	5	4	6	7	3	2	3
YES 問題の数	15	12	19	15	14	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6　r0.5r に対する IP1 と IP2 の $|V|, N$ ごとの解けた問題数、YES 問題の数

IP1																																		
$\|V\|$	9					64					169					324					529					784					1089				
N	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
解けた問題の数	20	19	10	11	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YES 問題の数	13	15	8	6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IP2																																			
$\|V\|$	9					64					169					324					529					784					1089				
N	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
解けた問題の数	20	20	20	20	20	8	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YES 問題の数	13	16	18	15	16	8	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 7　r0.5u に対する IP1 と IP2 の $|V|, N$ ごとの解けた問題数、YES 問題の数

IP1																																		
$\|V\|$	9					64					169					324					529					784					1089				
N	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
解けた問題の数	20	20	17	15	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YES 問題の数	13	17	9	12	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IP2																																			
$\|V\|$	9					64					169					324					529					784					1089				
N	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
解けた問題の数	20	20	20	20	20	8	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YES 問題の数	13	16	18	15	16	8	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NO 問題を直接的に生成することは難しいため、Algorithm 1 の 1〜17 行までのアルゴリズムを用いて全ての要求を満たすグラフを生成し、ランダムな値 ($U(1, 10)$) を各辺の容量から引いた。これでも問題に解がないとは言い切れないため、実際にソルバーに解かせて解が存在しないことが分かった場合の問題のみを採用した。

Algorithm 1 YES 問題の生成アルゴリズム

Require: N 個の要求 $\{(T_1, bw_1), \cdots, (T_N, bw_N)\}$, 辺の容量が全て 0 の無向グラフ $G(V, E)$

1: **for all** $T_i, \forall i \in \{1, 2, \cdots, N\}$ **do**
2: $selected \leftarrow \{\}$
3: **while** $T_i \not\subset V(selected)$ or $G(V(selected), selected)$ が連結でない **do**
4: **for all** $e \in E$ **do**
5: 0 または 1 を 1/2 の確率で選びそれを ok とする
6: **if** $ok = 1$ and $e \notin selected$ and e がサイクルを作らない **then**
7: e を $selected$ に追加
8: **end if**
9: **end for**
10: **end while**
11: **while** $\exists v \in V$ s.t. $degree(v) = 1$ and $v \notin T_i$ **do**
12: このような v を含む $e \in E$ を $selected$ から取り除く
13: **end while**
14: **for all** $e \in selected$ **do**
15: e に対応する G の辺の容量に bw_i を加える
16: **end for**
17: **end for**
18: **for all** $e \in E$ **do**
19: 5 を e の容量に加える
20: **end for**

3.2.2 実験結果

表 8 に YES 問題と NO 問題を解くのに要した平均時間を示す。YES 問題のうち制限時間内に解けなかった問題は 3 つあった。それらに関しては問題を解くのに要した時間が制限時間の 3 分であったとみなし、平均時間を計算した。NO 問題に関して、前節で生成した問題が YES 問題である場合それらを除くと述べたが、実際には生成した問題は全て NO 問題であった。表 8 を見ると YES 問題を解くのに平均して NO 問題の場合の約 70 倍程度の時間がかかっていることがわかる。

表 8　YES 問題と NO 問題を解くのに要した時間

	YES	NO
所要時間 (sec)	36.73	0.51

4　考察

追加実験では少なくともメッシュ型のグラフに関しては NO 問題は YES 問題よりもかなり早く解くことができることがわかった。このことは、3.1 節で行った実験でメッシュ型以外のトポロジの問題がノードが 9 より大きい場合ほぼ解けなかった理由を示唆していると考えられる。メッシュ型のグラフよりも他のグラフの方が辺の数が多いため YES 問題が生成される可能性が高く、そのためグラフが大きくなると今回の制限下では全く解けなくなるということである。今回の実験で見られた現象は SCIP の特性によって引き起こされたものかもしれないし、シュタイナー木詰め込み問題に特有の現象かもしれない。今回は時間の関係上、他の整数計画ソルバーや他の整数計画問題で実験することはできなかったが、このような検証も必要だと考えられる。また、今回の実験では 6 種類のトポロジに限定してランダムにグラフを生成したが、現実世界のトポロジでの実験も必要である。

IP2 は IP1 よりも解ける問題の範囲が大きいことが分かったが、辺の個数が比較的少ないメッシュ型のグラフでもノードが 64 の問題が解けないケースが見られたため、現実世界での大規模なネットワークに適用するにはまだ実用的であるとは言えない。

5 おわりに

本研究ではシュタイナー木詰め込み問題に対して 2 つの整数計画の定式化を紹介した。1 つ目の定式化では制約式を求めるためにターミナル集合の全てのカット集合を列挙する必要があるという欠点がある。それに対して我々が提案した定式化は変数の数は増えるものの、制約式を多項式個で抑えることができる。実験は 6 種類のトポロジ、複数のノード数と要求数の組み合わせに対して定められたメモリ、時間の制約下で解が求められるかを検証した。我々の定式化は、メッシュ型のグラフでノード数が増えても解ける問題が見られたが、それらの問題の多くは解が存在しない問題であることが分かった。そのため、追加実験としてメッシュ型のグラフで解が存在する問題としない問題を解くのにかかる時間の平均を求めたところ、解が存在しない問題の方が平均して 70 倍ほど早いことがわかった。

謝辞

本研究は、JST 京都地域スーパークラスタープログラム、JSPS 科研費 24500013 および 15K15979 の助成を受けたものである。

参考文献

[1] Yash P. Aneja. An integer linear programming approach to the Steiner problem in graphs. *Networks*, Vol. 10, No. 2, pp. 167–178, 1980.

[2] Joseph Cheriyan and Mohammad R. Salavatipour. Hardness and approximation results for packing Steiner trees. *Algorithmica*, Vol. 45, No. 1, pp. 21–43, 2006.

[3] Joseph Cheriyan and Mohammad R. Salavatipour. Packing element-disjoint steiner trees. *ACM Transactions on Algorithms (TALG)*, Vol. 3, No. 4, p. 47, 2007.

[4] Matt DeVos, Jessica McDonald, and Irene Pivotto. Packing Steiner Trees. *arXiv preprint arXiv:1307.7621*, 2013.

[5] Mamadi Diané and Ján Plesník. An integer programming formulation of the steiner problem in graphs. *Mathematical Methods of Operations Research*, Vol. 37, No. 1, pp. 107–111, 1993.

[6] Michael R Garey and David S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, Vol. 32, No. 4, pp. 826–834, 1977.

[7] Martin Grötschel, Alexander Martin, and Robert Weismantel. Packing Steiner trees: a cutting plane algorithm and computational results. *Mathematical Programming*, Vol. 72, No. 2, pp. 125–145, 1996.

[8] Martin Grötschel, Alexander Martin, and Robert Weismantel. Packing Steiner trees: further facets. *European Journal of Combinatorics*, Vol. 17, No. 1, pp. 39–52, 1996.

[9] Martin Grötschel, Alexander Martin, and Robert Weismantel. Packing Steiner trees: polyhedral investigations. *Mathematical Programming*, Vol. 72, No. 2, pp. 101–123, 1996.

[10] Martin Grötschel, Alexander Martin, and Robert Weismantel. Packing Steiner trees: separation algorithms. *SIAM Journal on Discrete Mathematics*, Vol. 9, No. 2, pp. 233–257, 1996.

[11] Martin Grötschel, Alexander Martin, and Robert Weismantel. The Steiner tree packing problem in VLSI design. *Mathematical Programming*, Vol. 78, No. 2, pp. 265–281, 1997.

[12] Kamal Jain, Mohammad Mahdian, and Mohammad R. Salavatipour. Packing Steiner Trees. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pp. 266–274, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[13] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. Plenum Press, 1972.

[14] Petteri Kaski. Packing Steiner trees with identical terminal sets. *Information Processing Letters*, Vol. 91, No. 1, pp. 1–5, 2004.

[15] Thorsten Koch and Alexander Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, Vol. 32, No. 3, pp. 207–232, 1998.

[16] Bernhard Korte, HJ Prömel, and Angelika Steger. Steiner trees in VLSI-layout. *Paths, Flows, and VLSI-layout*, Vol. 9, pp. 185–214, 1990.

[17] Matthias Kriesell. Packing Steiner trees on four terminals. *Journal of Combinatorial Theory, Series B*, Vol. 100, No. 6, pp. 546–553, 2010.

[18] Lap Chi Lau. An approximate max-Steiner-tree-packing min-Steiner-cut theorem. *Combinatorica*, Vol. 27, No. 1, pp. 71–90, 2007.

[19] Andrew Lee. The minimum diameter multiple Steiner tree problem for embedding multiple VLANs. Master's thesis, Kyoto University, 2015.

[20] William R. Pulleyblank. Two Steiner tree packing problems. In *Proceedings of the twenty-seventh annual ACM Symposium on Theory of Computing*, pp. 383–387. ACM, 1995.

[21] Douglas B. West and Hehui Wu. Packing of Steiner trees and S-connectors in graphs. *Journal of Combinatorial Theory, Series B*, Vol. 102, No. 1, pp. 186–205, 2012.