Vol.2016-AL-160 No.10
2016/11/24

シュタイナー森問題に対する近似アルゴリズムの性能評価

浅野 孝夫[1,a)]

概要：シュタイナー森問題は，辺 $e \in E$ に非負のコスト $c_e \geq 0$ の付随する無向グラフ $G = (V, E)$ と k 組のターミナル点対 $s_i, t_i \in V$ $(s_i \neq t_i)$ が与えられたときに，どのターミナル点対 s_i, t_i も連結となるような辺部分集合 $F \subseteq E$ で誘導される部分グラフのなかで最小コストのものを求める問題である．この NP 困難問題に対する近似性能保証アルゴリズムとしては，Agrawal, Klein and Ravi のアルゴリズムと Gupta and Kumar のアルゴリズムが有名である．本稿では，シュタイナー森問題に対する（これら二つのアルゴリズムを含む）各種アルゴリズムを実装して様々な入力に対する実験を行い性能の比較・評価を与える．

キーワード：近似アルゴリズム，近似性能保証，主双対法，グリーディ法，シュタイナー森

Performance of Steiner Forest Algorithms

Asano Takao[1,a)]

Abstract: In the Steiner forest problem, we are given an undirected graph $G = (V, E)$ with nonnegative cost c_e for each edge $e \in E$ and k pairs of terminals, $s_i, t_i \in V$ $(s_i \neq t_i,\ i = 1, 2, \ldots, k)$. Then the problem is to find a minimum cost subset F of E such that, for each terminal pair s_i, t_i $(i = 1, 2, \ldots, k)$, there is a path connecting s_i and t_i in the subgraph $G|F$ induced by F. The Steiner forest problem is NP-hard and two approximation algorithms with performance guarantee are known: one is a primal-dual algorithm proposed by Agrawal, Klein and Ravi in 1991 and the other is a greedy algorithm proposed by Gupta and Kumar in 2015. In this article, we compare and evaluate the performance of several algorithms including two algorithms stated above by computational experiments on various kinds of input data.

Keywords: Approximation algorithms, performance guarantee, primal-dual algorithms, greedy algorithms, Steiner forest

1. はじめに

シュタイナー森問題は，辺 $e \in E$ に非負のコスト $c_e \geq 0$ の付随する無向グラフ $G = (V, E)$ と k 組のターミナル点対 $s_i, t_i \in V$ $(s_i \neq t_i,\ i = 1, 2, \ldots, k)$ が与えられたときに，どのターミナル点対 s_i, t_i も連結となるような辺部分集合 $F \subseteq E$ で誘導される部分グラフのなかで最小コストのものを求める問題である．この NP 困難問題に対する近似性能保証アルゴリズムとしては，Agrawal, Klein and Ravi のアルゴリズム [1] と Gupta and Kumar のアルゴリズム [4] が有名である．菅原，鮭川，浅野 [6] は，DIMACS のシュタイナー木問題の一部の入力に対して，これら二つのアルゴリズムの実際的性能評価を与えた．また，浅野 [3] は，それを発展させて，シュタイナー森問題に対する（これら二つのアルゴリズムを含む）各種アルゴリズムを実装して様々な入力に対する実験を行い性能の簡単な比較を与えた．本稿では，より詳細な比較・評価を与える．

2. シュタイナー森近似アルゴリズム

本節では，シュタイナー森問題に対する（近似性能保証付きの）近似アルゴリズムといくつかのヒューリスティクス（近似性能保証のないアルゴリズム）の概略を述べる．

[1] 中央大学　東京都文京区春日 1-13-27
Chuo University, Bunkyo-ku, Tokyo, 112–8551
[a)] asano@ise.chuo-u.ac.jp

2.1 Gupta and Kumar のアルゴリズム

Gupta and Kumar により提案された**大食アルゴリズム** (gluttonous algorithm) [4] は，シュタイナー森問題に対する LP 緩和を用いない最初の近似性能保証アルゴリズムであり，貪欲法に基づいている．近似性能保証は高々 96 であることが示されているが，実際的な近似性能には不明な点が多い．以下は，大食アルゴリズムの概要である．

大食アルゴリズムは，ターミナル点の集合を，互いに素なターミナル点の部分集合族で管理する．このとき現れる部分集合をスーパーノードと呼ぶ（各スーパーノードに含まれるすべてのターミナル点は同一の点と見なされることになる）．したがって，アルゴリズムのどの時点でも，その時点でのスーパーノードのすべての集合を $\mathcal{C}$ とすると，

$$\bigcup_{S\in\mathcal{C}} S = \bigcup_{i=1}^{k} \{s_i, t_i\}$$

である．アルゴリズムの開始時点で，各ターミナル点がスーパーノードを形成する．すなわち，

$$\mathcal{C} = \{\{s_1\},\{t_1\},\{s_2\},\{t_2\},\ldots,\{s_k\},\{t_k\}\}$$

である．アルゴリズムは，二つのスーパーノードを選択して，併合することを繰り返す．そこで，ある $i = 1, 2, \ldots, k$ が存在して，$s_i \in S$ かつ $t_i \notin S$ となるスーパーノード S は**活性** (active) であると呼ぶことにする．図 1 は活性なスーパーノードと（活性でない）不活性なスーパーノードの例である．

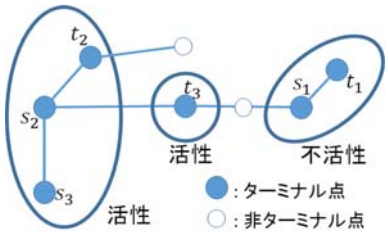


図 1 活性なスーパノードとと不活性なスーパーノード [5]

アルゴリズムは二つの活性なスーパーノードの併合（対応する二つの部分集合の和集合をとること）を，活性なスーパーノードがなくなるまで反復する．各反復で併合を行うと，併合されたスーパーノードに属するターミナル点間のコストを 0 とするので，更新されたスーパーノード間のコストは変化する．各反復の開始時の $\mathcal{C}$ と各スーパーノードに含まれる 2 点を結ぶ辺のコストを 0 としたグラフを $G_{\mathcal{C}}$ とする．$G_{\mathcal{C}}$ における（コストを長さと見なした）u, v 間の最短パスの長さを $d_{G_{\mathcal{C}}}(u, v)$ とする．その反復における二つのスーパーノード S_1, S_2 間の最短パスの長さは

$$d_{G_{\mathcal{C}}}(S_1, S_2) = \min_{u\in S_1, v\in S_2} d_{G_{\mathcal{C}}}(u, v)$$

と定義される．アルゴリズムは以下のように書ける．

(1) $\mathcal{C} = \{\{s_1\},\{t_1\},\{s_2\},\{t_2\},\ldots,\{s_k\},\{t_k\}\}$ とする．すなわち，各ターミナル点を（そしてそれのみを）スーパーノードとする．$E' = \emptyset$ とする．

(2) $G_{\mathcal{C}}$ に活性なスーパーノードが存在する限り，以下の (a)〜(c) を繰り返す．

- **(a)** $G_{\mathcal{C}}$ における二つの異なる活性なスーパーノード間で，最短パスの長さが最も短い二つの異なるスーパーノード S_1, S_2 を求める．
- **(b)** $G_{\mathcal{C}}$ における S_1, S_2 間の最短パス上の辺 e で，スーパーノード間にあるものをすべて E' に加える．
- **(c)** $\mathcal{C} = (\mathcal{C} \setminus \{S_1, S_2\}) \cup \{S_1 \cup S_2\}$ とする．

(3) E' から閉路となる辺を取り除いた辺部分集合 $F \subseteq E' \subseteq E$ を返す．

このアルゴリズムの近似性能保証が高々 96 であることの詳細については，文献 [4] を参照されたい．

計算時間についても簡単に述べる．一般性を失うことなく，$k \le n$ を仮定できることを注意しておく．$k > n$ のときには，実行可能解集合を変化させることなくターミナル点対を除去して，$k \le n$ とできるからである．点数 n，辺数 m のネットワークにおいて，Dijkstra のアルゴリズムで 1 点から全点への最短パスを求める計算時間を $S(n, m)$ と表記する．

アルゴリズムで最も時間のかかる部分は (2) の (a) である．$G_{\mathcal{C}}$ における二つの異なる活性なスーパーノード間で，最短パスの長さが最も短い二つの異なるスーパーノード S_1, S_2 を求めるのに，活性な各スーパーノードから活性な全スーパーノードへの最短パスを Dijkstra のアルゴリズムで求めると，$O(kS(n, m))$ の計算時間となる．さらに，(2) の反復ごとに活性なスーパーノードは少なくとも 1 個減るので，(2) の反復回数は高々 $2k$ であり，全体の計算時間は $O(k^2 S(n, m))$ となる．

2.2 大食アルゴリズムの縮約版

大食アルゴリズムの動作 (2) において E' に加える辺，すなわち，各反復で $G_{\mathcal{C}}$ における S_1, S_2 間の最短パス上の辺 e で，スーパーノード間にあるすべての辺，を縮約（両端点を同一視）するアルゴリズムも考えられる．簡単のため，これを縮約版という．大食アルゴリズムの縮約版の計算時間も $O(k^2 S(n, m))$ である．

図 2 のシュタイナー森問題の入力に対して，大食アルゴリズムとその縮約版を適用したときの動作を以下に示す．

大食アルゴリズムでは，最初の反復において，スーパーノード S_2 とスーパーノード S_3 が併合され図 3 のようになる．

次の反復において，スーパーノード S_1 とスーパーノード $S_2 \cup S_3$ が併合され最終的に図 4 のようになる．

一方，大食アルゴリズムの縮約版では，最初の反復にお

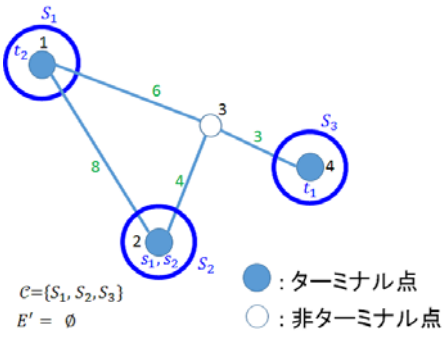


図 2 シュタイナー森問題の入力 [5]

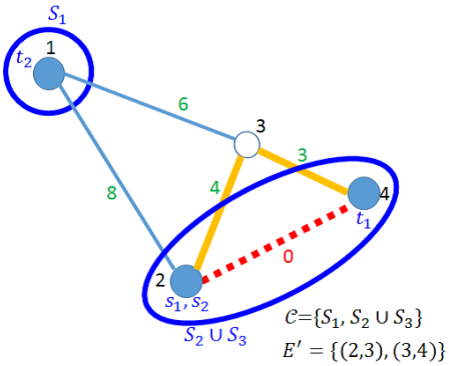


図 3 大食アルゴリズム（1 回目の反復後）[5]

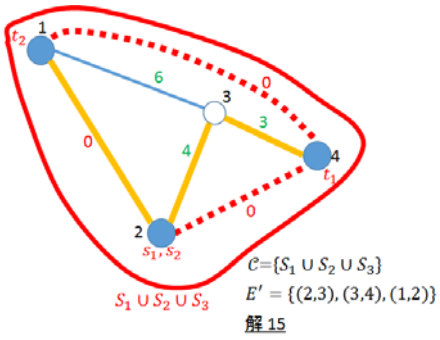


図 4 大食アルゴリズム（2 回目の反復後）[5]

いて，スーパーノード S_2 とスーパーノード S_3 が併合されると同時に，最短パス上のすべての辺のコストも 0 とされて，図 5 のようになる．

最短パス上のすべての辺のコストを 0 としたことで，次の反復では，一般に，大食アルゴリズムのときとは異なる辺が E' に加えられる．実際，この例では，次の反復において，スーパーノード S_1 とスーパーノード $S_2 \cup S_3$ が併合され最終的に図 6 のようになる．なおこの例では，大食アルゴリズムと比べて，得られる解は小さくなっている．

2.3 Agrawal, Klein and Ravi のアルゴリズム

1991 年に Agrawal, Klein and Ravi により提案されたアルゴリズム [1] は，シュタイナー森問題の IP 定式化の LP 緩和に基づく主双対アルゴリズムであり，シュタイナー森

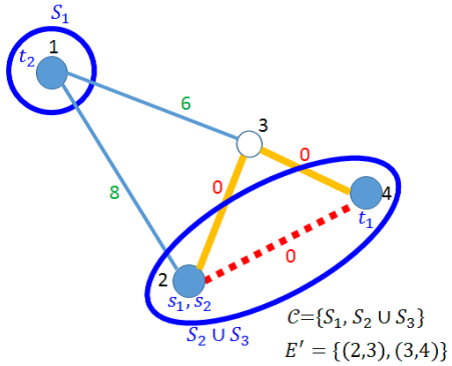


図 5 大食アルゴリズムの縮約版（1 回目の反復後）[5]

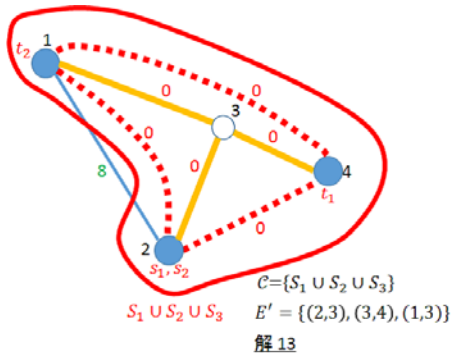


図 6 大食アルゴリズムの縮約版（2 回目の反復後）[5]

問題に対する最初の近似性能保証アルゴリズムである（詳細は文献 [2] を参照）．

そこで，はじめにシュタイナー森問題の整数計画問題としての定式化（IP 定式化）を与える．

一方の端点が部分集合 S に属し，他方の端点が S に属さないすべての辺の集合を $\delta(S)$ とし，s_i と t_i を分離する部分集合 S からなる集合族を $\mathcal{S}_i$ とする．すなわち，

$$\mathcal{S}_i = \{S \subseteq V : |S \cap \{s_i, t_i\}| = 1\}$$

とする．すると，シュタイナー森問題の IP 定式化は以下のように書ける．

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e \in \delta(S)} x_e \geq 1, \quad S \subseteq V : \quad \exists i, S \in \mathcal{S}_i, \\ & x_e \in \{0,1\}, \qquad e \in E. \end{aligned}$$

制約式の集合は，$s_i \in S, t_i \notin S$，あるいは $s_i \notin S, t_i \in S$ となる，すなわち，あるターミナル点対 s_i, t_i を分離するような S に対して（このとき $\delta(S)$ は s_i-t_i カットと呼ばれる），任意の s_i-t_i カット $\delta(S)$ から少なくとも 1 本の辺は選ばなければならないことを要求している．

この IP の制約式 $x_e \in \{0,1\}$ を $x_e \geq 0$ で置き換えると，線形計画緩和（LP 緩和，主問題）が得られ，その双対問題は以下のように書ける．

$$\begin{aligned} \max \quad & \sum_{S\subseteq V:\exists i, S\in\mathcal{S}_i} y_S \\ \text{s.t.} \quad & \sum_{S\subseteq V:e\in\delta(S)} y_S \leq c_e, \quad e\in E, \\ & y_S \geq 0, \quad S\subseteq V:\exists i, S\in\mathcal{S}_i. \end{aligned}$$

最初の制約式の左辺は，より正確には，各辺 $e\in E$ に対して，$\exists i, S\in\mathcal{S}_i, e\in\delta(S)$ を満たすようなすべての $S\subseteq V$ で y_S の和をとる．アルゴリズムは以下のように書ける．

(1) $F'=\emptyset$ とし，すべての $S\subseteq V$ で y_S を 0 とする．
(2) (V,F') で連結になっていないターミナル点対 s_i,t_i が存在する限り，以下の (a), (b) を繰り返す．
  (a) $|\,C\cap\{s_i,t_i\}\,|=1$ となるようなターミナル点対 s_i,t_i が存在する (V,F) の連結成分 $C\subseteq V$ のすべての集合を $\mathcal{C}$ とし，$C'\in\mathcal{C}$ かつ $e\in\delta(C')$ を満たす e のうちのどれかで最初に $\sum_{S:e\in\delta(S)} y_S = c_e$ となるまですべての $C\in\mathcal{C}$ の双対変数 y_C を一様に増加する．
  (b) そのような e を F' に加える．
(3) 最後に，F' から取り除いてもすべての対 s_i,t_i で連結性が保たれるような辺 $e\in F'$ をすべて取り除いて得られる F を返す．

双対変数を**堀**（あるいは風船）と見なす幾何的な解釈を用いて，図 7 のようにアルゴリズムの動作を可視化できる．

このアルゴリズムでは，どの反復の時点でも F' に属する辺の集合は森を形成することに注意しよう．このアルゴリズムの近似性能保証が高々 2 であることの詳細については，文献 [1] あるいは日本語訳書 [2] を参照されたい．

計算時間についても簡単に述べる．アルゴリズムでは，$y_S=0$ の双対変数を明示的には管理せずに，$y_S>0$ となる変数のみを管理する．したがって，(1) は $O(m)$ 時間でできる．アルゴリズムで最も時間のかかる部分は (2) の (a) である．これは，各辺 $e\in E$ に対して，$c'_e = c_e - \sum_{S:e\in\delta(S)} y_S \geq 0$ を管理しながら，$C'\in\mathcal{C}$ かつ $e\in\delta(C')$ を満たす e に対して，$C'\in\mathcal{C}$ かつ $e\in\delta(C')$ を満たす C' が唯一のとき $\delta'_e=c'_e$ とし，$C'\in\mathcal{C}$ かつ $e\in\delta(C')$ を満たす C' が正確に 2 個のとき $\delta'_e=c'_e/2$ とし，δ'_e が最小となるとなる辺 e を選ぶことで実行できる．したがって，(2) の (a) の 1 回の反復は $O(m)$ 時間でできる．また，1 回の反復ごとに点は 1 個減るので，反復回数は高々 n である．したがって，(2) の全体の計算時間は $O(mn)$ である．(3) は $O(m)$ 時間でできる．したがって，全体の計算時間は $O(mn)$ となる．

2.4 その他のアルゴリズム

その他のアルゴリズムとして，以下のヒューリスティクスを取り上げる．

最も単純なものとしては，各ターミナル点対 s_i,t_i 間を結ぶ最短パス P_i を求め，$E(P_i)$ を P_i に含まれる辺の集合とし，$E'=E(P_1)\cup E(P_2)\cup\cdots\cup E(P_k)$ とする．そして最後に，E' で誘導される部分グラフから閉路となる辺を除去して得られる辺部分集合 $F\subseteq E'\subseteq E$ を返すというヒューリスティクが挙げられる（パス上の辺をすべて縮約する縮約版も挙げられる）．このヒューリスティクの計算時間は $O(kS(n,m))$ である．

また，Gupta and Kumar のアルゴリズムが，最小全点木を求める Kruskal 版の拡張版と見なせることに注目して，シュタイナー森問題に対する Prim 版の拡張版も挙げられる．すなわち，活性なスーパーノードを 1 個選んで，そのスーパーノードから最も近い活性なスーパーノードを選んで併合し，さらに併合で得られたスーパーノードが活性なときには，そのスーパーノードから最も近い活性なスーパーノードを選んで併合し，ということを繰り返すヒューリスティクスである．なお，途中で併合で得られたスーパーノードが不活性になったときには，再度新しい活性なスーパーノードを 1 個選んで上記のことを繰り返し，最終的に，活性なスーパーノードがなくなるまで繰り返すというヒューリスティクである．このヒューリスティクの計算時間は $O(kS(n,m))$ である．

同様に，主双対アルゴリズムである最短パスを求める Dijkstra の拡張版も挙げられる．すなわち，活性なスーパーノードを 1 個選んで始点とし，その始点から最も近い（**スーパーノードとは限らない**）点を選んで併合し，さらに併合で得られたスーパーノードが活性なときには，最初の始点から最も近い点を選んで併合し，ということを繰り返すヒューリスティクスである．このときも Prim 版と同様に，途中で併合で得られたスーパーノードが不活性になったときには，再度新しい活性なスーパーノードを 1 個選んで上記のことを繰り返し，最終的に，活性なスーパーノードがなくなるまで繰り返すというヒューリスティクである．このヒューリスティクの計算時間は $O(kS(n,m))$ である．

3. 事後処理

前述のアルゴリズムで返される $F\subseteq E$ で誘導される部分グラフ $G|F=(V(F),F)$ は，いくつかの連結成分からなる．なお，$V(F)$ は

$$V(F)=\bigcup_{(u,v)\in F}\{u,v\}$$

である．このとき，各連結成分には，不要な辺（すなわち，削除しても，シュタイナー森問題の実行可能解となるような辺）も存在しうる．そこで，事後処理として，各連結成分から不要な辺をすべて除去して，得られる辺の集合を改めて F とする．さらに，$F\subseteq E$ で誘導される部分グラフ $G|F=(V(F),F)$ の各連結成分 $C\subseteq V(F)$ に対して，点集合 C で誘導される G の部分グラフの最小全点木（の辺集合）T_C を求める．すると，F_C を C を形成する $G|F$ の連結成分の辺集合とすると，T_C のコストは F_C のコスト以

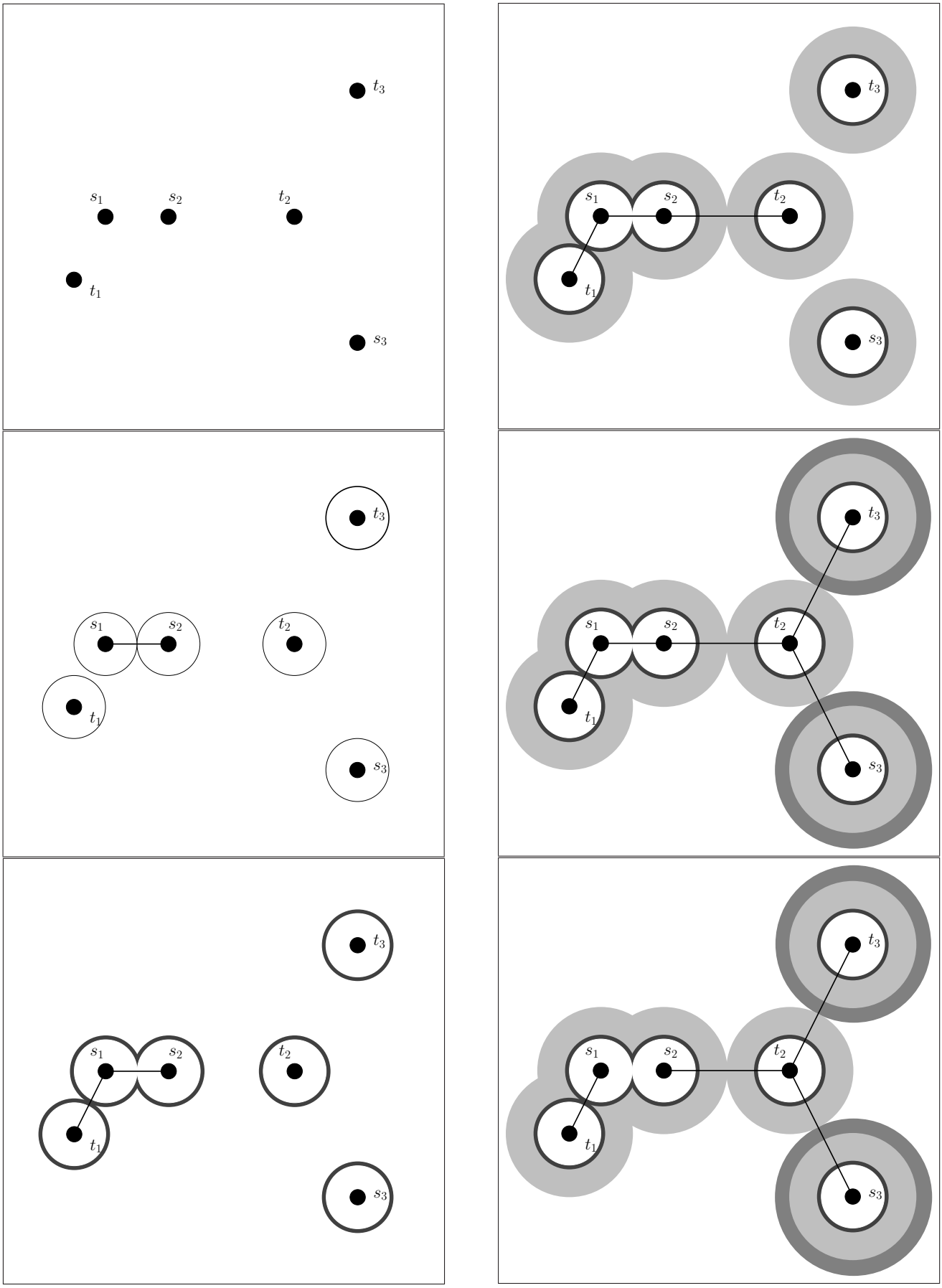
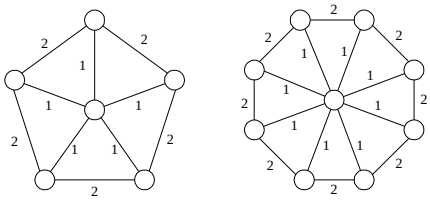
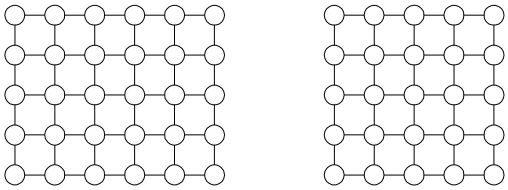


図 7 文献 [2] からの引用であるシュタイナー森問題に対する主双対アルゴリズムの説明図．(3) の削除ステップで最初の反復で加えられた辺 (s_1, s_2) が除去されている．返される最終的な辺の集合 F は最後の図に示されている．



車輪グラフ W_6　　車輪グラフ W_9

図 8　実験で用いた n 点からなる車輪グラフ W_n



5×6 格子グラフ　　5×5 格子グラフ

図 9　実験で用いた $a \times b$ 格子グラフ

下である．このことから，$G|F$ の連結成分 C を $C \in G|F$ と表記して $H = \bigcup_{C \in G|F} T_C$ とおくと，$V(H) = V(F)$ であり，$G|H = (V(H), H)$ もシュタイナー森問題の実行可能解で，コストは $G|F$ よりも小さくなりうる．そこで，改めて，$F = H$ と置いて，コストが減少する限りこれを繰り返すというヒューリスティクスが得られる．

4. 計算機実験

第 2 節で述べたアルゴリズムを C 言語で実装し，DIMACS のシュタイナー木問題のデータセットと車輪グラフ（図 8）および格子グラフ（図 9）をデータとして用いて実験をした．なお，Gupta and Kumar の大食アルゴリズムについては，（縮約をしない）論文のオリジナル版（以下では，GK と表示）と縮約と第 3 節の事後処理を適用した GK の変種版（以下では，GK 変と表示）を用いた（計算時間はともに $O(k^2 S(n,m))$）．2.4 項のその他のアルゴリズムで説明した Prim 版（以下では，Prim と表示）（計算時間は $O(kS(n,m))$），各ターミナル点対 s_i, t_i 間を結ぶ最短パス P_i を求め，$E(P_i)$ を P_i に含まれる辺の集合とし，$E' = E(P_1) \cup E(P_2) \cup \cdots \cup E(P_k)$ とする単純版（以下では，単純と表示）（計算時間は $O(kS(n,m))$），Dijkstra の拡張版（以下では，Dijk と表示）（計算時間は $O(kS(n,m))$）では，すべて縮約と第 3 節の事後処理を適用したものを用いた．Agrawal, Klein and Ravi のアルゴリズム（以下では，AKR と表示）（計算時間は $O(nm)$）も第 3 節の事後処理を適用したものを用いた．Dijkstra の最短パスアルゴリズムとして，実験では 2-ヒープを用いたので，$S(n,m) = O(m \log n)$ である．

実験に用いたパソコンの仕様は，プロセッサ：intel(R) Core(TM) i7-3840QM CPU @ 2.80GHz 2.80GHz，OS：Windows 8，メモリ：16.0GB である．

表 1　アルゴリズムで得られた解のコスト（k はターミナル点対数で括弧内の数値は近似性能）

入力	wh8000	w3c571	rl5934fst	ALUE7080	hc12p
点数	8000	3997	6827	34479	4096
辺数	15998	10278	7365	55494	24576
k	7998	2283	2967	2343	2047
GK	15996 (1.999)	3659 (1.282)	534477 (1.008)	65907 (1.055)	323079 (1.348)
GK 変	15996 (1.999)	3041 (1.065)	531300 (1.002)	63615 (1.018)	254219 (1.072)
Prim	15996 (1.999)	3041 (1.065)	531378 (1.002)	64265 (1.029)	254621 (1.074)
単純	15996 (1.999)	3178 (1.113)	532164 (1.004)	69897 (1.119)	262727 (1.108)
Dijk	7999 (1.000)	3423 (1.199)	533306 (1.006)	119744 (1.917)	335456 (1.415)
AKR	15996 (1.999)	3423 (1.199)	531930 (1.003)	65085 (1.042)	320666 (1.353)
最適解	7999	2854	529890	62449	236949

表 2　アルゴリズムの計算時間（秒）（k はターミナル点対数）

入力	wh8000	w3c571	rl5934fst	ALUE7080	hc12p
点数	8000	3997	6827	34479	4096
辺数	15998	10278	7365	55494	24576
k	7998	2283	2967	2343	2047
GK	1859.5	5.203	80.559	38.251	12.246
GK 変	1860.4	4.953	80.933	38.283	8.798
Prim	1.467	0.359	1.077	5.117	1.264
単純	1.357	0.343	0.734	5.429	1.155
Dijk	0.109	0.015	0.016	0.031	0.016
AKR	0.624	0.188	0.561	14.711	0.983

表 1 と表 2 はシュタイナー木問題の入力に対する実験結果からいくつかを選択したものである．すなわち，8000 点の車輪グラフ W_{8000}（wh8000 と表示），DIMACS のシュタイナー木問題の入力データである，w3c571（車輪グラフと奇数閉路の積のグラフ），rl5934fst（簡約水平垂直グラフ），ALUE7080（実際の VLSI で生じるグラフ），hc12p（12 次元超立方体グラフ）に対する結果である．表 1 はアルゴリズムで得られた解のコストであり，表 2 はその解を得るのに費やした計算時間である．なお，これらの DIMACS のシュタイナー木問題の入力データの w3c571，rl5934fst，ALUE7080，hc12p に対する最適なシュタイナー木のコストは既知で，それぞれ，2584，529890，62449，236949 である．また，wh8000 は，シュタイナー森問題に対する AKR のアルゴリズムの近似性能保証が 2 に限りなく近づく入力の一つであり，最適なシュタイナー木のコストは 7999 である．

表 3 から表 8 は，表 1 と表 2 の入力のネットワークに対して，ターミナル点対数 k をパラメーターとしたときの実験結果の一部である（したがって，シュタイナー木問題ではなくなり，真のシュタイナー森問題になる）．

表 3 wh8000 のターミナル点対数 k による解のコスト

k	2048	1024	512	256	128	64
GK	6809	3378	1746	860	430	206
GK 変	4096	2048	1024	512	256	128
Prim	4096	2048	1024	512	256	128
単純	4096	2048	1024	512	256	128
Dijk	4096	2048	1024	512	256	128
AKR	4096	2048	1024	512	256	128

表 4 wh8000 のターミナル点対数 k による計算時間（秒）

k	2048	1024	512	256	128	64
GK	434.172	108.609	27.188	6.796	1.688	0.437
GK 変	25.781	6.703	1.828	0.531	0.172	0.062
Prim	1.219	0.437	0.172	0.078	0.031	0.031
単純	0.625	0.203	0.078	0.031	0.016	0.016
Dijk	0.016	0.016	0.000	0.016	0.015	0.000
AKR	0.250	0.125	0.063	0.031	0.016	0.000

表 5 ALUE7080 のターミナル点対数 k による解のコスト

k	1172	586	293	146	73	36
GK	65907	34853	18352	9635	5388	4543
GK 変	63615	33580	17786	9380	5295	4484
Prim	64265	33908	17865	9422	5305	4518
単純	68867	36755	19629	10471	5816	4636
Dijk	119744	71874	42458	23617	16027	12943
AKR	65877	34852	18425	9694	5468	4582

表 6 ALUE7080 のターミナル点対数 k による計算時間（秒）

k	1172	586	293	146	73	36
GK	34.67	8.781	2.203	0.563	0.156	0,063
GK 変	34.59	8.656	2.188	0.562	0.157	0.046
Prim	4.657	1.172	0.281	0.079	0.031	0.015
単純	2.703	0.703	0.204	0.063	0.016	0.016
Dijk	0.016	0.000	0.016	0.015	0.015	0.016
AKR	13.687	7.390	5.422	4.765	4.422	4.359

表 7 hc12p のターミナル点対数 k による解のコスト

k	1024	512	256	128	64	32
GK	321774	160719	81660	40327	20813	9932
GK 変	254117	128597	65074	32470	16687	8498
Prim	254519	128198	65468	32653	16786	8398
単純	212407	106247	53127	26558	13297	6655
Dijk	334851	211058	135566	87370	63832	40969
AKR	320464	160304	81947	40020	20509	9731

表 8 hc12p のターミナル点対数 k による計算時間（秒）

k	1024	512	256	128	64	32
GK	10.907	2.813	0.750	0.188	0.047	0.016
GK 変	7.860	2.032	0.547	0.141	0.047	0.016
Prim	1.140	0.375	0.125	0.047	0.000	0.000
単純	0.016	0.000	0.000	0.000	0.000	0.000
Dijk	0.000	0.000	0.000	0.000	0.015	0.000
AKR	0.938	0.438	0.188	0.094	0.047	0.031

表 9 格子グラフ 160×160 のターミナル点対数 k による解のコスト

k	2048	1024	512	256	128	64
GK	7746	5614	4066	2908	2074	1541
GK 変	7430	5340	3863	2767	1998	1481
Prim	7450	5402	3889	2796	2009	1482
単純	8482	6088	4416	3151	2231	1582
Dijk	24602	23549	21851	17735	12645	7414
AKR	7621	5526	3987	2867	2047	1517

表 10 格子グラフ 160×160 のターミナル点対数 k による時間（秒）

k	2048	1024	512	256	128	64
GK	69.250	19.719	6.218	2.109	0.719	0.328
GK 変	65.343	18.891	5.781	1.875	0.703	0.328
Prim	4.906	2.609	1.297	0.672	0.344	0.172
単純	4.172	2.031	0.985	0.469	0.234	0.110
Dijk	0.031	0.016	0.000	0.000	0.000	0.016
AKR	12.344	9.859	8.891	7.203	6.734	6.297

表 11 格子グラフ $a \times a$ のターミナル点対数 $k = 512$ による計算時間（秒）（m は辺数で括弧内の数字は解のコスト）

m (a)	50880 (160)	25312 (113)	12640 (80)
GK	6.218 (4066)	2.922 (2824)	1.453 (1967)
GK 変	5.781 (3863)	2.938 (2670)	1.454 (1882)
Prim	1.297 (3889)	0.594 (2705)	0.265 (1893)
単純	0.985 (4416)	0.469 (3085)	0.218 (2163)
Dijk	0.000 (21851)	0.000 (11254)	0.000 (5933)
AKR	8.891 (3987)	2.594 (2778)	0.750 (1940)

表 12 車輪グラフ W_n のターミナル点対数 $k = 512$ による計算時間（秒）（n と m は点数と辺数で括弧内の数字は解のコスト）

n (m)	8000 (15998)	4000 (7998)	2000 (3998)
GK	27.188 (1746)	14.906 (1746)	7.063 (1746)
GK 変	1.828 (1024)	1.140 (1024)	0.687 (1024)
Prim	0.172 (1024)	0.110 (1024)	0.062 (1024)
単純	0.078 (1024)	0.062 (1024)	0.031 (1024)
Dijk	0.000 (1024)	0.000 (1024)	0.000 (1024)
AKR	0.063 (1024)	0.032 (1024)	0.000 (1024)

表 9 と表 10 は，格子グラフ 160×160（点数 25600，辺数 50880，辺のコスト 1）に対するターミナル点対数 k をパラメーターとしたときの解のコストと計算時間である．

表 11 と表 12 は，格子グラフと車輪グラフに対してターミナル点対数 k を固定して，格子グラフと車輪グラフで辺数 m をパラメーターとしたときの計算時間と解のコストである．

5. 観察と評価

近似性能と計算時間の観点から評価を与える．最初に，近似性能について議論する．

Gupta and Kumar の大食アルゴリズムの近似性能については，GK（縮約をしない論文のオリジナル版）と GK 変（縮約と第 3 節の事後処理を適用した GK の変種版）では，実験のどのケースでも，縮約の効果と事後処理の効果

表 13 GK における縮約の効果と事後処理の効果

入力	wh8000	w3c571	rl5934fst	ALUE7080	hc12p
点数	8000	3997	6827	34479	4096
辺数	15998	10278	7365	55494	24576
k	7998	2283	2967	2343	2047
GK	15996	3659	534477	65907	323079
GK 後	15996	3388	532074	65385	318673
GK 縮	15996	3041	531942	63745	257838
GK 縮後	15996	3041	531300	63615	254219

が確認できた．縮約と事後処理のそれぞれの効果の詳細を表 13 に与えている．なお，GK は縮約をしない論文のオリジナル版で，GK 後は GK に事後処理を適用したもの，GK 縮は GK の縮約版で，GK 縮後は GK 縮に事後処理を適用したものである．したがって，GK 縮後は，GK 変と同一である．この表から縮約と事後処理はともに効果のあることがわかる．

さらに，wh8000 と hc12p の入力を除いて，GK 変は最も良い近似性能を示した．シュタイナー森問題に対して，表 1 の入力 wh8000 では，Dijkstra 版が最適解を求めたのに対して，他のどのアルゴリズムも近似性能がほぼ 2 となった．したがって，それらのアルゴリズムは，近似性能保証が 2 より真に良くなることはないことが確認できた．しかし，Dijkstra 版は，表 5 や表 9 からもわかるように，近似性能は 2 より大きい値になることもあることが確認できた．Prim 版は GK 変とほぼ同じ近似性能を示し，AKR（Agrawal, Klein and Ravi のアルゴリズム）も Prim 版とほぼ同じ近似性能を示した．（各ターミナル点対 s_i, t_i 間を結ぶ最短パス P_i を求める）単純版は，hc12p の入力を除いて，Dijkstra 版以外の他のアルゴリズムより良い近似性能を示すことはなかった．結論として，GK 変と Prim 版と AKR は，実際的にも良い近似性能であることが確認できたと言える．

次に，計算時間について議論する．総合的に判断して，ほぼ見積もりどおりの結果が得られたと言える．

Gupta and Kumar の大食アルゴリズムについては，GK と GK 変の計算時間は，理論的にともに $O(k^2 S(n, m))$ であり，実験結果からも，ほぼ k^2 に比例することが確認できた．一方，AKR の計算時間は $O(nm)$ であり k に依存しないが，実験結果からも，k とともに増加するが，その割合は少ないことが確認できた．また，表 9 と表 10 から，m に比例することが確認できた．Prim 版と単純版の計算時間は，理論的にともに $O(kS(n, m))$ であり，実験結果からも，ほぼ k に比例することが確認できた．一方，Dijkstra 版の計算時間も，理論的に $O(kS(n, m))$ であるが，実験結果から k にそれほど依存せず，ほぼ $O(S(n, m))$ であることが確認できた．実際には，プログラムに様々な工夫を施していることから，最悪の計算時間よりもかなり計算時間が短くなっていると考えられる．

以上の観察から，実際の使用においては，k が小さいときは Gupta and Kumar の大食アルゴリズムが良い候補であり，k がそれほど大きくないときは Prim 版が良い候補であり，k が大きい（k が点数 n に近い）ときは AKR が良い候補であると言える．

6. 謝辞

図 1 から図 6 は菅原惇平氏の中央大学大学院理工学研究科情報工学専攻修士論文 [5] からの引用である．本研究に協力してくれた菅原惇平氏に感謝する．本研究は中央大学特定課題研究費および文科省科研費（研究課題番号 15K11988）から一部支援を受けた．

参考文献

[1] A. Agrawal, P. Klein, and R. Ravi: “When trees collide: An approximation algorithm for the generalized Steiner problem on networks.” *SIAM Journal on Computing* 24.3 (1995), 440-456.
[2] 浅野孝夫：“近似アルゴリズムデザイン”，共立出版，2015（D.P. Williamson and D.B. Shmoys: The Design of Approximation Algorithms, Cambridge, 2010 の邦訳）．
[3] 浅野孝夫：“シュタイナー森問題に対する近似アルゴリズムの実際的性能比較”，電子情報通信学会 2016 年ソサイエティ大会，A-1-3.
[4] A. Gupta and A. Kumar: “Greedy Algorithms for Steiner Forest”, ACM STOC (2015), 871-878.
[5] 菅原惇平：“シュタイナー森問題に対する近似アルゴリズムの実際的性能評価”，中央大学大学院理工学研究科情報工学専攻修士論文，2016 年 3 月．
[6] 菅原惇平，鮭川矩義，浅野孝夫：“シュタイナー森問題に対する近似アルゴリズムの実際的性能評価”，情報処理学会第 78 回全国大会 (2016), 6J-02 (I-401–I-402).