2009－AL－122　(9)
2009／1／30

木幅の上界を求めるMCSアルゴリズムについて

平井 一徹　山崎 浩一

群馬大学 大学院 工学研究科
〒376-8515 群馬県 桐生市 天神町 1-5-1

概要 木幅とは，グラフパラメータの一つで，グラフが木にどれだけ近いかを表すパラメータである．MCSは，Tarjan 等によって 1970 年代に提案されたアルゴリズムで [SIAM J.Comput., 13(1984) 566–579]，木幅の上界を求める上では，比較的高速かつ，良い値を出すことが知られている．Achievable Set は，近年 Lucena によって，木幅を考えるために導入された概念である [Discrete Appl. Math., 155(2007) 1055–1065]．本研究では，MCS アルゴリズムに Achievable Set の概念を導入することを試みた．そして，提案手法を実装し，実験的に評価を行った．その結果，提案手法は，一般のグラフに対しては効果が見られなかったが，特定のグラフにおいて改善が見られ，MCS が効きにくい一つのグラフ構造を特定することができた．

An Upper Bound Heuristics for Treewidth Based on MCS

Kazuyuki Hirai　Koichi Yamazaki

Department of Computer Science, Gunma University
1-5-1 Tenjin-cho, Kiryu, Gunma, 376-8515, Japan

Abstract In this paper, we propose and evaluate an algorithm for computing an upper bound on treewidth of graphs. The proposed algorithm is based on combining the well-known "Maximum Cardinality Search" algorithm (MCS for short) due to Tarjan et al. [SIAM J.Comput., 13(1984) 566–579] with the concept "Achievable Set" highlighted by Lucena [Discrete Appl. Math., 155(2007) 1055–1065]. The effectiveness of our algorithm was not experimentally confirmed for general graphs. Our experimental results, however, show that the algorithm works well for some specific graphs.

1　はじめに

木幅とは，グラフパラメータの一つで，グラフが木にどれだけ近いかを表すパラメータであり，このパラメータが小さいほどグラフが木に近い構造になる．このグラフの木幅が小さな値で抑えられていると，$\mathcal{NP}$ 完全に属する多くの問題が，多項式時間や線形時間で解けることが知られている [4].

グラフ問題における木幅を用いた多くの線形時間アルゴリズム，または，多項式時間アルゴリズムは，以下の形式である: まず，小さな幅のグラフの木分解を見つける; そして，この木分解に対して，元の問題を解くための動的計画法を行う．しかし，木幅を求めること自体が $\mathcal{NP}$ 完全に属するため [2]，木幅の上界を求める実用的なアルゴリズムを考えることには意味がある．

木幅の上界を求めるアルゴリズムは，Elimination Ordering と呼ばれる頂点の ordering を用いる方法 [6] や separator を用いる方法 [6] [1] が知られている．Maximum Cardinality Search [10] (以下 MCS と呼ぶ) は，Elimination Ordering を用いる手法に分類され，比較的高速にかつ，良い値を出すことが知られている [6].

MCS は，Rose 等によって提案された LexBFS [9]

をよりシンプルにしたアルゴリズムで，1970 年代に Tarjan 等によって提案された [10]．このアルゴリズムは，入力されたグラフがコーダルグラフであるかを判定することができ，Minimum Fill-In 問題に対する有効なアルゴリズムとして知られおり [3]，木幅の上界を求めるアルゴリズムとしても有効であることが知られている [5] [6]．

本研究では，この有名な MCS を改善するために，MCS に Achievable Set [7] を導入することを試み，その実験的な評価を行った．Achievable Set は，Lucena によって導入された木幅を考える上での概念である．

本論文の構成は次の通りである．2 節では，木幅の説明と説明に必要な諸定義を述べる．3 節では，Elimination Ordering と木幅の上界の関係を述べ，既存手法である MCS アルゴリズムについて述べる．4 節では，Achievable Set を紹介し，提案手法について述べる．5 節では，既存手法と提案手法の計算機実験による結果と考察を示す．また，実験により得られた，MCS が効きにくい構造に対する追加実験の結果と考察を示す．そして，最後にまとめと今後の課題を述べる．

2 準備

この節では，木幅の説明と説明に必要な定義を述べる．

2.1 諸定義

グラフとは，集合の組であり，$G=(V,E)$ と書く．ここで，V の要素は頂点と呼ばれる．また，E は頂点の非順序対の集合 $(E \subseteq \{\{v,w\} \mid v,w \in V, v \neq w\})$ で E の要素は辺と呼ばれる．グラフ G に辺 $\{x,y\}$ があるとき，G の 2 つの頂点 x,y は**隣接する**という．

パス P とは，$V=\{x_0,x_1,\ldots,x_k\}$，$E=\{\{x_0,x_1\},\{x_1,x_2\},\ldots,\{x_{k-1},x_k\}\}$ で与えられる空でないグラフ $P=(V,E)$ である．ここで，x_i は全て異なるものとする．**パスの長さ**とは，そのパスの辺の個数で，長さ k のパスを P_k と書く．**サイクル** C とは，パス P_{k-1} に辺 $\{x_{k-1},x_0\}$ を加えたグラフ C である．ここで $k \geq 3$ とする．**サイクルの長さ**とは，そのサイクルの辺 (または頂点) の個数で，長さ k のサイクルを C_k と書く．サイクルの頂点を結ぶ辺で，そのサイクルの辺でないものを，そのサイクルの**コード**という．

グラフ $G=(V,E)$ とグラフ $G'=(V',E')$ の関係が $V' \subseteq V$ かつ $E' \subseteq E$ ならば，G' を G の**部分グラフ**といい，$G' \subseteq G$ と書く．また，グラフ G はグラフ G' を含むという．

$G' \subseteq G$ であり，G' が $x,y \in V'$ となる辺 $\{x,y\} \in E$ を全て含むとき，G' は G の**誘導部分グラフ**であるという．このとき，V' は G において，G' を**誘導する**という．

空でないグラフ G において，どの 2 頂点もパスで結ばれているとき，G は**連結**であるという．**木** T とは，サイクルを部分グラフとして含まない連結なグラフ T である．

グラフ G が**完全グラフ**であるとは，G において任意の 2 頂点間に辺があることである．グラフ G において，V の部分集合 S によって誘導される部分グラフが完全グラフであるとき，S を G の**クリーク**という．

$v \in V(G)$ の**近傍**とは，v の G における隣接頂点全体の集合で，$N_G(v)$ と書く．より一般的に，$U \subseteq V$ に対して，U に属する頂点の $V \setminus U$ にある隣接頂点全体の集合を U の近傍といい，$N_G(U)$ と書く．$v \in V(G)$ の $N_G(v)$ が G 上でクリークを成しているとき，この頂点 v を **simplicial** と呼ぶ．

グラフ G の **Elimination Ordering** とは，全単射 $f: V \to \{1,2,\ldots,|V|\}$ であり，**Perfect Elimination Ordering (PEO)** とは，次の条件を満たす Elimination Ordering である; $\forall v \in V$ において，$\{w \mid \{v,w\} \in E \wedge f(w) > f(v)\}$ がクリークを成す．

グラフが**コーダルグラフ**であるとは，そのグラフにおいて，長さ 4 以上のどのサイクルにもコードがあるグラフである．コーダルグラフは PEO を持つことが知られている．また，コーダルグラフは，Triangulation Graph とも呼ばれる．コーダルグラフではないグラフに新しい辺を加えることで，コーダルグラフを構成できる．このとき，新しい辺を加えることを **Fill-In** と呼び，新しく付け加えられた辺を **Fill Edge(s)** と呼ぶ．グラフ G に対して，ordering π が PEO になるように Fill-In を行ったグラフを π **から構成した Fill-In Graph** と呼ぶ．

2.2 木幅

グラフの木幅と木分解は，Robertson と Seymour によるグラフマイナーの研究によって導入された [8]．グラフ $G = (V, E)$ の木分解は下記の条件を満たす $(\{X_i \mid i \in I\}, T = (I, F))$ である．

- $\bigcup_{i \in I} X_i = V$.
- $\forall\{v, w\} \in E$ に対して，$v, w \in X_i$ であるような $i \in I$ が存在する．
- $\forall v \in V$ に対して，集合 $I_v = \{i \in I \mid v \in X_i\}$ は T の連結な部分グラフを成す．

ここで，$\{X_i \in I\}$ は V の部分集合の族であり，**バッグ**と呼ばれる．また，T は木である．木分解 $(\{X_i \mid i \in I\}, T = (I, F))$ の幅とは，"最大バッグのサイズ -1" である．グラフ G の木幅 $tw(G)$ とは，G の全ての木分解の幅の中で最小の値である．

図 1，図 2 にグラフとそのグラフに対する木分解の例を示す．図 1 の丸は頂点を表し，直線は辺を表す．頂点の中の数字は，頂点番号である．図 2 の丸はバッグを表す．バッグの中の数字は，図 1 の頂点番号に対応する．図 2 の木分解は，バッグの最大サイズが 3 であるので，この木分解の幅は，2 である．そして，木幅が 1 であるグラフは，森でなければならないため [8]，このグラフの木幅は 2 である．

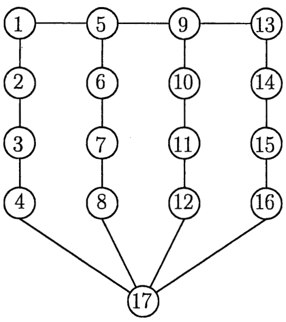

図 1　グラフ

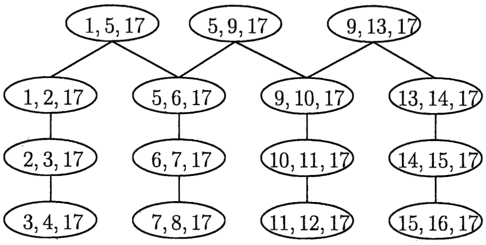

図 2　グラフの木分解

3 既存手法

この節では，Elimination Ordering と木幅の上界の関係を説明し，既存手法である MCS アルゴリズム [10] について述べる．

3.1 Elimination Ordering による木幅の上界

先に述べたように，一般のグラフの木幅を求める問題は，$\mathcal{NP}$ 完全である．しかし，もしグラフがコーダルグラフであれば，その木幅は簡単に求めることができる．コーダルグラフの木幅は，"最大クリークのサイズ -1" であり，また，コーダルグラフは PEO を持つことが知られている [5]．そして，この PEO を用いることで，コーダルグラフの最大クリークサイズを簡単に求めることができる．木幅，コーダルグラフ，Elimination Ordering の関係として，以下は同値関係である [5]．

- グラフ G の木幅が高々 k である．
- G を部分グラフとして含み，クリークの最大サイズが高々 $k + 1$ のコーダルグラフ H が存在する．
- グラフ G の次の条件を満たす Elimination Ordering π が存在する; π から構成した Fill-In Graph のクリークの最大サイズが高々 $k + 1$ となる．
- グラフ G の次の条件を満たす π が存在する; π から構成した Fill-In Graph において，$\{w \mid \{v, w\} \in E \wedge \pi(w) > \pi(v)\}$ のサイズは k を超えない．

つまり，G に辺を加えて，G を部分グラフとして含むコーダルグラフ G' の PEO を作ることができれば，$tw(G) \leq tw(G')$ となり，G の木幅の上界を容易に求めることができる．このとき，木幅の増大が起きないような最適な辺の加え方であるならば，$tw(G) = tw(G')$ となり，そのような辺の加え方を求めるアルゴリズムが良いアルゴリズムとなる．

3.2 Maximum Cardinality Search (MCS)

MCS アルゴリズムを Algorithm 1 に示す.

MCS アルゴリズムは，グラフがコーダルグラフであるかどうかを判定するアルゴリズムとして Tarjan 等によって提案され [10]，Minimum Fill-In 問題に対する有効なアルゴリズムとして知られている [3]，そして，木幅の上界を求めるアルゴリズムとしても研究され，有効であることが知られている [5] [6].

MCS アルゴリズムは，Algorithm 1 にあるように，入力としてグラフをとり，頂点の ordering を出力する．各頂点には重みと呼ばれる基準値が割り振られ，その重みを基に頂点を選択していく．そして，重みは，頂点が選択される毎にある規則に従って更新される．ここで，更新時のある規則とは，選択された頂点の隣接頂点に対して重みを 1 増やすことである.

Algorithm 1 MCS Algorithm

Input: グラフ $G = (V, E)$

Output: G の頂点の ordering π

\# 重みの初期化
for all $v \in V$ **do**
　$w(v) := 0$
end for
for $i = n$ to 1 **do**
　\# 重みを基準に頂点を選択
　未選択で最大の重み $w(v)$ を持つ v を選ぶ
　$\pi(v) = i$
　for all まだ選ばれていない $u \in N(v)$ **do**
　　\# 重みの更新
　　$w(u) = w(u) + 1$
　end for
end for

3.3 MCS による木幅の上界

3.1 項で述べたように，グラフの PEO を考えることで木幅の上界を求める．MCS アルゴリズムによって得られた ordering の逆順が PEO になるように Fill-In を行うことで，木幅の上界を求めることができる.

図 1 に対して，MCS による木幅の上界を求めた例を次に示す．図 3 は，図 1 の頂点番号 17 から開始した MCS による頂点選択の過程の例である．図 1 の頂点の中の数字は，選択された順番を表し，頂点の右下の (w) は，その頂点が選択されたときの重み w を表す．順番は，選択された順に，17 から降順になるように付けた．得られた ordering π が PEO になるように Fill-In を行い，$\{w \mid \{v, w\} \in E \wedge \pi(w) > \pi(v)\}$ を調べると，その最大サイズは 4 となるので，このときの木幅の上界は 4 である.

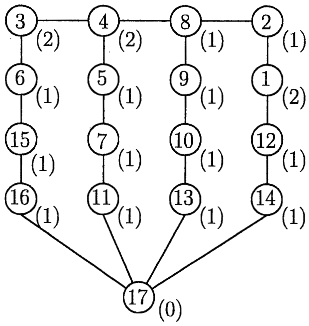

図 3　MCS

4　提案手法

この節では，既存手法である MCS に Achievable Set の概念を導入した提案手法 (以降 MCS+Achievable と呼ぶ) について述べる.

3 節の MCS アルゴリズムにおいて，頂点を選択していく際に，頂点の重みが等しい場合の選択については，特に考慮していなかった．そのため，頂点の重みが等しい状況が多数発生すると，頂点の選択次第で結果が悪くなってしまうことが考えられる．よって我々は，頂点の重みが等しいときに，新しい基準を導入することにより，MCS を改善することを考えた．具体的には，既存の MCS において，もし次の頂点を選択する際に頂点の最大の重みを持つ頂点が複数存在するならば，ここで導入する新しい基準値に従い頂点の選択を行う.

4.1　新しい基準値 (Achievable Set [7])

グラフの G において，頂点集合 $S \in V$ の **border** $\beta(S)$ を次のように定義する; もし S が連結ならば，$\beta(S) = |N(S) \setminus S|$．もし S が非連結ならば，$\beta(S) = \max_i \beta(C_i)$．ここで，$C_i$ は，S によって誘導される G の部分グラフの連結成分である.

グラフ G での ordering π の **max border**

$MB_\pi(G)$ を次のように定義する．$MB_\pi(G) = \max_i \beta(T_i)$．ここで，$T_i = \{\pi(1), \pi(2), \ldots, \pi(i)\}$.

minmax border $MMB(G)$ を次のように定義する．$MMB(G) = \min_\pi MB_\pi(G)$

$S \in V$ が **k-achievable** であるとは，次のような頂点集合 S の ordering π が存在することである; $\max \beta(T_s) \leq k$. ここで，$T_s = \{\pi(1), \pi(2), \ldots, \pi(|S|)\}$ である．

4.2 新しい基準値の使い方

MCS で頂点を選択する際，最大の重みを持つ頂点を選択候補と呼ぶ．選択候補が複数存在するときに，選択候補それぞれに対して新しい基準値を調べ，その基準値が最小となる頂点を選択する．新しい基準値とは，選択候補を一つ選んだときの β (まだ選ばれていない頂点集合) である．

MCS において，Achievable Set を基にした基準値を導入することで，重みが同点のときに，未選択頂点集合に隣接する頂点集合を小さくできる．

図 1 に対して，MCS + Achievable による木幅の上界を求めた例を次に示す．図 4 は，図 1 の頂点番号 17 から開始した MCS + Achievable による頂点選択の過程の例である．図 4 の頂点の中の数字は，選択された順番を表し，頂点の右下の (w) は，その頂点が選択されたときの重み w を表す．順番は，選択された順に，17 から降順になるように付けた．得られた ordering π が PEO になるように Fill-In を行い，$\{w \mid \{v, w\} \in E \wedge \pi(w) > \pi(v)\}$ を調べると，その最大サイズは 2 となり，最適解を導出している．

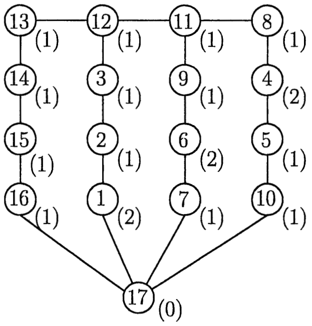

図 4　MCS+Achievable

5 計算機実験

この節では，実験に使用したグラフについて説明し，実験結果を述べる．実験環境は，CPU 2.4GHz (Intel Core 2 Duo 6600)，RAM 2GB，OS Windows XP の PC である．実装は C++ で行い，gcc 4.1.2 でコンパイルした．

5.1 実験手順と記録方法

次項以降で述べるグラフデータに対して，3 節で述べた既存手法 (MCS) と 4 節で述べた提案手法 (MCS+Achievable) を適用した．求めた値は，各グラフに対して複数回行ったときの最小値と平均値である．

提案手法は既存手法に比べて，頂点数 300 程度のグラフでは，およそ 10 倍の計算時間がかかるため，公平性を保つために，提案手法を基準にして，既存手法の実行時間が同じになるように実行を繰り返した．具体的には，提案手法は，グラフのある頂点を開始頂点とした実行を 1 回行い，それを全ての頂点に対して行った．それに対して，既存手法は，ある頂点を開始点としたときの実行を複数回行い，それを全ての頂点に対して行った．

5.2 一般のグラフに対する実験

一般のグラフに対する既存手法と提案手法による実験について述べる．

5.2.1 ベンチマークデータ

一般のグラフデータは，Bodlaender 等が Web サイト [11] で提供しているグラフデータから無作為に 20 個を選んだものを用いた．この Web サイトは，木幅を求めるアルゴリズムや木幅に関連する問題を解くアルゴリズムの比較のためのベンチマークデータを提供している．

5.2.2 ベンチマークデータに対する結果および考察

一般のグラフに対する実験結果を表 1 に示す．

表 1 より，提案手法と既存手法の最小値の差はグラフによって異なるが，そのほとんどは，1 以下である．また，平均値の差もグラフによって異なるが，そのほとんどは，1 以下であるので，アルゴリズムによる効果の違いはほとんど無いといえる．しかし，三つのグラフ (diabetes，link，munin2) に対しては大きく値が異なり，改善されていることが分かる．つまり，既存手法はこれらのグラフにおいて，うまく効

表 1　一般のグラフに対する結果

グラフ	$\|V\|$	$\|E\|$	木幅			
			既存手法		提案手法	
			最小	平均	最小	平均
1a62	122	1516	38	54.7	38	54.7
1a8o	64	536	26	35.2	28	34.9
1aba	85	886	31	43.4	31	42.9
1ail	69	631	25	31.7	25	31.5
1b67	68	559	16	19.7	16	19.5
1i07	59	397	16	22.1	17	21.8
1kth	52	426	21	25.7	21	25.8
bier127	127	368	16	19.3	17	18.2
barley	48	84	7	8.2	7	8.4
celar01	458	1499	18	23.2	19	24.2
celar02	100	311	10	11.0	10	11.1
graph01	100	283	28	35.8	28	35.2
huck	74	602	10	10	10	10
myciel4	23	71	10	12.4	10	12.3
myciel5	47	236	21	26.4	21	26.0
mainuk	48	84	6	8.4	6	8.0
oesoca	39	55	3	3	3	3
diabetes	413	602	9	17.6	7	10.3
link	724	1738	33	76.5	31	68.1
muin2	1003	1162	9	23.2	8	13.5

かない．次項でこのグラフデータに対する追加実験について述べる．

5.3　特定のグラフ構造に対する追加実験

前項で，既存手法はあるグラフにおいて，うまく効かないことを述べた．本項では，既存手法の弱点を見つけるために前項の三つのうちの一つのグラフ(diabetes) に注目して実験を行った (残り二つは，まだ解析には至っていない)．

5.3.1　特定のグラフ構造

既存手法がうまく効かなかったグラフ (diabetes) を図 5，図 6 で示す．図 5 がグラフの概略を示し，図 5 の斜線部分の詳細が図 6 である．

このデータに対して，予備実験を行いながら，簡単化すると図 7 に示すような構造を導くことができた．

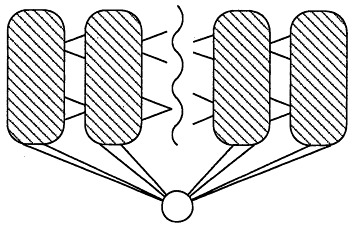

図 5　diabetes 概略

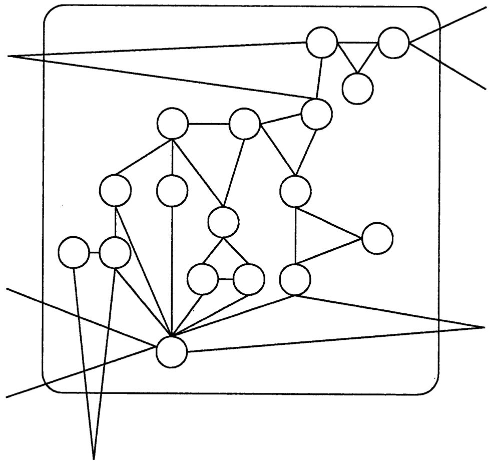

図 6　diabetes 詳細

以降の実験は，この簡単化したグラフデータに対して行う．

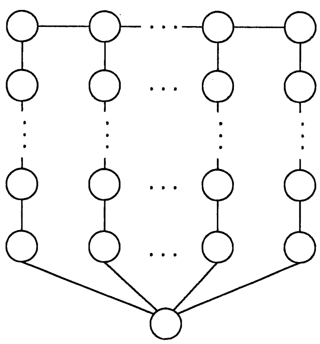

図 7　効きにくいグラフ構造

5.3.2　効きにくいグラフデータ

実験では，図 7 の行数を $5, 10, 15$ で固定し，列数を 5 から 30 で変化させたグラフを用いた．各グラフ

の木幅は，図 2 のように木分解を作れば良いため，列数や行数に依存せず 2 である．

5.3.3 効きにくいグラフデータに対する実験結果および考察

効きにくいグラフデータに対する実験結果を表 2 (行数=5)，表 3 (行数=10)，表 4 (行数=15) に示す．

表 2 効きにくいグラフ (行数 5) に対する実験結果

列数	木幅			
	既存手法		提案手法	
	最小	平均	最小	平均
5	2	4.51	2	3.84
7	3	5.82	2	4.80
9	3	6.95	2	5.78
11	4	8.23	2	6.76
13	4	9.30	2	7.75
15	5	10.3	2	8.75
17	6	11.6	2	9.74
19	6	12.7	2	10.73
21	7	14.0	2	11.73
23	7	15.3	2	12.73
25	7	16.6	2	13.73
27	8	17.8	2	14.72
29	8	18.9	2	15.72
30	8	19.5	2	16.22

表 2，表 3，表 4 より，既存手法は，グラフの列数や行数が大きくなるに従って木幅が増大する傾向が見られるが，提案手法では，グラフが大きくなっても最適な木幅を求められていることが分かる．

このグラフ構造に対して，既存手法がうまく効かない理由を次に述べる．既存手法では，重みが同点である頂点が複数あるときは，頂点をランダムに選択をする．このため，このグラフ構造においては，ある縦の列を全て選択する前に，別な列の頂点を選択することが発生する．もし，ある縦の列の頂点を全て選択する前に別な列の頂点を選択すると，ordering 決定後 Fill-In を行って PEO を考えたとき，クリークサイズが大きくなってしまい，木幅の上界が増大してしまう．このように，既存手法では，重みが同点であるときの頂点の選択によって，木幅の上界が

表 3 効きにくいグラフ (行数 10) に対する実験結果

列数	木幅			
	既存手法		提案手法	
	最小	平均	最小	平均
5	2	4.74	2	3.92
7	3	6.45	2	4.90
9	3	8.07	2	5.89
11	4	9.47	2	6.88
13	4	10.91	2	7.87
15	4	12.24	2	8.87
17	5	13.62	2	9.87
19	6	14.88	2	10.86
21	6	16.19	2	11.86
23	7	17.59	2	12.86
25	8	18.96	2	13.86
27	8	20.44	2	14.86
29	9	21.78	2	15.86
30	9	22.58	2	16.36

表 4 効きにくいグラフ (行数 15) に対する実験結果

列数	木幅			
	既存手法		提案手法	
	最小	平均	最小	平均
5	2	4.85	2	3.94
7	3	6.61	2	4.93
9	4	8.36	2	5.92
11	4	9.98	2	6.92
13	5	11.65	2	7.91
15	5	13.08	2	8.91
17	5	14.54	2	9.91
19	6	16.04	2	10.91
21	7	17.29	2	11.91
23	7	18.79	2	12.91
25	8	20.18	2	13.90
27	8	21.46	2	14.90
29	9	22.81	2	15.90
30	9	23.57	2	16.40

増えてしまう現象が多数発生するため，うまく効かない．

6 まとめと今後の課題

本論文では，MCS を改善するために，Achievable Set を導入する手法を試みた．計算機実験による評価より，提案手法は，グラフデータによっては，結果を改善することが分かった．また，既存の MCS は，ある特定の構造だと，うまく効かないことを実験的に示した．

今後の課題としては，MCS の弱点の詳細な解析とその対策を考えることが挙げられる．

参考文献

[1] E. Amir. Efficient approximation for triangulation of minimum treewidth. In *Proc.17th UAI '01*, pp. 7–15, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[2] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, Vol. 8, No. 2, pp. 277–284, 1987.

[3] A. Berry, J. R. S. Blair, P. Heggernes, and B. W. Peyton. Maximum cardinality search for computing minimal triangulations of graphs. *Algorithmica*, Vol. 39, No. 4, pp. 287–298, 2004.

[4] H. L. Bodlaender. Treewidth: Characterizations, applications, and computations. Technical Report UU-CS-2006-041, Department of Information and Computing Sciences, Utrecht University, 2006.

[5] H. L. Bodlaender and A. M. C. A. Koster. Treewidth computations I, upper bounds. Technical Report UU-CS-2008-032, Department of Information and Computing Sciences, Utrecht University, 2008.

[6] A. M. C. A. Koster, H. L. Bodlaender, and S. P. M. van Hoesel. Treewidth: Computational experiments. *Electron. Notes Discrete Math.*, Vol. 8, pp. 54 – 57, 2001. 1st Cologne-Twente Workshop on Graphs and Combinatorial Optimization.

[7] B. Lucena. Achievable sets, brambles, and sparse treewidth obstructions. *Discrete Appl. Math.*, Vol. 155, No. 8, pp. 1055 – 1065, 2007.

[8] N. Robertson and P. D. Seymour. Graph minors II: algorithmic aspects of tree-width. *J. Algorithms*, Vol. 7, pp. 309–322, 1986.

[9] D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, Vol. 5, No. 2, pp. 266–283, 1976.

[10] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, Vol. 13, No. 3, pp. 566–579, 1984.

[11] J. W. v. d. Broek and H. L. Bodlaender. TreewidthLIB a benchmark for algorithms for treewidth and related graph problems. http://people.cs.uu.nl/hansb/treewidthlib/.