G-02

最大辺重みクリーク問題に対する分枝限定法に関する一考察
A study on a branch-bound algorithm for the maximum edge-weight clique problem

池田 総志 † 山口 一章 † 増田 澄男 †
Soshi Ikeda **Kazuaki Yamaguchi** **Sumio Masuda**

1. まえがき

最大辺重みクリーク問題は，最大クリーク問題 [3] を一般化した問題であり NP 困難である．従来手法として，発見的手法である MN/TS[1] や厳密解法である清水らの手法 [2] が知られている．MN/TS は高速であるものの，局所解に陥る可能性があり最適性が保証されていない．清水らの手法は最適性が保証されるが，ロシア人形サーチと呼ばれる，小さな部分問題から順に解を求めていくボトムアップ的な計算法であり，計算を中断した際の暫定解の質が良くない．本研究では，最適性が保証され，かつ計算を中断した際の暫定解の質も良い手法を目指す．

本稿ではトップダウンな手法として分枝限定法に着目し，単純な上界計算法と彩色を用いる上界計算法での分枝限定法を比較した．

2. 最大辺重みクリーク問題

無向グラフ $G = (V, E)$ において，V の部分集合 C の任意の 2 頂点間に辺が存在するとき C はクリークと呼ばれる．グラフ G の各辺 e に重み $w(e)$ が与えたられたとき，クリークの重みを $w(C) = \sum_{u,v \in C} w((u, v))$ と定める．

最大辺重みクリーク問題とは，辺に重みのあるグラフが与えられたとき，重みが最大のクリークを見つける問題である．

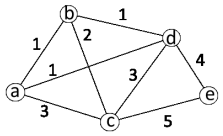

図 1(a)　入力例

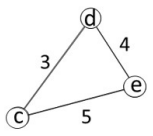

図 1(b)　最適解

3. 分枝限定法

分枝限定法は分枝操作と限定操作から構成される．分枝限定法は分枝操作によって問題を部分問題に分割し，それぞれを再帰的に探索する．その際，最適解を含み得ない不要な部分問題を枝刈りする限定操作によって，計算時間を短縮する．各部分問題で最適解に近い上界を求めることができれば，効率的に枝刈りを行うことができる．

Algorithm 1: 分枝限定法

```
1: function branch(問題Ｐ) {
2: if V′ が空 then
3:     return
4: end if
5: 上界計算
6: if 暫定解 ≥ 上界 then
7:     return
8: end if
9: 部分問題 p_1, p_2...p_k を作成
10: for i = 1 to k do
11:     branch(p_i)
12: end for
13: }
```

4　上界計算

最大辺重みクリークの自明な上界として，グラフの辺重みの総和がある．頂点数 n のグラフ $G = (V, E)$ の隣接行列で与えられた場合，その時間計算量は $O(n^2)$ である．以下では，より良い上界を得る方法として，グラフの頂点彩色を用いる方法を示す．

頂点彩色を用いてグラフ $G = (V, E)$ のクリークの重みの上界を計算する方法は次の 3 ステップからなる．

ステップ１　グラフを頂点彩色する．

ステップ２　グラフとその彩色から頂点の仮重み $w'(\cdot)$ を定める．

ステップ３　G の辺重みを無視し，頂点に重み $w'(\cdot)$ が与えられたグラフとして，クリーク中に含まれる頂点の重みの和の上界を求める．

ステップ 1 ではグラフに対して頂点彩色を行い，各頂点を色組に分ける．彩色法としては Greedy 法や DSATUR 法が知られている．各彩色法は頂点を塗る順番が異なり，Greedy 法ではあらかじめ決められた順序で頂点を塗るのに対し，DSATUR 法では頂点を一つ塗るたびに未彩色の頂点の飽和次数というものを計算する．飽和次数の値が

† 神戸大学，Kobe University

大きな頂点を次に彩色する．全て頂点が彩色されるまでこれを繰り返す．いずれの方法も時間計算量は $O(n^2)$ である．DSATUR 法の方が計算時間がかかるが，DSATUR の方が色数が少なく済むことが多い．

ステップ 2 ではグラフの各頂点 v について，接続する各辺の重みの半分を足し合わせ，仮重み $w'(\cdot)$ を計算する．この際，隣接頂点に同じ色組のものが複数あった場合は辺重みが最も大きいもののみを足し合わせる．

ステップ 3 では各色組の中で頂点重みが最大のものを探し、それらの重みの総和が上界となる．

ステップ 2，ステップ 3 の疑似コードを以下に示す．

Algorithm 2: 頂点彩色を用いた上界計算

Require: グラフ $G = (V, E)$, 各辺の重み $ew(u, v)$, 各頂点の色 $c(v)$
Ensure: 上界 K

```
Initialize w'(·)
for all v ⊆ V do
    Initialize a(·)    //maximum weight for each color class
    for all u ⊆ V such that (u, v) ⊆ E do
        if a(c(u)) < ew(u, v) then
            w'(v)− = a(c(u))/2
            w'(v)+ = ew(u, v)/2
            a(c(u)) ← ew(u, v)
        end if
    end for
end for
Initialize a(·)
for all v ⊆ V do
    if a(c(v)) < w'(v) then
        K− = a(c(v))/2
        K+ = w'(v)
        a(c(v)) ← w'(v)
    end if
end for
```

頂点彩色を用いる上界計算法の全体の計算量は $O(n^2)$ である．

5. 手法

4 節で紹介した上界計算法を分枝限定法で用い，計算時間を比較する．手法 1，手法 2，手法 3 のいずれの手法でも分枝変数をグラフの頂点とし，分枝順序を接続する辺の重みの総和の降順とした．各手法で異なるのは上界計算法のみである．

手法 1　辺の重みの総和を上界とする．

手法 2　頂点彩色を用いる方法で上界を計算する．その際，彩色に Greedy 法を用いる．

手法 3　頂点彩色を用いる方法で上界を計算する．その際，彩色に DSATUR 法を用いる．

6　計算機実験 1

本節では二つの実験を示す．一つは表 1，表 2 に示す上界計算法単体での比較である．もう一つは表 3，表 4 に示す各上界計算法を分枝限定法に導入した手法 1 〜手法 3 の分枝数，計算時間の比較である．

6.1　実験環境

各実験では，各辺密度，頂点数に対しランダムグラフを 10 個作成し実験し，その平均をとった．実験環境は次の通りである．CPU : 2.70GHz，メモリ : 8.0GB，OS : Windows10，使用言語：Java.

6.2　実験結果

表 1 を見ると，単純な辺重み総和の方法に比べ，頂点彩色を用いた手法の方が上界が大幅に下がっていることがわかる．また，頂点彩色を用いた手法でも，彩色法に DSATUR を採用した方が上界が下がっていることがわかる．これは DSATUR の方が一般的に色数を少なく彩色できるため，第 **4.2** 項のステップ 2，ステップ 3 において重みを余分に足し合わせることが少なくなるからである．

上界が下がった効果は表 3 から見て取れ，分枝数における優位性が手法 3，手法 2，手法 1 の順となっている．

しかし表 4 を見ると，計算時間における優位性は手法 1，手法 2，手法 3 の順であることがわかる．分枝限定法では分枝するたびに上界計算を行っている．この上界計算の計算コストの増加 (表 2) の影響が分枝数削減の効果よりも強く出る結果となった．

表 1　各上界計算法の上界の比較

辺密度	頂点数	厳密解	辺重み総和	Greedy	DSATUR
0.3	100	93.4	5023.3	402	368.5
	200	115.7	19110.0	1041	897.2
	300	141.5	41897.5	1866	1658.3
0.5	100	175.1	7965.5	896.9	794.8
	200	244.4	30914.5	2474.1	2225.5
	300	282.2	68998.9	4726.1	4190.0
0.7	100	395.2	10937.5	1823.5	1626.8
	200	594.2	42806.3	5257.2	4793.6

表 2　各上界計算法の計算時間の比較 [μs]

辺密度	頂点数	辺重み総和	Greedy	DSATUR
0.3	100	85	331	1267
	200	132	707	2868
	300	282	1638	6380
0.5	100	88	361	1629
	200	127	1018	3654
	300	289	1977	8985
0.7	100	88	344	1839
	200	132	864	3671

表 3　手法 1 ～手法 3 の分枝数の比較

辺密度	頂点数	手法 1	手法 2	手法 3
0.3	100	18825.7	13124.5	12878.3
	200	228169.9	114755.8	109062.7
	300	843369.6	338044.1	298709.6
0.5	100	247744.8	85139.3	77090.9
	200	7059511.5	1420386.5	1180570.0
	300	6.90017169E7	1.08976924E7	8633973.3
0.7	100	9584129.8	937453.2	757993.6
	200	2.65530181E8	1.3076241E7	9297031.0

表 4　手法 1 ～手法 3 の計算時間の比較 [ms]

辺密度	頂点数	手法 1	手法 2	手法 3
0.3	100	0.0	3.2	8.6
	200	6.8	20.4	96.4
	300	35.1	93.8	367.8
0.5	100	11.4	24.8	87.0
	200	235.5	527.3	1361.7
	300	2175.7	4589.5	11225.9
0.7	100	413.2	505.7	1389.4
	200	71906.2	62807.0	147029.1

7　手法 3 の改良

6 節の結果から，頂点彩色による上界計算法を用いながらも各部分問題での計算コストを下げる必要があると考え，手法 2 を改良した手法 4 を提案する．

7.1　手法 4

手法 4 では部分問題のサイズによって上界計算法を切り替える．部分問題のサイズが大きい場合は手法 2 と同じ処理を行う．部分問題が小さい場合，手法 2 におけるステップ 1 の頂点彩色を省略し，親問題での色組を用いて，ステップ 2，ステップ 3 の上界計算を行う．

8　計算機実験 2

手法 4 の計算機実験を行った．上界計算の切り替えは部分問題の頂点数が 30 以下になった時と設定した．実験環境は **6.1** 項と同じである．

8.1　実験結果

手法 2 に比べ手法 4 では分枝数が増加するものの計算時間は短くなっている．これはサイズが小さい部分問題で頂点彩色を省略したことで上界の精度が悪くなり分枝数が増加した影響よりも，彩色の計算コストがなくなった効果が強く出た結果である．

手法 1 と手法 4 の計算時間を比べてみると，辺密度が 0.3 と 0.5 では手法 1 の方が優れているものの辺密度 0.7 では手法 4 の方が優れていた．

表 5　手法 4 の実験結果

辺密度	頂点数	分枝数	計算時間 [ms]
0.3	100	15024.3	3.0
	200	104846.2	26.7
	300	462160.1	91.9
0.5	100	103352.9	22.1
	200	1618631.2	420.6
	300	1.14142914E7	3649.0
0.7	100	1557695.3	383.8
	200	8.25074021E7	56161.3

9　あとがき

各上界計算法について分枝限定法に導入し，比較実験の結果を示した．辺重み総和の方法と頂点彩色による手法では，後者の方が上界の精度が良く，それによって分枝数も削減できるが，各分枝での計算コストの増加から，前者よりも計算時間がかかるということがわかった．

計算機実験で得られた結果から，手法 3 の改良として手法 4 を提案した．計算時間において最も優位だった手法 1 と比べたところ，辺密度が 0.3,0.5 では手法 1 よりも多くの計算時間がかかったが，辺密度 0.7 では手法 1 より計算時間が短くなった．

今後の課題は，辺密度が小さい時にも手法 1 より計算時間の短い方法を考えることである．また，ある程度分枝限定法が高速化できれば，次の段階として LDS 法 [4] の導入が挙げられる．LDS 法は，解の探索において有望な部分問題を優先して探索する手法である．これにより分枝限定法の更なる高速化と，計算を中断した際の暫定解の精度の向上が期待できる．

参考文献

[1] Qinghua Wu, Jin-Kao Hao and Fred Glover, “Multi-neighborhood tabu search for the maximum weight clique problem," Annals of Operations Re-search, Volume 196, issue 1, pp.611-634, 2012.

[2] 清水悟司, 山口一章, 増田澄男, “分枝限定法による最大辺重みクリーク抽出法," 情報処理学会, 研究報告アルゴリズム（AL）2016-AL-160, vol.11, pp.1-6, 2016

[3] Richard M. K, “Reducibility Among Combinatorial Problems," Complexity of Computer Computations, New York: Plenum, pp.85-103, 1972.

[4] W.D.Harvey and M.L.Ginsberg, “Limited Discrepancy Search,” Proceedings of the International Joint Conference on Artificial Intelligence, pp.7-13, Montreal, August 1995.