B-10

# 最大重みクリーク抽出アルゴリズムの提案および実験的評価

## Experimental Evaluation of a New Approximation Algorithm for the Maximum Weight Clique Problem

西尾 達也† 山口 一章 † 増田 澄男 †
Tatsuya Nishio Kazuaki Yamaguchi Sumio Masuda

## 1　まえがき

無向グラフ $G = (V, E)$（以降，$n = |V|$, $m = |E|$ とする）において，$V$ のある部分集合 $C$ の，任意の二頂点間に辺が存在するとき，$C$ をクリークという．各頂点に正の重みが与えられているとき，頂点の重みの和が最大のクリークを求める問題は，最大重みクリーク問題と呼ばれる．特に，全ての頂点の重みが 1 のときは，要素数が最大のクリークを求める問題である最大クリーク問題となる．グラフ $G$ の最大クリークの要素数を $\omega(G)$ と記す．

最大重みクリーク問題は，組み合わせオークション，符号理論，並列計算，パターン認識，バイオ情報学などに応用があることが示されており，高速な解法が望まれている．しかし，最大クリーク問題や最大重みクリーク問題は NP 困難であり，近似に関しても困難であることが示されている [1].

NP 困難な最適化問題に対し，メタヒューリスティックスによる解法の研究が数多く行われている [2]．メタヒューリスティックスの多くは，局所探索法を基礎とするものであるか，得られた暫定解に局所探索法を用いて解の改良を行うものである．局所探索法は近傍が大きければよい解が得られるが，近傍の探索に多くの時間がかかる．大きな近傍に対しても用い得る高速な近似解法があれば，局所探索にそれを用いることで解がより高速に改善されることが期待できる．

我々は，比較的規模の大きな最大重みクリーク問題に対して用い得る近似解法を開発することを目指している．本稿では新たに近似最大重みクリーク抽出法を提案する．計算機を用いた実験でいくつかの従来法と比較し，その良否について考察する．

本稿の構成を以下に示す．まず **2** において最大重みクリークを求める近似解法を紹介する．**3** で提案法について述べ，**4** において計算機実験の結果を示す．**5** で本研究の結果をまとめ，今後の課題について述べる．

†神戸大学，Kobe University

## 2　従来法

最大重みクリーク問題に対する近似解法は様々なものが提案されているが，本稿では時間計算量が比較的小さな二つの方法を紹介する．

### 2.1　貪欲算法

近似最大重みクリーク抽出を行う高速な計算法としては，以下に示すような貪欲算法が考えられる．

**入力** グラフ $G = (V, E)$ と頂点の重み

**出力** 近似最大重みクリーク $C$

**手順 1** 頂点を価値のある（と思われる）ものから順に並べる．その順に $v_1, v_2, \cdots, v_n$ とする．

**手順 2** $C \leftarrow \emptyset$, $i \leftarrow 1$ とする．

**手順 3** $v_i$ が $C$ の全ての頂点に隣接するならば $C \leftarrow C \cup \{v_i\}$ とする．

**手順 4** $i \leftarrow i + 1$ とする．$i \leq n$ ならば 手順 3 に行き，そうでなければ $C$ を出力して終了する．

手順 1 での頂点の並べ方は様々なものが考えられるが，最も簡単なものとしては，頂点の重みの降順に並べる方法が考えられる．この方法は一見単純すぎて実用性がなさそうに見えるが，組合せオークションに対する Lehmann の近似アルゴリズム [3] においてパラメータ $c$ を 0 に定めたものと等価であり，メタヒューリスティックに組み込んで使う手法が提案されている [4].

この貪欲算法の計算手間を評価する．手順 1 は頂点を重みの降順に並べ替えるだけなので時間計算量は $O(n \log n)$ である．手順 3 は，$|C| \leq \omega(G)$ であることより，$G$ の隣接行列があれば $O(\omega(G))$ 時間で実行可能である．手順 3 と手順 4 からなるループは $n$ 回実行されるので，アルゴリズム全体の時間計算量は $O(n \log n + n\omega(G))$ となる．

### 2.2 系列を用いる方法

文献 [5] において，最大重みクリーク問題に対する上界計算法が提案されている．その計算では頂点の系列を生成する（以降，系列生成法１と呼ぶ）．系列生成法１は，近似解法 [6] や厳密解法 [7] で用いられている．

以下では，系列生成法１の概要を述べる．系列生成法１は $G$ の無向辺を有向辺で置き換えていく．まず $V$ から最初に重みの最も小さな頂点を選び $v_1$ とする．$v_1$ に接続する辺は $v_1$ を始点とする有向辺に置き換える．$v_i$ を決める際は，$v_1, v_2, \cdots, v_{i-1}$ からなる部分有向グラフに対し，まだ選ばれていない各頂点 $u$ を付け加えたときに，$u$ を終点とする道の中で最も長い道（経路中の重みの頂点の和が最大の道）を計算する．その値が最小のものを $v_i$ とする．$v_i$ に接続する無向辺は，$v_i$ を始点とする有向辺に置き換える．これを $v_n$ が定まるまで繰り返す．時間計算量は $O(n^2)$ である．

各頂点 $v_i$ を終点とする最長路の長さは $v_1, v_2, \cdots, v_i$ による頂点誘導部分グラフの最大重みクリークの重みの上界となることが示されている．系列生成法１は，上界が小さいような頂点を先に処理していくことにより，価値の高い頂点ほど系列の後におかれやすいという性質がある．

吉川らは，$v_n$ を $C$ に入れ，$v_n$ に隣接する頂点による部分グラフに対して同じ操作を行うという方法で近似最大重みクリークを計算する方法（以降，吉川らの方法と呼ぶ）を提案している [6]．吉川らの手法は，系列生成法１を高々$O(\omega(G))$ 回繰り返す．よってその時間計算量は $O(n^2\omega(G))$ である．

## 3 提案法

系列生成法１の改良版として，計算時間がより短い頂点系列生成法が提案されている [8]（以降，系列生成法２と呼ぶ）．提案法では系列生成法２を用いるので，まずその概略を示しておく．系列生成法２では，頂点に仮の順序が付けられているものとする．その順序に従って $v_1, v_2, \cdots, v_n$ とする．最初に $v_1$ と $v_2$ の重みが比べられる．重みの小さな頂点に接続する辺を，その頂点を始点とする有向辺に置き換える．残った頂点と $v_3$ が次の候補となり，各頂点を終点としたときの最長路の長さが小さい方について，接続する辺を有向辺に置き換える．これを全ての辺に向きが付けられるまで繰り返す．最後に，各頂点を終点とする最長の有向路の長さの昇順に，新たに $v_1, v_2, \cdots, v_n$ とする．

系列生成法２は，直感的には，頂点が価値の高さの昇順におおよそ並べられているときに，最長路の長さに従って少し並べ替えるというものである．系列生成法１では $v_i$ を定める際にその候補が $n-i+1$ 個あったのに対し，系列生成法２では対象が常に 2 個であるため，実行時間は系列生成法２の方がずっと短い．ただし，系列生成法２においても最長路の計算は毎回行われるので最悪計算量は $O(n^2)$ であり，系列生成法１と変わらない．

吉川らの手法では，系列生成法１によって解に入れる頂点を一つずつ選択するが，提案法は「別解」をその都度計算する．別解の計算では系列生成法１と系列生成法２を１：２の割合でランダムに実行して解を求める．各々の別解の時間計算量は吉川らの手法と同じく $O(n^2\omega(G))$ である．

別解を求めるアルゴリズムを以下に記す．

**入力** グラフ $G = (V, E)$, 頂点の重み $w(\cdot)$，頂点系列 $\Pi$

**出力** 近似最大重みクリーク $C$

**手順 1** $C \leftarrow \emptyset$ とする．

**手順 2** 確率 2/3 で手順 4 に行く．

**手順 3** $G, w(\cdot)$ に対し系列生成法１を適用し，頂点系列を求め，その頂点系列を改めて $\Pi$ とする．手順 5 に行く．

**手順 4** $G, w(\cdot), \Pi$ に対し系列生成法２を適用し，頂点系列を求め，その頂点系列を改めて $\Pi$ とする．

**手順 5** $\Pi$ の最後の要素を $v^*$ とする．$v^*$ を $C$ に加える．

**手順 6** $\Pi$ の要素の中で $v^*$ に隣接しない頂点を $\Pi$ から除く．

**手順 7** $\Pi$ が空でなければ，$\Pi$ の要素による $G$ の頂点誘導部分グラフを新たに $G$ とし，手順 2 に行く．

**手順 8** $C$ を出力して終了．

吉川らの手法で新たな頂点が見つかる度に上記の別解計算を行う方法を提案法とする． 別解の計算は，系列生成法１または２を高々 $\omega(G)$ 回実行する．系列生成法の時間計算量はいずれも $O(n^2)$ なので，別解の計算にかかる時間計算量は $O(n^2\omega(G))$ である．吉川らの手法による処理と別解との分岐は高々 $\omega(G)$ 回なので，提案手法の時間計算量は $O(n^2(\omega(G))^2)$ である．

## 4 計算機実験

提案法の有効性を確かめるために，ランダムグラフを用いた計算機実験を行う．以下では，吉川らの手法を単に従来法と呼ぶ．最適解を計算する時間の都合により，頂点数が $100, 200$ のグラフについては辺密度を $0.1 \sim 0.9$ とし，頂点数が $500, 900, 1000$ のグラフについては辺密度を $0.1 \sim 0.5$ とした．各頂点には 1 から 10 のランダ

表 1: ランダムグラフにおける提案法と他解法との比較結果

| | | 貪欲算法 | | 従来法 | | 提案法 | |
|---|---|---|---|---|---|---|---|
| 頂点数 | 辺密度 | 近似率 | 時間 [ms] | 近似率 | 時間 [ms] | 近似率 | 時間 [ms] |
| 100 | 0.1 | 82.86 | 0.05 | 91.35 | 0.37 | 91.52 | 0.41 |
| 100 | 0.2 | 84.84 | 0.03 | 91.10 | 0.42 | 92.26 | 0.46 |
| 100 | 0.3 | 83.84 | 0.01 | 90.80 | 0.45 | 92.77 | 0.51 |
| 100 | 0.4 | 84.75 | 0.02 | 90.68 | 0.50 | 92.98 | 0.59 |
| 100 | 0.5 | 85.29 | 0.03 | 91.01 | 0.54 | 92.87 | 0.80 |
| 100 | 0.6 | 83.05 | 0.01 | 91.19 | 0.67 | 94.01 | 1.04 |
| 100 | 0.7 | 85.04 | 0.08 | 92.31 | 0.88 | 94.41 | 1.59 |
| 100 | 0.8 | 85.97 | 0.06 | 93.01 | 1.27 | 95.74 | 2.64 |
| 100 | 0.9 | 90.33 | 0.11 | 94.82 | 2.12 | 96.88 | 6.39 |
| 200 | 0.1 | 80.31 | 0.01 | 87.20 | 1.23 | 88.36 | 1.33 |
| 200 | 0.2 | 81.24 | 0.03 | 86.82 | 1.49 | 88.98 | 1.54 |
| 200 | 0.3 | 80.95 | 0.06 | 87.61 | 1.71 | 90.57 | 1.88 |
| 200 | 0.4 | 81.40 | 0.05 | 88.45 | 1.82 | 90.95 | 2.09 |
| 200 | 0.5 | 81.63 | 0.03 | 87.01 | 2.21 | 90.76 | 2.65 |
| 200 | 0.6 | 82.43 | 0.07 | 88.30 | 2.60 | 91.30 | 3.49 |
| 200 | 0.7 | 82.57 | 0.06 | 89.99 | 3.23 | 92.61 | 5.25 |
| 200 | 0.8 | 83.94 | 0.15 | 89.52 | 4.45 | 93.27 | 8.11 |
| 200 | 0.9 | 86.57 | 0.23 | 92.26 | 7.52 | 95.06 | 19.89 |
| 500 | 0.1 | 78.81 | 0.08 | 84.80 | 8.45 | 87.27 | 8.74 |
| 500 | 0.2 | 76.91 | 0.04 | 83.80 | 8.30 | 86.27 | 8.58 |
| 500 | 0.3 | 77.95 | 0.13 | 84.74 | 8.94 | 87.93 | 9.40 |
| 500 | 0.4 | 78.67 | 0.09 | 84.08 | 9.68 | 87.57 | 10.13 |
| 500 | 0.5 | 78.81 | 0.07 | 85.21 | 10.23 | 88.33 | 10.88 |
| 900 | 0.1 | 75.42 | 0.08 | 81.57 | 12.08 | 83.31 | 12.01 |
| 900 | 0.2 | 77.19 | 0.12 | 83.05 | 12.61 | 86.36 | 13.33 |
| 900 | 0.3 | 75.69 | 0.06 | 82.96 | 13.26 | 86.04 | 13.90 |
| 900 | 0.4 | 75.90 | 0.12 | 82.28 | 14.36 | 85.73 | 15.52 |
| 900 | 0.5 | 76.95 | 0.19 | 82.86 | 15.78 | 86.89 | 17.38 |
| 1000 | 0.1 | 74.22 | 0.14 | 81.91 | 17.04 | 84.23 | 17.05 |
| 1000 | 0.2 | 76.04 | 0.11 | 81.60 | 17.56 | 84.11 | 18.08 |
| 1000 | 0.3 | 76.10 | 0.16 | 82.06 | 18.28 | 85.04 | 19.15 |
| 1000 | 0.4 | 76.62 | 0.18 | 82.59 | 19.15 | 85.79 | 19.96 |
| 1000 | 0.5 | 76.10 | 0.20 | 83.19 | 19.09 | 86.45 | 20.84 |

ムな整数値を重みとして与える．各頂点数・辺密度のグラフをそれぞれ 100 通り作成し，貪欲算法，従来法，提案法を実行し，近似率（最適解に対する百分率）及び計算時間を求めた．使用計算機の OS は Linux，CPU は Intel(R) Core(TM) i5 CPU 750 @ 2.67GHz，プログラミング言語は Java である．

表 1 に提案法及びその他の近似解法の実験結果を示す．まず，近似率について述べる．表 1 の結果より，提案法は従来法に対し，全てのグラフにおいて，近似率の改善に成功しているといえる．頂点数に着目して見ると，頂点数 100 のグラフにおいては，辺密度が 0.1 と小さいときは，近似率にさほど改善が見られないが，辺密度が 0.3 以上では 2% ほど改善されている．頂点数が 200 のときは，辺密度が 0.1 のグラフで 1% ほどの改善が，辺密度が大きくなるにつれてより改善されている．頂点数が 500,900,1000 と上がっていくと，辺密度が 0.1 のグラフにおいても 2% ほどの改善がなされており，辺密度を上げていくと，改善の度合いは大きくなっている． よって，提案法は従来法に対して，グラフの頂点数や辺密度が大きくなるほど，より近似解の改善がされているといえる．また，頂点の重みによる貪欲算法と比較してみると，従来法の時点で近似率は大きく上回っており，提案法においては規模の大きいグラフでは 10% ほど近似率が良かった． グラフの規模による近似性能の遷移については，貪欲算法では規模が大きくなるに連れ，近似性能が大きく低下していくのに対して，提案法は規模が大きくなっても貪欲算法ほどは性能が低下しなかった． 具体的に言うと，辺密度 0.5 頂点数 100 及び 1000 のグラフで比較してみると，貪欲算法の方は 9% 強の近似率の低下が見られるが，提案法の方は性能の低下を 6% 強で留めている．以上のことより，提案法は規模の大きいグラフに対しても，比較的性能を落とすこと無く，近似解を求めることが出来ると言える．

次に，計算時間について述べる． 提案法は $O(n^2(G(\omega))^2)$ であるが，$O(n^2G(\omega))$ である従来法に比べ，辺密度が小さいときは極端な計算時間の増加は見られない．この理由は，別解の計算は吉川らの手法よりもずっと高速であり，また，系列生成法 1 と 2 で同じ頂

点を選んだ際は分岐する必要がないからだと思われる．辺密度が大きいグラフでは，計算時間は従来法に比べ増加する傾向が見て取れた． 具体的には，頂点数 100，辺密度 0.1 では約 1.1 倍だった計算時間が，辺密度 0.8 のグラフでは 2.64[ms] と従来法の約 2 倍, 辺密度 0.9 のグラフでは 6.39[ms] と約 3 倍の計算時間となっている．これは，辺密度が高いグラフでは $\omega(G)$ の値が大きいため，計算量の違いがそのまま計算時間に現れたと考えられる．提案法と貪欲算法との計算時間の差は極めて大きい．いずれのグラフにおいても，計算時間は約 100 倍程度あり，計算時間に関しては貪欲算法に大きく劣る．

## 5 むすび

本稿では最大重みクリーク問題に対する従来手法に改良を加えた新たな近似解法を提案した．ランダムグラフを用いた計算機による実験を行い，頂点数が 1000 程度の比較的大きなグラフに対しても計算時間は従来法とさほど変わらず，かつ従来法に比べて良い近似解が得られることを示した．辺密度が高い場合は従来法に比べ計算時間が大幅に増えることも確認した．

今後の課題について述べる．与えられた時間内で，できるだけ良い解を得ようとするような場面では，一般にはメタヒューリスティックスによる解法が用いられる．提案法は解の改良を行っていくような計算法ではないので，単独で用いても与えられた時間を有効に使うことができない．文献 [4] では，高速な近似解法 [3] をサブルーチンとして利用したメタヒューリスティックが提案されている．同様に，提案法についても，計算時間が十分ある場合は，提案法をメタヒューリスティックスの初期解として用いたり，得られた近似解の近傍を探索するのに用いるという方法が考えられる．すなわち，メタヒューリスティックスに基づく何らかの手法に提案法を組み込んだ際に有効に使えるかどうかを今後検証する必要がある．

## 参考文献


[1] V.V. ヴァジラーニ，近似アルゴリズム，シュプリンガー・フェアラーク東京, 2002.

[2] 久保，ペドロソ，メタヒューリスティックスの数理，共立出版，2009.

[3] D.Lehmann, L.I.O'Callaghan and Y.Shoham "Truth revelation in rapid, approximately efficient combinatorial auctions," Journal of the ACM, vol.49, pp.577–602, 2002.

[4] 福田 直樹，伊藤 孝行，"組合せオークションにおける多数入札時での勝者決定の近似解法に関する一考察，" 電子情報通信学会論文誌，vol.J90–D, no.9, pp.2324–2335, 2007.

[5] 山口，増田，"最大重みクリークの重みの上界の高速な計算法," 電子情報通信学会技術研究報告，vol.105, no.7, pp.1–4，2005.

[6] 吉川，山口，増田，"最大重みクリーク問題に対する貪欲アルゴリズムの提案," 電気関係学会関西支部連合大会講演論文集 (CD–ROM)，G10–11，2008.

[7] K.Yamaguchi and S.Masuda, "A New Exact Algorithm for the Maximum Weight Clique Problem," Proceedings of the 23rd International Technical Conference on Circuits/Systems, Computers and Communications, pp.317–320, 2008.

[8] 山口，増田，"最大重みクリークを効率良く抽出するための頂点系列の生成法," 電子情報通信学会技術研究報告, vol.105, no.273, pp.39–42, 2005.