社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2005－MPS－57 (13)
2005／12／20

サイズの大きな極大クリークの効率的列挙アルゴリズム

仲川 崇史，富田 悦次
電気通信大学 電気通信学研究科 情報通信工学専攻
〒182-8585 東京都調布市調布ヶ丘 1-5-1
{tak, tomita}@ice.uec.ac.jp

概要．与えられたグラフ中の極大クリークを列挙するアルゴリズムには様々な多くの応用がある．しかし，極大クリークを全て列挙するには多大な時間を要する．本稿ではサイズの大きな極大クリークのみを効率的に列挙するアルゴリズムを提唱し，その性能を計算機実験により評価した．

An Efficient Algorithm for Generating Large Maximal Cliques

Takashi Nakagawa and Etsuji Tomita
Graduate School of Electro-Communications
The University of Electro-Communications
Chofugaoka 1-5-1, Chofu, Tokyo 182-8585, Japan
{tak, tomita}@ice.uec.ac.jp

Abstract. Generating maximal cliques of a given graph has many diverse applications. It takes, however, much time to generate all maximal cliques. In this note, we present an efficient algorithm for generating only large maximal cliques, and evaluate this algorithm by computational experiments.

1 はじめに

無向グラフ中の極大クリークまたは極大独立節点集合の抽出はグラフ理論の基礎的な問題であり，クラスタリングやバイオインフォマティクス等に対して様々な多くの応用がある [1]．この問題について多くのアルゴリズムが発表され実験的または理論的に評価がなされてきた [2]，[3]．筆者らはグラフ中の極大クリークを節点数について最適な最大時間計算量で全列挙するアルゴリズム CLIQUES を提唱し，計算機実験によりこれが実際に高速であることを示した [3], [4]．しかし，極大クリークを全列挙するには最大クリークを１つだけ抽出する場合と比較して多大な時間を要する．また，実問題においてはサイズの大きな極大クリークのみを列挙すれば十分な場合が多い．ここで，最大サイズクリークのみの全列挙は最大クリーク抽出アルゴリズムに若干の変更を加えるだけで高速に行えるが，最大クリークより僅かに小さな極大クリークを重複なく全て列挙するには極大クリーク全列挙アルゴリズムで用いられている手法も必要となる．本稿では CLIQUES を基にしたサイズの大きな極大クリークのみを効率的に列挙するアルゴリズムを提唱し，計算機実験によりその性能を評価した．

2 諸定義と記法

V を節点，E を枝の集合とする単純無向グラフ $G=(V,E)$ について，$v \in V$ に隣接している節点の集合を $\Gamma(v)$ と表す．また，集合 S に含まれる要素の数を S の大きさまたはサイズと言い，$|S|$ と表す．更に，$|\Gamma(v)|$ を v の次数と呼び $deg(v)$ と表す．節点部分集合 $C \subseteq V$ によって誘導される G の部分グラフ中の任意の節点が互いに隣接している時，C を G のクリークと呼ぶ．C が他の G のクリークの真部分集合でない時 C を極大クリークと呼ぶ．グラフ G 中で最大の極大クリークを最大クリークと呼び，その大きさを $\omega(G)$ と表す．節点部分集合 $C \subseteq V$ によって誘導される G の部分グラフ中の任意の節点が互いに隣接していない時，C を G の独立節点集合と呼ぶ．グラフ中の節点を独立節点集合に分割することを彩色と言い，その時の独立節点集合の最少数を彩色数と呼ぶ．本稿におけるサイズの大きな極大クリークとは，グラフ G と非負整数 r が与えられた時，サイズが $\omega(G)-r$ 以上の極大クリークのことを指す．この時 r を出力する極大クリークのサイズの幅と呼ぶ．

3 アルゴリズム r-CLIQUES

本稿で提唱するアルゴリズム r-CLIQUES について説明する．r-CLIQUES は極大クリーク全列挙アルゴリズム CLIQUES[3] を基にしたアルゴリズムであり，グラフ $G=(V,E)$ と幅 r を入力とし，深さ優先探索により G 中のサイズ $\omega(G)-r$ 以上の極大クリーク全てを抽出し，CLIQUES と同様に木の形で出力する．図１に r-CLIQUES を疑似 PASCAL で表す．図１において $ex\text{-}deg_R(p)$ は節点集合 R によって誘導される部分グラフにおける p に隣接する節点の次数の総和を表す．

r-CLIQUES は初めに最大クリーク抽出アルゴリズム MCQ'[5] により $\omega(G)$ を求めておき，再帰的手続き r-EXPAND において，現在保持しているクリークのサイズと $CAND$ 中のクリークサイズの上界との和

```
procedure r-CLIQUES(G = (V,E), r)
begin
{EXECUTE MCQ'}
    MCQ'(G = (V,E));
    global ω := size of the maximum clique found in G;
{SORT V}
    i := 1;
    R := V; V := ∅;
    R_max := set of vertices
      with the maximum degree in R;
    while R ≠ ∅ do
        if |R_max| ≥ 2 then
            p := a vertex in R_max such that
              ex-deg_R(p) = Max{ex-deg_R(q) | q ∈ R_max}
        else p := R_max[1]
        fi
        V[i] := p; R := R − {p};
        i := i + 1;
        for j := 1 to |R| do
            if R[j] is adjacent to p then
                deg(R[j]) := deg(R[j]) − 1
            fi
        od
        R_max := set of vertices
          with the maximum degree in R;
    od
{GENERATE MAXIMAL CLIQUES}
    global Q-size := 0;
    r-EXPAND(V,V,r)
end{of r-CLIQUES}

procedure r-EXPAND(SUBG, CAND, r)
begin
    if SUBG = ∅ then
        output "clique,"
    else
        u := the first vertex in SUBG;
        GAM_u := CAND ∩ Γ(u);
        EXT_u := CAND − Γ(u);
        NS-CAND(GAM_u, EXT_u, No);
        q := the last vertex in EXT_u;
        while EXT_u ≠ ∅ and
          Q-size + No[q] ≥ ω − r do
            output "q,";
            Q-size := Q-size + 1;
            SUBG_q := SUBG ∩ Γ(q);
            CAND_q := CAND ∩ Γ(q);
            if Q-size+|CAND_q| ≥ ω − r then
                r-EXPAND(SUBG_q, CAND_q, r)
            fi
            output "back,";
            Q-size := Q-size − 1;
            CAND := CAND − {q};
            EXT_u := EXT_u − {q};
            q := the last vertex in EXT_u
        od
    fi
end{of r-EXPAND}
```

図 1: r-CLIQUES

```
procedure NS-CAND(GAM_u, EXT_u, No)
begin
{SET Flag}
    for all vertices v ∈ GAM_u do
        Flag[v] := 1
    od
    for all vertices v ∈ EXT_u do
        Flag[v] := 0
    od
{COLOR GAM_u}
    maxno := 0; C_1 := ∅;
    while GAM_u ≠ ∅ do
        p := the first vertex in GAM_u;
        k := 1;
        while C_k ∩ Γ(p) ≠ ∅ do
            k := k + 1
        od
        if k > maxno then
            maxno := k;
            C_{maxno+1} := ∅
        fi
        C_k := C_k ∪ {p};
        GAM_u := GAM_u − {p}
    od
{NUMBER EXT_u}
    G-max := maxno;
    while EXT_u ≠ ∅ do
        p := the first vertex in EXT_u;
        k := 1;
        while C_k ∩ Γ(p) ≠ ∅ do
            k := k + 1
        od
        if k < G-max then
            No[p] := G-max
        else
            No[p] := k
        fi
        if k > maxno then
            maxno := k;
            C_{maxno+1} := ∅
        fi
        C_k := C_k ∪ {p};
        EXT_u := EXT_u − {p}
    od
{SORT EXT_u}
    i := 1;
    for k := 1 to maxno do
        for j := 1 to |C_k| do
            if Flag[C_k[j]] = 1 then
                EXT_u[i] := C_k[j];
                i := i + 1
            fi
        od
    od
end{of NS-CAND}
```

図 2: NS-CAND

が $\omega(G)-r$ 未満となった時1つ前の再帰レベルに戻るという深さ優先探索を行うことにより $\omega(G)-r$ 未満のサイズの極大クリークを抽出しないようにしている．初めに $\omega(G)$ を求めるのは，極大クリークの探索を行いながら $\omega(G)$ を求める手法では探索の初めの方でサイズ $\omega(G)-r$ 未満の極大クリークを抽出してしまうことを防げないためである．抽出した極大クリークをメモリに保存し，サイズが $\omega(G)-r$ 未満だと判明したら列挙しないという方法もあるが，これは必要なメモリ量が膨大となり得るため実用的ではない．また，r-CLIQUES では $CAND$ 中のクリークサイズの上界を手続き NS-CAND により求める．図2に NS-CAND を疑似 PASCAL で表す．NS-CAND は MCQ′ 等で用いられている逐次的近似彩色手続き NUMBER-SORT を基にした手続きである．グラフ $G=(V,E)$ について，G を彩色する独立節点集合の数を $\chi(G)$ とすると，$\omega(G)\leq\chi(G)\leq|V|$ という関係が成り立つ．これは $\chi(G)$ が厳密な彩色数でなくても成り立つ．この関係を利用して NUMBER-SORT は対象となる節点集合中のクリークサイズの上界を求める．NUMBER-SORT は対象の節点集合中の節点を逐次的に彩色し，1番目の独立節点集合に含まれる節点には "1"，2番目の独立節点集合に含まれる節点には "2",... と正整数番号を付け，付けた番号の昇順に節点を整列し，節点に付けた番号の情報を配列 No に保存する手続きである．NUMBER-SORT の実行後は，整列された節点集合の最後尾の節点の番号がその節点集合中のクリークサイズの上界となるという関係が，節点集合の最後尾から順に節点を取り除く限り常に成り立つ．NS-CAND は NUMBER-SORT と同様に $CAND$ 中の節点を彩色することにより $CAND$ 中のクリークサイズの上界を求める．但し，NS-CAND は $CAND$ を $EXT_u:=CAND-\Gamma(u)$ と $GAM_u:=CAND\cap\Gamma(u)$ に分割して扱い，番号付けと整列を行うのは EXT_u 中の節点のみである．これは，不適切な探索を防ぐために，r-EXPAND において NS-CAND 後の操作の対象となる節点は EXT_u 中の節点のみとしているためである．更に，GAM_u 中の節点のみを彩色した時の独立節点集合の数 (即ち GAM 中のクリークサイズの上界)$G\text{-}max$ について，EXT_u 中の節点の内 $G\text{-}max$ 以下の番号の独立節点集合に含まれる節点には番号 $G\text{-}max$ を，$G\text{-}max$ より大きい番号の独立節点集合に含まれる節点にはその独立節点集合の番号を付ける．これは，NS-CAND 後の操作において GAM_u 中の節点は決して $CAND$ から取り除かれず，操作が終了するまで $CAND$ 中のクリークサイズの上界は $G\text{-}max$ 未満とならないためである．上記の処理により，NS-CAND の実行後は EXT_u の最後尾の節点を EXT_u と $CAND$ から取り除くという操作を何度繰り返しても常に EXT_u の最後尾の節点の番号が $CAND$ 中のクリークサイズの上界となる．ところで，前述したように r-CLIQUES では不適切な探索を防ぐために NS-CAND 後の操作の対象から $CAND\cap\Gamma(u)$ $(u\in SUBG)$ 中の節点を除外している．この際，u は $CAND\cap\Gamma(u)$ を最大とする節点であることが望ましいが，各再帰レベルで厳密に $CAND\cap\Gamma(u)$ を最大とする u を求める処理は重いため，r-CLIQUES では $SUBG$ の先頭の節点を u としている．その代わり，r-EXPAND(V,V) の実行前に，V によって誘導される部分グラフにおいて最大次数を持つ節点の内，隣接する節点の次数の和が最大のものが V の先頭となるという関係が，V の先頭から順に節点を取り除き続ける限り常に成り立つように V の整列を行い，各再帰レベルで $CAND\cap\Gamma(u)$ が大きくなる可能性を高めている．

4 計算機実験

r-CLIQUES を C 言語で実働化し計算機実験により性能を評価した．使用した計算機は Pentium4 2.20GHz の CPU と 2GB の主メモリ及び Linux OS を搭載したものである．使用したコンパイラ及びフラグは gcc -O2 である ([3], [4] と同じ環境)．対象としたグラフはランダムグラフ及び DIMACS ベンチマークグラフである．列挙する極大クリークサイズの幅 $r=0,1,2$ の場合について実験を行った．ランダムグラフは n 個の節点について各節点間に確率 p で枝を張ったグラフである．実験では n と p の組み合わせごとに異なるグラフを10個用意し，各グラフの実行時間の平均を求めた．結果を表1に示す．表中において $r=0,1,2$ とした時の r-CLIQUES をそれぞれ 0-CLIQUES, 1-CLIQUES, 2-CLIQUES と表す．また，$dens$ は枝密度 =(グラフ中の枝数)/$\{(n-1)n/2\}$，ω は最大クリークサイズ，♯cliques は全極大クリークの数，r-♯cliques はサイズ $\omega-r$ 以上の極大クリークの数を表す．表には比較対象として CLIQUES[3]，MCQ′[5] 及び MCQ′ を基にした最大クリーク全列挙アルゴリズム all-MCQ′ の実行時間を併記する．all-MCQ′ は，現在保持しているクリーク Q，NUMBER-SORT によって求めた Q 中の全ての節点と隣接している候補節点集合中のクリークサイズの上界 $\chi'(R)$，現在までに見付かっている最大のクリーク Q_{max} に対し，MCQ′ では $|Q|+\chi'(R)\leq|Q_{max}|$ としていた探索打ち切り条件を $|Q|+\chi'(R)<|Q_{max}|$ と変更し，全ての最大クリークを列挙するようにしたアルゴリズムである．

結果より，全体的に r-CLIQUES は CLIQUES よ

表 1: 実験結果

ランダムグラフデータ

n	p	ω	#cliques	2-#cliques	1-#cliques	0-#cliques
300	0.5	12	4.4×10^6	26,657-49,437	644-1,635	3-30
300	0.7	20	3.0×10^9	17,628-81,329	242-2,035	1-19
500	0.3	8-9	7.1×10^5	14,096-239,686	225-14,705	1-196
500	0.5	13-14	9.6×10^7	4,611-279,787	39-6,598	1-73
1,000	0.3	9-10	1.5×10^7	50,829-1,694,767	371-53,082	1-388
3,000	0.1	6-7	2.9×10^6	193,351-2,723,714	992-201,162	1-1,066
3,000	0.2	9	1.1×10^9	848,914-884,849	4,170-4,421	1-6
5,000	0.1	7	1.8×10^7	2,491,106-2,526,341	21,257-21,968	14-25
5,000	0.2	9-10	1.1×10^9	249,278-31,199,880	359-262,661	1-388
10,000	0.1	7	2.3×10^8	76,597,392-77,548,534	1,371,893-1,395,156	1,890-2,021

ランダムグラフに対する実行時間 [sec]

n	p	CLIQUES	2-CLIQUES	1-CLIQUES	0-CLIQUES	all-MCQ'	MCQ'
300	0.5	7.61	2.50	1.46	0.99	0.32	0.166
300	0.7	5,515.36	352.15	216.33	145.78	50.05	30.89
500	0.3	1.3	0.78	0.43	0.30	0.096	0.064
500	0.5	202.33	44.18	27.66	19.25	6.14	4.67
1,000	0.3	32.40	15.15	9.60	7.54	1.86	1.57
3,000	0.1	10.92	12.56	8.41	7.44	1.60	1.37
3,000	0.2	337.37	212.75	160.48	130.46	30.74	26.05
5,000	0.1	86.68	85.99	65.56	53.16	12.56	11.67
5,000	0.2	4,942.70	3,084.44	2,293.33	1,791.83	463.87	420.74
10,000	0.1	1,856.50	1,644.88	1,171.48	946.43	210.45	185.88

DIMACS ベンチマークグラフデータ

Name	n	$dens$	ω	#cliques	2-#cliques	1-#cliques	0-#cliques
MANN_a9	45	0.927	16	590,887	393,804	131,076	9,540
brock200_3	200	0.605	15	4,595,644	496	10	1
c-fat200-5	200	0.426	58	7	7	5	3
hamming6-2	64	0.905	32	1,281,402	2	2	2
johnson16-2-4	120	0.765	8	2,027,025	2,027,025	2,027,025	2,027,025
keller4	171	0.649	11	10,284,321	388,672	10,752	2,304
p_hat300-2	300	0.489	25	79,917,408	6,593	935	52
san200_0.7_1	200	0.700	30		1	1	1

DIMACS ベンチマークグラフに対する実行時間 [sec]

Name	CLIQUES	2-CLIQUES	1-CLIQUES	0-CLIQUES	all-MCQ'	MCQ'
MANN_a9	0.23	0.61	0.23	0.030	0.0075	0.00018
brock200_3	7.34	0.90	0.55	0.38	0.11	0.071
c-fat200-5	0.0054	0.012	0.0088	0.0057	0.0031	0.00252
hamming6-2	1.21	0.012	0.0088	0.0057	0.0031	0.00011
johnson16-2-4	4.38	6.44	6.49	5.35	0.97	0.21
keller4	4.98	2.42	0.95	0.51	0.070	0.037
p_hat300-2	99.72	1.21	0.64	0.39	0.069	0.0425
san200_0.7_1	$> 24hrs.$	0.042	0.033	0.027	22.57	0.027

りも大幅に実行時間の短縮となり，特に $r = 0$ の場合多くのグラフで CLIQUES の 1/10 以下の実行時間となった．これより，列挙する極大クリークをサイズの大きいものに限定することにより実行時間が大幅に増大しないで済むことが確認できた．ただ，c-fat200-5 や johnson16-2-4 のようにグラフ中の殆ど或いは全ての極大クリークのサイズが大きいグラフでは CLIQUES の方が高速となった．これは，これらのグラフではサイズの小さな極大クリークが少ないため分枝限定の効果が小さく，分枝限定処理の重みにより逆に実行時間が増大したものと考えられる．また，殆どの場合 r-CLIQUES($r = 0, 1, 2$) の実行時間は MCQ' の 10 倍以下となった．これより，MCQ' と比較して r-CLIQUES の実行時間はさほど膨大ではないと言える．なお，殆どのグラフで 0-CLIQUES よりも all-MCQ' の方が 2 倍以上高速となった．よって，最大クリークのみを列挙する場合は all-MCQ' の方が優れていると言える．

5 まとめ

極大クリーク全列挙アルゴリズム CLIQUES を基にしたサイズの大きな極大クリークを効率的に列挙するアルゴリズム r-CLIQUES を提唱し，その性能を計算機実験により評価した．その結果，多くの場合で極大クリークを全列挙するよりも大幅に実行時間が削減できることを確認した．また，最大クリークを 1 つだけ抽出するのに要する時間と比較しても，r-CLIQUES($r = 0, 1, 2$) の実行時間はさほど膨大ではないことも確認した．ただし，最大クリークのみを列挙する場合は，MCQ' に簡単な変更を施したアルゴリズムの方が高速となることも確認した．

謝辞．本研究に有用なコメントをいただいた卒研生の東貴紀氏，及び貴重なご意見をいただいた京大バイオインフォマティクスセンター阿久津達也教授に感謝します．なお，本研究は科学研究費補助金基盤研究 (B) の支援を受けている．

参考文献

[1] S. Mohseni-Zadeh, A. Louis, P. Brézellec, and J.-L. Risler, "PHYTOPROT: a database of clusters of plant proteins," Nucleic Acids Res. 32, pp.D351-D353 (2004).

[2] K. Makino and T. Uno, "New algorithms for enumerating all maximal cliques," SWAT 2004, LNCS 3111, pp.260-272 (2004).

[3] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments," Theoret. Comput. Sci. (Special Issues on COCOON 2004, to appear).

[4] 仲川崇史, 冨田悦次, "極大クリーク全列挙アルゴリズムの実験的比較評価," 情処研報, 2004-MPS-52, pp.41-44 (2004).

[5] 亀田宗克, 冨田悦次, "最大クリーク抽出アルゴリズムの高速化と解析・評価," 情処研報, 2004-AL-93, pp.32-39, (2004).