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

2007－MPS－67 (7)
2007－BIO－11 (7)
2007／12／20

理論評価付き最大クリーク抽出アルゴリズムの実験的評価

玉田 和洋† 富田 悦次† 中西 裕陽†

あらまし 最大クリーク抽出問題に対して，幾つかのアルゴリズムと最大時間計算量が与えられている．本稿では，筆者らの理論上優れた評価のアルゴリズムが実働上も高速であることを示す．

キーワード 最大クリーク，アルゴリズム，最大時間計算量，実験的評価

Experimental Evaluations of Algorithms with Known Theoretical Time-Complexity for Finding a Maximum Clique

Takahiro TAMADA† Etsuji TOMITA† Hiroaki NAKANISHI†

Abstract. Some results have been given for the worst-case time complexity of algorithms for finding a maximum clique. We demonstrate by computational experiments that our theoretically better algorithm also generally runs faster than another algorithm with known theoretical time-complexity.

Key words maximum clique, algorithm, worst-case time complexity , experimental evaluation

1 はじめに

無向グラフ中の最大クリークを抽出する問題は，理論と応用の両面において重要な問題であり，多くの研究が行われてきた．この問題はNP困難な問題であり，過去多くの研究者によって多くのアルゴリズムが考案された．

計算量理論の視点から，この問題は自明な時間計算量が$O(P(n)2^n)$(nは対象グラフの節点数，$P(n)$はnを変数とする多項式)となることが知られている．この時間計算量をはじめにTarjan，Trojanowskiは，$O(2^{0.333n})$[1]まで改善した．この結果を基に時間計算量の改善が進められ，一般のグラフに対する多項式計算領域での最近の結果はFominらによって考案されたアルゴリズムMISの$O(2^{0.288n})$[3]であった．

これらの背景を踏まえ，中西，富田は極大クリーク全列挙アルゴリズムCLIQUES[4]を基に，最大クリーク抽出に適した形に特化した多項式領域アルゴリズムMAXCLIQUE[2]の計算量解析を行い，$O(2^{0.19669n})$[6]となる結果を示した．

†電気通信大学 情報通信工学科
Department of Information Communication and Engineering, The University of Electro-Communications, Tokyo 182-8585, Japan
E-mail:{celesty,tomita,hironaka}@ice.uec.ac.jp

しかしながら，これらの結果は理論的な解析によるもので，実働上高速である保証はない．また，実際に最大クリークを求められる範囲に限れば，理論的に優れていても，実験的には理論的に劣るアルゴリズムより遅くなってしまうこともある[2]．

以上のことから本稿では，理論的に優れたアルゴリズムMAXCLIQUEが，実働上でも理論評価のある他のアルゴリズム[3]よりもほぼ全般的に高速であることを示す．

2 アルゴリズム

2.1 アルゴリズム MAXCLIQUE

MAXCLIQUE[2]は，以下の一連の処理によって最大クリークを探索する．

はじめに与えられた節点集合($SUBG$とよぶ)中の最大次数となる節点uに着目し，$SUBG$を

$SUBG_u = SUBG \cap \Gamma(u)$

と，uに隣接しない

$EXT_u = SUBG - SUBG_u - \{u\}$

に分割する．次に$SUBG_u$を改めて$SUBG$とおき再帰的に繰り返して，逐次大きなクリークを探索する．EXT_uについては次数降順に節点を選択し，同様の処理を行う．

しかし，このままでは無駄な探索が多数あり非効率であるから，MAXCLIQUE では次数を利用した限定操作を行っている．この限定操作では，探索木の根からある節点までの深さとこの節点から延びる 1 段深い再帰での節点集合の大きさ (つまり，この深さでの節点集合におけるこの節点の次数) の和が最大クリークの上界となることを用いる．

また，EXT_u は次数降順に探索を行うので，EXT_u 中の節点について限定操作が行われれば，まだ探索を行っていない残りの EXT_u の節点に対しても限定操作を行うことができる．

2.1.1 近似彩色の導入

MAXCLIQUE は，計算論的には優れたアルゴリズムであるが，アルゴリズムを解析するために最大クリーク上界による緩い限定操作しか行われていない．理論解析するだけであればこれでよいが，実働上においてもこのアルゴリズムの有効性を発揮するためには，十分であるとは言えない．なぜならば，たとえばある節点の次数が n であれば，その節点以降の再帰の深さは $1 \sim n$ と，かなりばらつきがあることがわかる．これでは次数が同じである節点が多くあり，それらが互いに隣接していない場合はこの限定操作だけでは探索効率が落ちることが予想される．

そこで，近似彩色による新たな限定条件を導入することによりさらに効率のよい探索ができると思われる．ところで，この彩色によって時間計算量結果を変えてしまっては，MAXCLIQUE の理論的保証を失い，本稿の意義を失ってしまう．しかし，MAXCLIQUE の再帰 1 回あたりの時間計算量が $O(n^2)$[6] となっているので，$O(n^2)$ 以下の処理であれば理論的オーダー評価結果に影響を及ぼさない．以上のことから，MCQ[5] などで利用されている $O(n^2)$ となる近似彩色アルゴリズムを導入した．この限定操作では，近似彩色アルゴリズムで用いた色の数と探索路の深さの和が最大クリークの上界となることを利用している．近似彩色に用いた色の数は明らかに次数による限定操作で用いる節点集合の大きさ以下となるので，MAXCLIQUE と比べより有効な上界での限定操作であると言える．

以上のことから，理論的保証を保ちつつ探索の効率化を行えると考えられる．近似彩色を導入したアルゴリズム MAXCLIQUE1 を以下に示す．なお，MAXCLIQUE1 のコードから手続き Number による限定処理を除いたものが MAXCLIQUE となる．

```
procedure MAXCLIQUE1(G)
begin
  Q := ∅;
  Qmax := ∅;
  EXPAND(V)
end {of MAXCLIQUE1}
  procedure EXPAND(SUBG)
  begin
   if SUBG = ∅ then
     if |Q| > |Qmax| then Qmax := Q fi
   else u :=a vertex in SUBG
        that maximizes |SUBG ∩ Γ(u)|;
     Q := Q ∪ {u};
     SUBG_u := Γ(u) ∩ SUBG;
     if |Q| + |SUBG_u| > |Qmax| then
       if |Q| + Number(SUBG_u) > |Qmax| then
         EXPAND(SUBG_u) ;fi fi
     Q := Q − {u};
     EXT_u := SUBG − {u} − SUBG_u;
     sort vertices in EXT_u in a
     decreasing order w. r. t their degrees ;
     While |EXT_u| > 1 do
       v_i := EXT_u[1];
       U_i := Γ(v_i) ∩ EXT_u;
       SUBG_{v_i} := U_i ∪ (Γ(v_i) ∩ SUBG_u);
       if |Q| + 1 + |SUBG_{v_i}| > |Qmax| then
         if |Q|+1+Number(SUBG_u) > |Qmax|then
           Q := Q ∪ {v_i};
           EXPAND(SUBG_{v_i});
           Q := Q − {v_i}; fi
         EXT_u := EXT_u − {v_i};
         EXT_u[1] :=a maximum degree vertex
                  in EXT_u;
       else break; fi od fi
  end {of EXPAND}

  procedure Number(R)
  begin
    maxno := 0;
    C_1 := ∅;
    while R ≠ ∅ do
      p := the first vertex in R;
      k = 1;
      while C_k ∩ Γ(p) ≠ ∅ do
        k := k + 1; od
      if k > maxno then
        maxno := k;
        C_maxno := ∅; fi
```

$C_k := C_k \cap \{p\}$;
$R := R - \{p\}$; **od**
return *maxno*;
end {of Number}

図 1:アルゴリズム MAXCLIQUE1

2.2 アルゴリズム MIS

MIS[3] は文献 [6] の解析以前には最良の最大時間計算量評価が得られた多項式領域アルゴリズムである，MIS は最大独立節点集合問題に対するアルゴリズムであるが，この問題は最大クリーク抽出問題と双対な問題である．

このアルゴリズムは問題を連結グラフごとに分割を行い，再帰するたびに問題サイズを小さくする．このとき各ステップにおいて，次数が 4 以下の節点について独自の分枝限定を行っている．詳細は，[3] 参照．

しかしながら，極めて小さい次数の節点に関してのみ分枝限定が行われるため，枝密度が低くないグラフについてはあまり効果が期待できない．

3 計算機実験

3.1 実験対象および環境

MAXCLIQUE が実働上でも高速であることを示すために，計算機実験によって 2 節で述べた MAXCLIQUE，MAXCLIQUE1，MIS，MCQ についてを実装し，同じグラフに対して各アルゴリズムの実行時間および分枝数を測定し，比較実験を行った．ここでアルゴリズム MCQ[5] は再帰 1 回あたりの計算量は $O(n^2)$ であるものの計算量の理論解析は困難ではある。しかし，実働上には高速であることが示されている．ただし，MIS に関してはアルゴリズムの構造上，分枝数を測定しても意味はないので測定しなかった．

対象とするグラフは，ランダムグラフと DIMACS ベンチマーク用グラフである．ランダムグラフは，密度 p のグラフの場合，任意の 2 節点間に一様乱数を用いて確率 p で辺を張るグラフである．また，DIMACS ベンチマーク用グラフは DIMACS が提供している実問題に即したグラフである．ただし，ランダムグラフについてはそれぞれ異なる一様乱数を用いて作成したグラフ 10 個に対する平均値をとった．

・実行環境

CPU : Pentium4 3.6GHz
Memory : 2.0GB
OS : Linux
使用言語 : C 言語
コンパイラ: gcc -O2 (ver 3.3.2)

3.2 実験結果および考察

表 1，2 は，ランダムグラフおよび DIMACS ベンチマークグラフについて，実行時間, 分枝数を記載している．以降では，アルゴリズム名を MAX(MAXCLIQUE), MAX1(MAXCLIQUE1) と略す．また，ω は最大クリークのサイズ，n は節点数，p は枝密度を表す．

表 1，2 から，MAX は MIS に比べてほぼ全般的に高速であることが示された．しかしながら，一部のグラフについては，MAX は MIS よりも大幅に遅い結果となった．ランダムグラフに関して言えば，枝密度が高いグラフについてこのような結果となった．これは MIS が対象グラフの補グラフについて実行しているので，低次数の節点が多く存在し，分枝限定の効果が発揮されたためであると考えられる．

MAX に近似彩色による分枝限定を適用した MAX1 であるが，MAX と比較すると実行時間，分枝数ともに大幅な改善が行われていることがわかる．このことから，近似彩色による分枝限定が最大クリーク抽出問題に関して極めて有効であることも示された．

実働上にのみ注視すると，MCQ の方がに一部のグラフを除き，より高速であることがわかる．

4 おわりに

本稿では，理論的に優れたアルゴリズム MAXCLIQUE が実働上においても，理論評価のある他のアルゴリズム [3] よりも広い範囲にわたって高速であることを示した．また近似彩色による分枝限定が，クリーク抽出問題に関して有効であることも再確認した．

参考文献

[1] R. E. Tarjan, A. E. Trojanowski, "Finding a maximum independent set, "SIAM Journal on Computing 6, pp.537-546 (1977).

[2] M. Shindo, E. Tomita, "A simple algorithm for finding a maximum clique and its worst-case time complexity, " Systems and Computing in Japan 21, pp.1-13 (1990).

[3] F. V. Fomin, F. Grandoni, D. Kratsch, "Measure and conquer : A simple $O(2^{0.288n})$ independent set algorithm, " Proc. Symp. on Disrete Algorithms, pp.18-25 (2006).

[4] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments, "Theoret. Comput. Sci. 363, pp.28-42 (2006).

[5] E. Tomita, T. Kameda, "An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments, " Journal of Global Optimization 37, pp.95-111 (2007).

[6] 中西裕陽，富田悦次，"最大クリークを抽出する時間計算量 $O(2^{0.19669n})$-時間の多項式領域アルゴリズム，"情報処理学会技術研究報告，2007-AL-115，pp.17-24 (2007).

表 1: ランダムグラフに対する結果

			実行時間 [sec]				分枝数		
n	p	ω	MCQ	MAX1	MAX	MIS	MCQ	MAX1	MAX
1000	0.2	7-8	0.13	0.32	0.57	1,057	43,379	5,774	222,331
3000	0.2	9-9	13	37	61	$> 10^5$	2,769,068	228,685	15,526,492
500	0.4	11-11	0.37	1.13	3.42	216.97	124,058	19,229	1,253,660
500	0.5	13-14	3.8	14.4	59.8	1,083	1,124,108	278,507	21,262,983
300	0.5	12-13	0.16	0.60	2.36	45.06	56,911	14,290	914,134
300	0.6	15-16	1.5	6.3	39.0	275.2	473,628	156,561	13,798,311
200	0.6	14-14	0.1	0.4	2.2	15.9	38,377	11,595	836,489
150	0.9	35-39	7	49	2,527	139	1,329,394	2,262,894	239,907,274

表 2: DIMACS ベンチマークグラフに対する結果

		実行時間 [sec]				分枝数		
グラフ名	ω	MCQ	MAX1	MAX	MIS	MCQ	MAX1	MAX
sanr200_0.7	18	0.59	2.82	28.78	95.02	183,528	82,193	9,593,784
sanr200_0.9	42	323	1,605	$> 10^5$	$> 10^5$	42,865,151	42,926,945	-
sanr400_0.5	13	0.92	3.42	14.09	262.37	300,173	71,508	5,143,557
sanr400_0.7	21	384	1,977	27,084	70,078	90,091,325	40,220,815	3,306,730,447
c-fat200-5	58	0.0014	0.0010	0.0069	0.3020	310	58	520
c-fat500-2	26	0.0015	0.0009	0.0034	22.5	546	26	771
c-fat500-5	64	0.004	0.002	0.011	15.300	622	64	622
c-fat500-10	126	0.014	0.009	0.113	10.720	746	126	2,203
brock200_1	21	1.7	9.0	145.5	294.1	4,319	261,013	44,445,940
brock200_2	12	0.01	0.04	0.19	4.45	4,319	847	66,204
brock200_3	15	0.06	0.44	2.37	16.84	16,523	15,018	942,071
brock200_4	17	0.20	0.89	7.73	42.96	63,077	23,214	2,534,772
p_hat300-1	8	0.004	0.012	0.027	7.150	2,215	285	11,820
p_hat300-2	25	0.049	0.551	11.100	16.570	10,034	12,288	2,682,250
p_hat300-3	36	16.959	125.73	13,722	2,187	2,472,735	2,421,496	2,577,201,717
p_hat500-1	9	0.031	0.125	0.294	61.180	11,284	2,354	115,009
p_hat500-2	36	4.07	35.67	3,867	1,120	543,524	405,824	603,341,793
p_hat700-1	11	0.113	0.429	1.195	254.640	34,421	5,270	373,581
p_hat1000-1	10	0.6	2.7	6.6	1192.9	202,680	36,320	2,309,070
p_hat1500-1	12	5.2	23.8	72.8	8306.5	1,283,445	197,520	20,072,563
san200_0.7_1	30	0.01	0.05	26,726.22	14.07	1,243	673	519,774,200
san200_0.7_2	18	0.007	0.213	179,639.470	3.730	1,580	2,952	4,289,016,279
san400_0.5_1	13	0.03	0.25	2,994.19	13.21	3,409	898	725,114,909
hamming8-2	128	0.007	0.010	$> 10^5$	$> 10^5$	255	128	-
hamming8-4	16	0.23	0.64	12.77	120.95	41,604	9,047	3,843,396
johnson16-2-4	8	0.21	1.17	1.81	24.22	430,131	307,806	3,120,413
keller4	11	0.03	0.37	1.86	4.32	12,751	14,548	1,189,476