2004－MPS－52　(11)
2004／12／20

極大クリーク全列挙アルゴリズムの実験的比較評価

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

概要. グラフ中の極大クリークを全列挙するためのアルゴリズムとして，筆者らは節点数に関して最適オーダのアルゴリズム CLIQUES をこれまでに提唱してきている．本稿では，計算機実験によりその速度評価を行い，対象グラフが非常に疎であるか極大クリーク数が非常に少ない場合を除いた多くのグラフに対して実際にもそれが高速であることを示す.

Experimental Comparison of Algorithms for Generating All Maximal Cliques

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

Abstract. We have previously presented an algorithm CLIQUES for generating all maximal cliques that is proved to be optimal as a function of the number of vertices. We show here by computational experiments that it also runs very fast in practice for many graphs except for those which are very sparse or have very few maximal cliques.

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を抽出すること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などに対して多くの様々な応用がある [1]，[2]，[3]．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検証されてきた．Bron と Kerbosch[4] は，ランダムグラフについてそのアルゴリズムの極大クリーク当たりの計算時間がグラフの大きさにほとんど依存せず，また節点数 n の *Moon and Moser* グラフに対する全体の計算時間がほぼ $(3.14)^{n/3}$ に比例するとの実験結果と共に全極大クリークを抽出するアルゴリズムを発表した．Tsukiyama ら [5] は $O(nm\mu)$ の時間で G 中の全極大独立節点集合を列挙するアルゴリズムを考案した．ここで n，m，μ はグラフ G 中の節点，枝，極大独立節点集合の数である．更に Chiba と Nishizeki[6] は Tsukiyama らのアルゴリズムを改良し，G の極大クリークを $O(a(G)m\mu)$ の時間で列挙するより効率的なアルゴリズム CLIQUE を発表した．ここで $a(G)$ は G の樹化数 (arboricity) であり連結グラフ G において $a(G) \leq O(m^{1/2})$ となる．また，μ は G 中の極大クリークの数である.

極く最近，Makino と Uno [7] は Tsukiyama らのアルゴリズムを基にした新しいアルゴリズムを幾つか発表した．その内，同文献 [7] の Theorem 2 の結果を具体的に実現するアルゴリズムは文献 [8] の Enum_Maximal_Clique (EMC と略記する) として提唱されているものであり，G の全ての極大クリークを $O(\Delta^4\mu)$ の時間で列挙する．ここで Δ は G の最大次数である．彼らは計算機実験によりそのアルゴリズム EMC が疎なグラフにおいて Tsukiyama ら [5] のアルゴリズムより大幅に高速であることを示した.

筆者らは Bron と Kerbosch のアルゴリズム [4] における枝刈り法を援用した，深さ優先探索により G 中の全極大クリークを抽出するアルゴリズム CLIQUES を発表している [9]，[10]．このアルゴリズムは全ての極大クリークを木の形で出力する．その最大計算時間は $O(3^{n/3})$ であることが証明されているが，これは n の関数としては最適なものであり，その結果はクリーク問題に関する著名なサーベイ論文 [11] においても紹介されている．本稿では，このアルゴリズムが非常に疎であるか極大クリーク数が非常に少ないグラフを除いて実際に高速であることを計算機実験により示す.

2　表記法と諸定義

[1] 本稿を通して，節点の有限集合 V と，枝と呼ばれる異なる節点の対 (v, w) の有限集合 E よりなる単純無向グラフ $G = (V, E)$ について議論する．もし $(v, w) \in E$ であれば v と w は隣接していると言う．$\overline{E} = \{(v, w) \in V \times V \mid v \neq w$ 及び $(v, w) \notin E\}$ より

なる $\overline{G} = (V, \overline{E})$ を G の補グラフと言う．

[2] 節点 $v \in V$ について，$\Gamma(v)$ を $G = (V, E)$ 中で v に隣接している全ての節点の集合とする．即ち $\Gamma(v) = \{w \in V \mid (v, w) \in E\}$ ($\ni v$).

[3] 部分節点集合 $W \subseteq V$ について，$E(W) = \{(v, w) \in W \times W \mid (v, w) \in E\}$ よりなる $G(W) = (W, E(W))$ を W によって誘導される $G = (V, E)$ の部分グラフと呼ぶ．節点集合 W について，$\mid W \mid$ は W の要素数を表す．

[4] 部分節点集合 $Q \subseteq V$ が与えられた時，全ての $v \neq w$ である $v, w \in Q$ について $(v, w) \in E$ であれば $G(Q)$ は完全であると言う．この場合，単に Q は完全部分グラフであるとも言う．完全部分グラフはクリークとも呼ぶ．あるクリークが他のクリークの真部分グラフでないならばそれを極大クリークと呼ぶ．もし全ての $v, w \in W$ について $(v, w) \notin E$ であるならば部分節点集合 $W \subseteq V$ は独立であると言う．ここで，$Q \subseteq V$ が補グラフ $\overline{G}$ の極大独立節点集合である時，かつその時に限り Q は G の極大クリークである．

3 CLIQUES[9],[10]

CLIQUES は Bron と Kerbosch [4] のアルゴリズムにおける枝刈り法を援用したアルゴリズムであり，深さ優先探索によって与えられたグラフ $G = (V, E)$ 中の全ての極大クリークを抽出し，木の形で出力する．まず，全節点集合 V を節点集合 $SUBG$ 及び節点集合 $CAND$ の初期値として与え，再帰的処理 EXPAND($SUBG, CAND$) を実行する．EXPAND($SUBG, CAND$) は以下のようにして極大クリークを抽出する．もし $SUBG = \emptyset$ ならば “clique,” と出力する．そうでないならば以下のような操作を行う．まず $u \in SUBG$ で，$|CAND \cap \Gamma(u)|$ を最大にする節点 u を選ぶ．次に $CAND - \Gamma(u)$ 中の節点 q について，“q ,” と出力した後に，節点集合 $SUBG_q := SUBG \cap \Gamma(q)$ ，$CAND_q := CAND \cap \Gamma(q)$ を求め，EXPAND($SUBG_q, CAND_q$) を実行した後，$CAND := CAND - \{q\}$ としてから “back,” と出力する．これらの操作を $CAND - \Gamma(u) = \emptyset$ となるまで繰り返す．以上の手続きにより CLIQUES はグラフ G 中の全極大クリークを，$O(3^{n/3})$ の最大計算時間で列挙する．なお、節点数 n のグラフ中には最大で $3^{n/3}$ 個の極大クリークが存在するので [12]，これは n の関数としては最適なものとなる．ここで，出力される全極大クリークは木の形で表される．

4 計算機実験

アルゴリズム CLIQUES と，その比較対象として CLIQUE [6]，AllMaxCliques (以下 AMC と略記) 及び AllMaxCliques* (以下 AMC*と略記) [7] を C 言語で実働化し，同一の環境で計算機実験を行い性能を比較した．使用した計算機は Pemtium4 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．使用コンパイラ及びフラグは gcc -O2 である．本実験において，全ての極大クリークを実際に列挙する負荷を取り除くために，CLIQUES 以外のアルゴリズムにおいては極大クリークの個数の値のみを出力するように変更した．文献 [7] の AMC*は入力されたグラフ G の全節点を次数が Δ^* 以下の節点と Δ^* より大きい節点とに分けて処理を行うアルゴリズムである．本実験では $\Delta^* = n/100$ とした [8]．これは一般に未知のグラフに対し最適な Δ^* の値を選ぶことは難しく，また，色々Δ^* の値を変えて実験を行った結果，$\Delta^* = n/100$ とした時が最も多くの種類のグラフで高速となったからである．また，$\Delta^* = n/100$ とした場合の AMC*は，その基本としている EMC よりも一般に非常に高速となることを確認している．

表 1 に節点数 n と枝の存在確率 p の組み合わせで作成したランダムグラフに対する実行時間を示す (太字は最速箇所)．ここにおけるグラフは n 個の節点に対して各節点の間に確率 p で枝を張ったものである．表中において ♯cliques とはグラフ中の極大クリークの個数を表す．また，/cliques は CLIQUES による極大クリーク 10^6 個当たりの抽出時間を表す．表 2 には Moon and Moser グラフ [12] と DIMACS のベンチマークグラフ [13] に対する実験結果を示す．表中において，M-M-n とは節点数 n の Moon and Moser グラフを表し，dens. はグラフの節点数 n に対し (グラフ中の枝の数)/$\{n(n-1)/2\}$ として求めた枝密度を表す．また，[7] の実験結果と比較するために，[7] で提唱された疎で局所的にランダムなグラフに対しても実験を行った．これは節点数 n とパラメータ r に対し，グラフ中の節点 v_i と v_j が $i + n - j (\mathrm{mod}\ n) \leq r$ または $j + n - i (\mathrm{mod}\ n) \leq r$ を満たしている場合にのみこれら 2 節点間に確率 1/2 で枝を張るという方法で作成されたグラフである．表 3 には，$r = 10$，$r = 30$ とした場合の実行時間を示し，表 4 にはグラフの節点数を 10,000 に固定し，r の値を様々に変えた場合の実行時間を示す．

実験結果より，CLIQUES は CLIQUE よりも大幅に高速である．また，文献 [7] の 8 章の計算機実験結果より，CLIQUES は Tsukiyama らのアルゴリズムに対しても大幅に高速であると認められる．更に，CLIQUES は AMC 及び AMC*に対してほとんどのグラフで高速であったが，c-fat200-5，c-fat500-10 などのような極大クリークの個数が非常に少ないグラフでは AMC，AMC*の方が高速になり得る．さらに，

表 1: ランダムグラフに対する実行時間 [sec]

n	p	♯cliques	CLIQUE	AMC	AMC*	CLIQUES	/cliques
300	0.3	96,298	41.28	26.77	9.55	**0.14**	1.45
300	0.4	559,838	364.89	197.20	78.26	**0.88**	1.57
300	0.5	4,874,385	5,645.05	1,759.61	789.23	**8.54**	1.75
300	0.6	132,240,024	$> 24hrs$	24,104.49	12,937.56	**140.75**	1.06
300	0.7	3,356,452,714	$> 24hrs$	$> 24hrs$	$> 24hrs$	**6,279.51**	1.87
700	0.1	37,563	29.91	21.86	3.15	**0.051**	1.36
700	0.2	325,479	485.38	367.96	98.64	**0.51**	1.57
700	0.3	3,094,828	9,197.77	5,201.25	1,806.24	**5.42**	1.75
700	0.4	40,591,244	$> 24hrs$	$> 24hrs$	34,873.65	**84.05**	2.07
700	0.5	917,376,496	$> 24hrs$	$> 24hrs$	$> 24hrs$	**2,144.31**	2.34
1,000	0.1	99,062	179.80	143.03	19.39	**0.21**	2.12
1,000	0.2	1,183,584	3,750.59	4,486.30	829.52	**2.25**	1.90
1,000	0.3	15,362,096	$> 24hrs$	$> 24hrs$	20,615.89	**33.18**	2.16
2,000	0.1	747,300	6,384.56	10,149.20	665.59	**2.32**	3.10
3,000	0.001	4,610		1.86	**0.15**	0.32	69.41
3,000	0.003	13,319		12.89	1.03	**0.33**	24.78
3,000	0.005	21,287		33.65	2.87	**0.33**	15.50
3,000	0.01	37.862		114.47	5.62	**0.36**	9.51
3,000	0.1	2,945,211		$> 24hrs$	5,905.18	**10.85**	3.68
4,000	0.1	8,215,969		$> 24hrs$	29,228.07	**34.41**	4.19
5,000	0.001	12,527		16.81	**0.77**	2.02	161.25
5,000	0.003	36,538		132.98	7.50	**2.05**	56.11
5,000	0.005	57,501		350.48	21.18	**2.19**	38.09
5,000	0.01	96,312		1,258.21	43.08	**2.49**	25.85
5,000	0.1	18,483,855		$> 24hrs$	$> 24hrs$	**86.60**	4.67
10,000	0.0007	34,877		144.06	**6.37**	10.77	308.80
10,000	0.001	49,738		282.40	13.30	**10.82**	217.54
10,000	0.005	215,129		6,138.69	364.29	**11.74**	54.57
10,000	0.01	348,552		22,426.93	645.75	**14.78**	42.40
10,000	0.03	3,738,814		$> 24hrs$	11,315.03	**41.29**	11.04
10,000	0.05	12,139,182		$> 24hrs$	$> 24hrs$	**109.78**	9.04
10,000	0.1	229,786,397		$> 24hrs$	$> 24hrs$	**1,825.45**	7.94

表 2: Moon and Moser グラフ及び DIMACS のベンチマークグラフに対する実行時間 [sec]

グラフ名	n	dens.	♯cliques	CLIQUE	AMC	AMC*	CLIQUES	/cliques
M-M-48	48	0.957	43,046,721	5,057.65	153.21	224.41	**12.41**	0.29
M-M-51	51	0.960	129,140,163	16,532.50	496.76	740.57	**32.95**	0.26
M-M-60	60	0.966	3,486,784,401	$> 24hrs$	16,585.75	26,152.31	**894.90**	0.26
M-M-63	63	0.968	10,460,353,203	$> 24hrs$	47,817.37	$> 24hrs$	**2,666.90**	0.25
MANN_a9	45	0.927	590,887	52.64	2.24	2.93	**0.23**	0.39
brock200_2	200	0.496	431,586	181.42	75.16	35.91	**0.65**	1.51
brock200_3	200	0.605	4,595,644	3,689.62	684.23	437.22	**7.71**	1.68
c-fat200-1	200	0.077	37	0.025	0.0011	**0.00065**	0.0010	27.03
c-fat200-5	200	0.426	7	0.30	**0.0029**	0.0031	0.0054	771.43
c-fat500-1	500	0.036	80	0.20	0.0062	**0.0029**	0.0054	67.50
c-fat500-10	500	0.374	8	6.62	**0.025**	0.028	0.058	7,250.00
hamming6-2	64	0.905	1,281,402	301.00	11.97	16.98	**1.21**	0.94
hamming6-4	64	0.349	464	0.018	0.0086	0.0056	**0.00043**	0.93
johnson8-2-4	28	0.556	105	0.0019	0.00062	0.00052	**0.00012**	1.14
johnson8-4-4	70	0.768	114,690	13.85	1.82	1.95	**0.11**	0.96
johnson16-2-4	120	0.765	2,027,025	907.51	150.68	153.42	**4.38**	2.16
keller4	171	0.649	10,284,321	3,446.61	1,145.66	490.76	**4.98**	0.49
p_hat300-1	300	0.244	58,176	19.23	13.52	3.52	**0.070**	1.20
p_hat300-2	300	0.489	79,917,408	$> 24hrs$	16,035.71	4,130.20	**99.72**	1.25

表 3: 局所的ランダムグラフに対する実行時間 [sec]

(a) $r = 10$

n	dens.	♯cliques	CLIQUE	AMC	AMC*	CLIQUES	/cliques
1,000	0.010	4,487	2.38	0.45	0.04	**0.016**	3.57
2,000	0.0051	9,369	18.76	4.70	0.38	**0.075**	8.01
4,000	0.0025	19,166		31.48	1.55	**0.86**	44.87
6,000	0.0017	29,192		88.91	3.56	**3.30**	113.04
8,000	0.0013	39,179		160.16	**6.25**	**6.25**	159.52
10,000	0.0010	48,975		261.27	**9.51**	10.47	213.78

(b) $r = 30$

n	dens.	♯cliques	CLIQUE	AMC	AMC*	CLIQUES	/cliques
1,000	0.030	13,043	8.93	3.38	0.38	**0.025**	1.92
2,000	0.016	25,803	57.13	44.79	1.49	**0.10**	3.88
4,000	0.0075	53,059		243.66	9.31	**0.89**	16.77
10,000	0.0030	139,304		2,075.70	73.55	**10.03**	72.00

表 4: 節点数 10,000 の局所的ランダムグラフに対する実行時間 [sec]

r	dens.	♯cliques	AMC	AMC*	CLIQUES	/cliques
10	0.0010	48,975	261.27	**9.51**	10.47	213.78
20	0.0020	95,120	952.25	49.45	**10.19**	107.13
80	0.0080	386,360	14,448.21	431.20	**10.87**	28.13
160	0.016	1,408,360	$> 24hrs$	1,066.62	**16.85**	11.96
320	0.032	11,488,405	$> 24hrs$	15,655.05	**65.06**	5.66

AMC*は与えられたグラフが非常に疎である時に，特に Δ^* を個々のグラフに対し最適なものとなるように選べば CLIQUES よりも高速になり得る．また，文献 [7],[8] には，変形アルゴリズムによる大規模な現実問題への良好な適用結果も報告されている．なお，表中の/cliques の値より，CLIQUES はグラフの枝密度が大きくなっても極大クリーク当たりの計算時間が爆発的には増加しないことが確認できた．

5 まとめ

筆者らが既に提唱している極大クリーク全列挙アルゴリズム CLIQUES に対して計算機実験による評価を行い，その結果，当アルゴリズムは対象グラフが非常に疎であるかあるいは極大クリーク数が非常に少ない場合を除いた多くのグラフに対して高速であることを確認した．

謝辞. 本研究に先んじて CLIQUE と CLIQUES との実験的評価を行っていただいた元卒研生の木鉛登之氏，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] 宇野毅明, “効率的な列挙アルゴリズムの構築と利用 (3),” 人工知能学会誌, 18 巻 5 号, pp.586-591 (2003).

[2] M. Hattori, Y. Okuno, S. Goto, and M. Kanehisa, “Development of a chemical structure comparison method for integrated analysis of chemical and genomic information in the metabolic pathways,” J. Am. Chem. Soc. 125, pp.11853-11865 (2003).

[3] 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).

[4] C. Bron and J. Kerbosch, “Algorithm 457, Finding all cliques of an undirected graph,” Comm. ACM 16, pp.575-577 (1973).

[5] S. Tsukiyama, M. ide, H. Ariyoshi, and I. Shirakawa, “A new algorithm for generating all the maximal independent sets,” SIAM J. Comput 6, pp.505-517 (1977).

[6] N. Chiba and T. Nishizeki, “Arboricity and subgraph listing algorithms,” SIAM J. Comput. 14, pp.210-223 (1985).

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

[8] 宇野毅明, “大規模グラフに対する高速クリーク列挙アルゴリズム,” 信学技報, COMP2003-8, pp.55-62 (2003).

[9] E. Tomita, A. Tanaka, and H. Takahashi, “An optimal algorithm for finding all the cliques,” IPSJ SIG Technical Report, 1989-AL-12, pp.91-98 (1989).

[10] E. Tomita, A. Tanaka, and H. Takahashi, “The worst-case time complexity for generating all maximal cliques,” COCOON 2004, LNCS 3106, pp.161-170 (2004).

[11] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo, “ The maximum clique problem, ” In: D.-Z. Du and P. M. Pardalos (eds.), Handbook of Combinatorial Optimization, Supplement vol. A, Kluwer Academic Publishers, pp.1–74 (1999).

[12] J. W. Moon and L. Moser, “On cliques in graphs,” Israel J. Math. 3, pp.23-28 (1965).

[13] D. S. Johnson and M. A. Trick(Eds), “Cliques, Coloring, and Satisfiability,” DIMACS Series in Descrete Mathematics and Theoretical Computer Science, vol. 26, American Mathematical Society (1996).