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

2007－MPS－67　(6)
2007－BIO－11　(6)
2007／12／20

[Invited Lecture]

The Maximum Clique Problem and Its Applications

Etsuji TOMITA†

Abstract. We show some recent results in the Advanced Algorithms Research Laboratory at the University of Electro-Communications on the maximum clique problem and its applications. These are parts of our various work in the AARL at UEC with which the author is involved together with his colleagues and his students.

1 The Maximum Clique Problem

Many problems can be formulated as graphs where a graph consists of a set of vertices and a set of edges, in which the vertices stand for objects in question and the edges stand for some relations among the objects. A *clique* is a subgraph in which all vertices are pairwise adjacent[6]. Hence, a clique represents a set of objects in which every pair is related. In addition, a maximum clique of the direct product of two graphs represents a maximum matching in the two graphs[6]. Therefore, a maximum clique and maximal cliques play an important role and have received considerable attention[11, 6].

However, the so called *maximum clique problem* is considered to be very hard to solve, that is, it is proved to be an *NP-complete* problem. Nevertheless, many researchers including the author are engaged in devising as fast algorithms as possible for finding a maximum clique and generating all maximal cliques, because of their importance in practice.

1.1 Fast Algorithms for Finding a Maximum Clique

Recently, we presented a simple and fast branch-and-bound algorithm **MCQ**[28] for finding a maximum clique, and we improved it to get a new algorithm **MCR**[30], primarily by introducing more appropriate sorting of vertices at the beginning. MCR in turn was improved to a more efficient algorithm MCR-Re[31], by employing sophisticated approximate coloring and sorting of vertices. In addition, we have improved MCR-Re to have an algorithm **MCS** (previously called MCS_0) by localizing the memory usage in order to make more effective use of cache memory[22].

A comparison of MCQ, MCR, and MCS is shown in Table 1, where the branches correspond to the extent of search spaces[28, 30] and ω is the size of a maximum clique in a given graph. Some computational time comparisons with other algorithms are shown in Tables 2 and 3. In Table 2, n is the number of vertices and p the edge probability. The computer used in these experiments is a Pentium4 3.6GHz CPU operating on Linux[31]. It is confirmed that MCS is by far the fastest among all the presently existing algorithms for almost all cases.

We have also developed algorithms for weighted graphs[27, 24].

1.2 Algorithms for Generating Maximal Cliques

In addition to finding a maximum clique, generating all maximal cliques is required in many diverse applications such as clustering, data mining and others[6, 13]. We proposed an algorithm **CLIQUES**[29] for generating all maximal cliques.

A part of its computational time comparisons with other algorithms is shown in Table 4, where ♯cliques is the number of maximal cliques. The computer used in this experiment is a Pentium4 2.2GHz CPU operating on Linux[29]. CLIQUES is very fast and space efficient.

Some variations of CLIQUES have also been developed[14, 15].

1.3 Theoretical Analyses

We proved that the worst-case time complexity of CLIQUES is $O(3^{n/3}) = O(2^{0.528n})$ for an $n-$vertex graph, and that is *optimal* with respect to n.

Steady improvements have been made to the time complexity for finding a maximum clique in an n-vertex graph in polynomial-space from $O(2^{0.333n})$[26] to $O(2^{0.288n})$[9] in the last almost 30 years. We have remarkably improved this complexity to $O(2^{0.19669n})$[16] by an algorithm that is based on CLIQUES[29, 20]. Our algorithm is also fast in practice[20, 25]. Further theoretical analysis is in progress.

†Department of Information and Communication Engineering, *and* The Advanced Algorithms Research Laboratory, The University of Electro-Communications, Tokyo 182-8585, Japan. E-mail: `tomita@ice.uec.ac.jp`

Table 1: Comparison of MCQ, MCR, and MCS

Graph		CPU time [sec]			branches $\times 10^{-3}$		
Name	ω	MCQ	MCR	**MCS**	MCQ	MCR	**MCS**
brock400_2	29	748	742	297	116,224	116,328	33,513
brock400_4	33	680	639	248	118,855	114,925	30,855
MANN_a27	126	2.61	2.54	0.78	38	38	9
MANN_a45	345	2,775	3,090	281	2,852	2,952	225
p_hat300-3	36	17	11	3	2,473	1,546	235
p_hat500-3	50	2,895	1,788	150	237,077	138,300	7,923
p_hat700-3	62	122,264	68,187	2,392	7,046,183	3,733,665	88,168
p_hat1000-2	46	2,764	2,434	221	221,797	197,147	12,618
san200_0.9_3	44	10.59	0.16	0.06	1,182	22	6
san400_0.9_1	100	32.8	3.4	0.1	708	74	2
sanr200_0.9	42	322	289	41	42,865	40,470	3,471

Table 2: CPU time [sec] for random graphs

Graph			dfmax [11]	**MCS** *(Ours)*	New [18]	COCR [19]
n	p	ω				
100	0.6	11-13	0.0041	**0.0016**	0.0022	0.092
	0.7	14-16	0.018	**0.0036**	0.0067	0.12
	0.8	19-21	0.14	● **0.0078**	0.065	0.15
	0.9	29-32	3.67	★ **0.013**	0.66	0.20
	0.95	39-48	23.74	★ **0.0028**	0.20	
	0.98	56-68	26.54	★★★ **0.00087**		
150	0.7	16-18	0.36	● **0.047**		0.33
	0.8	23	6.88	○ **0.23**		0.75
	0.9	36-39	1,058.96	**1.01**		1.16
	0.95	50-59	37,436.79	★★★ **0.35**		
	0.98	73-85	$> 10^5$	★★★ **0.0061**		
200	0.5	11-12	0.038	**0.015**	0.020	0.25
	0.6	14	0.29	○ **0.072**	0.17	0.52
	0.7	18-19	3.85	○ **0.41**	3.02	1.65
	0.8	24-27	192.68	**4.48**	147.29	8.69
	0.9	40-44	$> 10^5$	73.62		○ **36.79**
	0.95	58-66	$> 10^5$	★★★ **58.83**		
	0.98	90-103	$> 10^5$	★★★ **0.21**		
300	0.5	12-13	0.36	**0.13**	0.20	1.13
	0.6	15-16	4.88	○ **0.99**	3.50	4.98
	0.7	19-21	144.11	★ **12.00**	121.02	
	0.8	28-29	26,235.96	★ **393.57**		
	0.9	49	$> 10^5$	**79,628.80**		
500	0.5	13-14	8.99	○ **2.79**	7.25	17.43
	0.6	17	242.29	○ **40.70**	183.28	
	0.7	22-23	24,998.42	★ **1,538.74**		
	0.75	26	$> 10^5$	○ **20,403.68**		
1,000	0.3	9-10	1.98	**1.15**	1.64	
	0.4	12	33.28	**13.25**	23.19	
	0.5	15	1,107.70	○ **290.03**		
	0.6	20	$> 10^5$	● **13,554.05**		
5,000	0.1	7	6.29	**3.32**		
10,000	0.1	7-8	137.05	○ **59.55**		
15,000	0.1	8	792.57	○ **326.78**		

Entries indicated by ★★★, ★, ●, and ○ represent those that are more than or equal to 1000, 10, 5, and 2 times faster than all the others confirmed within the time limits in the same row, respectively.

Table 3: CPU time [sec] for DIMACS benchmark graphs

Graph Name	ω	dfmax [11]	**MCS** (*Ours*)	New [18]	χ + DF [8]	COCR [19]	MIPO [5]	Target/5 [21]
brock200_1	21	14.53	★ **0.86**	12.12	68.70			69.80
brock200_4	17	0.90	**0.14**	0.22	6.04	0.91		3.60
brock400_1	27	22,051	★ **693**		>10,640			>4,320
brock400_2	29	13,519	★ **297**		>10,640	>415		>4,320
brock400_3	31	14,795	★ **468**		>10,640			>4,320
brock400_4	33	10,633	★ **248**		>10,640	>415		>4,320
brock800_1	23	$> 10^5$	★ **9,347**		>10,640			
brock800_2	24	$> 10^5$	★ **8,368**		>10,640	>415		
brock800_3	25	91,031	★ **5,755**		>10,640			
brock800_4	26	78,737	★ **3,997**		>10,640	>415		
c-fat500-10	126	$> 10^5$	0.026	0.016	**0.015**			
hamming8-4	16	1.85	0.20	**0.19**	4.51	1.00	29.13	
hamming10-2	512	$> 10^5$	○ **0.19**	0.56	3.81			
johnson16-2-4	8	0.75	0.14	0.060	5.88		★ **0.0017**	
MANN_a27	126	$> 10^5$	○ **0.78**	>2,232	7,647	2.75		
MANN_a45	345	$> 10^5$	★★ **281**		>10,640			
p_hat300-2	25	0.63	★ **0.018**	0.22	2.23	0.61		
p_hat300-3	36	780	○ **2.55**		633	5.39		
p_hat500-1	9	0.051	**0.030**	0.065	0.44			0.20
p_hat500-2	36	133	★★ **0.74**	95.71	151			>4,320
p_hat500-3	50	$> 10^5$	★★ **150**		>10,640			>4,320
p_hat700-1	11	0.20	**0.10**	0.15	1.98	2.74		1.40
p_hat700-2	44	5,300	○ **5.60**		1,542	25.44		>4,320
p_hat700-3	62	$> 10^5$	★ **2,392**		>10,640	>415		>4,320
p_hat1000-1	10	1.05	○ **0.49**	1.30	12.14			
p_hat1000-2	46	$> 10^5$	★★ **221**		>10,640			
san200_0.9_1	70	$> 10^5$	0.22	0.060	46.27		**0.050**	4.60
san200_0.9_2	60	$> 10^5$	0.41	0.96	1,427		○ **0.15**	65.60
san200_0.9_3	44	42,643	★★ **0.063**		144		15.15	>4,320
san400_0.5_1	13	433	0.020	○ **0.0067**	4.98		85.44	0.80
san400_0.7_1	40	$> 10^5$	★ **0.54**	>2,232	315			24.40
san400_0.7_2	30	$> 10^5$	★★ **0.13**	113	118		505	113.20
san400_0.7_3	22	$> 10^5$	★★ **1.44**		456			>4,320
san400_0.9_1	100	$> 10^5$	★★★ **0.12**		5,335			>4,320
san1000	15	$> 10^5$	2.17	★ **0.11**	2,249			
sanr200_0.9	42	86,954	★★★ **41**		>10,640			
sanr400_0.5	13	2.12	○ **0.72**	1.48	17.06			
gen200_p0.9_44	44	48,262	○ **0.47**			1.88	13.01	
gen200_p0.9_55	55	9,281	1.23			0.96	● **0.19**	
gen400_p0.9_55	55	$> 10^6$	★ **58,502**					
C125.9	34	50.05	● **0.060**			0.56	46.6	
C250.9	44	$> 10^6$	★★ **3,257**					

Entries indicated by ★★★, ★★, ★, ●, and ○ represent those that are more than or equal to 1000, 100, 10, 5, and 2 times faster than all the others confirmed within the time limits in the same row, respectively.

Table 4: CPU time [sec] for random graphs

Graph Name	♯cliques	**CLIQUES** [29]	AMC [13]	AMC* [13]
r1000.1	118,325	0.2	143.1	19.4
r1000.2	1,183,584	2	4,486	830
r3000.1	2,945,211	11	>86,400	5,905
r5000.1	18,483,855	87	>86,400	>86,400

2 Applications

The above algorithms and their extensions are being successfully applied to many problems. These include the followings:

- Clustering[33],
- Bioinformatics[2], [3], [4], [7], [1],
- Image processing[10],
- Design of quantum circuits[17],
- Design of DNA and RNA sequences for biomolecular computation[12], [23].

Parallel processing is also under study[32] in order to solve large practical problems.

Acknowledgment

The author would like to express his gratitude to his colleagues and students for their collaborative studies. This research was partially supported by Grants-in-Aid for Scientific Research Nos. 13680435, 16300001, 19500010, and others from the MEXT, Japan.

References

[1] T. Akutsu, M. Hayashida, D. K. C. Bahadur, E. Tomita, J. Suzuki, K. Horimoto. Dynamic programming and clique based approaches for protein threading with profiles and constraints. *IEICE Trans.*, E89-A: 1215–1222, 2006.

[2] D. K. C. Bahadur, T. Akutsu, E. Tomita, T. Seki, A. Fujiyama. Point matching under non-uniform distortions and protein side chain packing based on an efficient maximum clique algorithm. *Genome Inform.*, 13: 143–152, 2002.

[3] D. K. C. Bahadur, E. Tomita, J. Suzuki, K. Horimoto, T. Akutsu. Protein side-chain packing problem: A maximum edge-weight clique algorithmic approach. *J. Bioinform. Comput. Biology*, 3: 103–126, 2005.

[4] D. K. C. Bahadur, E. Tomita, J. Suzuki, K. Horimoto, T. Akutsu. Protein threading with profiles and distance constraints using clique based algorithms. *J. Bioinform. Comput. Biology*, 4: 19–42, 2006.

[5] E. Balas, S. Ceria, G. Cornuéjols, G. Pataki. Polyhedral methods for the maximum clique problem. In [11]: 11–28, 1996.

[6] I. M. Bomze, M. Budinich, P. M. Pardalos, M. Pelillo. The maximum clique problem,. In: D. -Z Du, P. M. Pardalos(Eds.). *Handbook of Comb. Optim.*, Suppl. vol. A, Kluwer Acad. Publ.: 1–74, 1999.

[7] J. B. Brown, D. K. C. Bahadur, E. Tomita, T. Akutsu. Multiple methods for protein side chain packing using maximum weight cliques. *Genome Inform.*, 17: 3–12, 2006.

[8] T. Fahle. Simple and fast: Improving a branch-and-bound algorithm for maximum clique. *ESA 2002, LNCS 2461*: 485–498, 2002.

[9] F. V. Fomin, F. Grandoni, D. Kratsch. Measure and conquer: A simple $O(2^{0.288n})$ independent set algorithm. *Proc. SODA 2006*: 18–25, 2006.

[10] K. Hotta, E. Tomita, H. Takahashi. A view-invariant human face detection method based on maximum cliques. *Trans. IPSJ*, 44, SIG14(TOM9): 57–70, 2003.

[11] D. S. Johnson, M. A. Trick (Eds.). Cliques, Coloring, and Satisfiability, *DIMACS Series in DMTCS*, vol.26, Amer. Math. Soc.: 1996.

[12] S. Kobayashi, T. Kondo, K. Okuda, E. Tomita. Extracting globally structure free sequences by local structure freeness. *Proc. DNA 9*: 206, 2003.

[13] K. Makino, T. Uno. New algorithms for enumerating all maximal cliques. *SWAT 2004, LNCS 3111*: 260–272, 2004.

[14] T. Nakagawa, E. Tomita. An efficient algorithm for generating large maximal cliques. *Tech. Rep. IPSJ SIG*, 2006-MPS-57: 49–52, 2005.

[15] T. Nakagawa, E. Tomita. An algorithm for generating all maximal bipartite cliques based on CLIQUES that generates all maximal cliques. *Tech. Rep. IPSJ SIG*, 2006-MPS-62: 73–76, 2006.

[16] H. Nakanishi, E. Tomita. An $O(2^{0.19669n})$-time and polynomial-space algorithm for finding a maximum clique *Tech. Rep. IPSJ SIG*, 2007-AL-115: 17–24, 2007.

[17] Y. Nakui, T. Nishino, E. Tomita, T. Nakamura. On the minimization of the quantum circuit depth based on a maximum clique with maximum vertex weight. *Tech. Rep. RIMS*, 1325, Kyoto Univ.: 45-50, 2003.

[18] P. R. J. Östergård. A fast algorithm for the maximum clique problem. *Disc. Appl. Math.*, 120: 197–207, 2002.

[19] E. C. Sewell. A branch and bound algorithm for the stability number of a sparse graph. *INFORMS J. Comput.*, 10: 438–447, 1998.

[20] M. Shindo, E. Tomita. A simple algorithm for finding a maximum clique and its worst-case time complexity. *Syst. Comput. in Japan*, 21: 1–13, 1990.

[21] V. Stix. Target-oriented branch and bound method for global optimization. *J. Global Optim.*, 26: 261–277, 2003.

[22] Y. Sutani, T. Higashi, E. Tomita, S. Takahashi. H. Nakatani. A faster branch-and-bound algorithm for finding a maximum clique. *Tech. Rep. IPSJ SIG*, 2006-AL-108: 79–86, 2006.

[23] Y. Sutani, E. Tomita, S. Kobayashi. A branch-and-bound algorithm for finding a maximum clique in a uniform hypergraph. *Tech. Rep. IPSJ SIG*, 2007-MPS-66: 111–114, 2007.

[24] J. Suzuki, E. Tomita, T. Seki. An algorithm for finding a maximum clique with maximum edge-weight and computational experiments. *Tech. Rep. IPSJ SIG*, 2002-MPS-42: 45–48, 2002.

[25] T. Tamada, E. Tomita, H. Nakanishi. Experimental evaluations of algorithms with known theoretical time-complexity for finding a maximum clique. *Tech. Rep. IPSJ SIG*, 2007-MPS-67, 2007.

[26] R. E. Tarjan, A.E. Trojanowski. Finding a maximum independent set. *SIAM J. Comput.*, 6: 537–546, 1977.

[27] E. Tomita, Y. Wakai, K. Imamatsu. An efficient algorithm for finding a maximum weight clique and its experimental evaluations. *Tech. Rep. IPSJ SIG*, 1999-MPS-26: 33–36, 1999.

[28] E. Tomita, T. Seki. An efficient branch-and-bound algorithm for finding a maximum clique. *DMTCS 2003, LNCS 2731*: 278–289, 2003.

[29] E. Tomita, A. Tanaka, H. Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *(Invited paper for the special issue on COCOON 2004). Theoret. Comput. Sci.*, 363: 28–42, 2006.

[30] E. Tomita, T. Kameda. An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. *J. Global Optim.*, 37: 95–111, 2007.

[31] E. Tomita, Y. Sutani, T. Higashi. A more efficient algorithm for finding a maximum clique with an improved approximate coloring. *Proc. PDPTA 2007*: 719–725, 2007.

[32] S. Urabe, E. Tomita. NetMCQ: A distribtted exact maximum clique solver. *Tech. Rep. IPSJ SIG*, 2007-MPS-67, 2007.

[33] C. Yonemori, T. Matsunaga, E. Tomita. An analysis of enterprise communities by cliques. *Tech. Rep. IPSJ SIG*, 2007-MPS-67, 2007.