Volume 7, Issue 3
June 2018

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

Maximum Clique Finder: MCF

[1]Atul Srivastava, [2]Anuradha Pillai, [3]Dimple Juneja Gupta

[1] Department of Computer Engineering, Pranveer Singh Institute of Technology, Kanpur, India

[2]Department of Computer Engineering, YMCA University of Science & Technology, Faridabad, India

[3]Department of Computer Applications, NIT, Kurukshetra, India

E-mail: [1]atul.nd2@gmail.com, [2]anuangra@yahoo.com, [3] dimplejunejagupta@gmail.com

ABSTRACT

In maximum clique problem, it is desired to find maximum number of vertices, any two of which are adjacent. The maximum clique problem falls into category of NP-hard problems. It is, therefore, often avoided to detect maximum clique by practitioners in many applications despite the fact that it has significant applications in the field of information retrieval, data mining, network analysis etc. Community Detection in social networks is one of the recent trends in computer science. Maximum Clique and community in social networks have overlapping definitions in respective domains. Thus problem of community detection in social networks reduces to finding cliques in graphs, provided social networks are represented as graphs. Several exact algorithms to find maximum clique already exist in literature that promise acceptable runtimes on certain graphs. But problem arises when these algorithms are applied on real world graphs which are massive in size. In this work, a novel branch and bound exact algorithm to find maximum clique, Maximum Clique Finder (MCF) has been presented with new pruning steps. This algorithm has been tested on real world graphs and DIMACS benchmark graphs, where it exhibits runtimes several times better than other existing algorithms and it performs notably well on large sparse real world graphs.

Keywords: *Community Detection in Social Networks, Information Retrieval, Data Mining, Graphs, NP-Hard problems.*

1. INTRODUCTION

An undirected graph is denoted by G = (V, E), where V is set of vertices and E is set of edges. A clique in the graph is set of vertices in which any two vertices are adjacent to each other. Two vertices are adjacent to each other if there is an edge between them. Therefore clique is a complete sub-graph of a given undirected graph. Finding the clique that has maximum number of vertices in a graph is called maximum clique problem [1].

There is a wide range of prominent applications of maximum clique problem such as community detection in networks [2, 3, 4], data mining in biometrics [5], information retrieval [6], data mining [7], symptoms correlation based disease classification [8], computer vision [9], coding theory [10], and pattern recognition [11]. Many more applications are listed in [12, 13].

Finding maximum clique is an NP-Hard problem [14]. An independent set is set of vertices in which no two vertices are adjacent. Finding the largest such set in a graph is called maximum independent set problem. A problem similar to this is vertex cover problem. A vertex cover is set of vertices which covers all the edges in the graph. To find smallest such set in the graph is called maximum vertex cover problem. The maximum clique problem is computationally equivalent to these two problems. Since all of these problems are NP-Hard problems, no polynomial time exact algorithm is expected to be found. Nevertheless, maximum clique problem finds several significant applications in prominent fields of computer science, it is of great interest to try to develop fast and exact algorithms. Almost every exact algorithm employs branch and bound approach which continuously optimizes the search of solution by discarding (pruning) the branches which will not lead to solutions any better than previously acquired solutions.

Carraghan and Pardalos [15] presented a simple and effective branch and bound algorithm for maximum clique problem. Ostergard [16] proposed an improvement to this algorithm at computation level. Tomita and Seki [17] used vertex coloring to compute upper bounds. Kone and Jenezic [27] provided improvement on this approach later. Other examples of branch and bound algorithms are Bomze et al., Segendo et al., and Babe and Tinhofer [6, 18, 19, 20]. Batageli and Pajek [21] compared different exact maximum clique algorithms.

This paper presents an algorithm to find maximum clique in an undirected graph with novel pruning steps. Some very promising exact algorithms and the development in finding maximum clique are briefly discussed in section 2. In section 3 our algorithm Maximum Clique Finder (MCF) is described in detail. Section 4 discusses implementation and result analysis. Section 4 concludes this article.

2. RELATED WORK

A simple approach to find maximum clique in an undirected simple graph G is to find all the cliques present in the graph and then select the largest one. But enumerating all the cliques requires infeasible time. Hence a simple algorithm is presented in [15], which reduces enumerations significantly. By pruning the fruitless branches, the search space is tremendously

Volume 7, Issue 3
June 2018

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

reduced. The algorithm finds largest clique containing vertex v_i at each step i by performing depth first search from vertex v_i. At each depth i, if the member of remaining vertices, which can possibly constitute a clique containing vertex v_i, is smaller than the size of largest clique found so far, the algorithm backtracks by pruning this branch of enumeration. Algorithm proposed in [16] incorporates an additional pruning in algorithm presented in [15] with the help of some auxiliary bookkeeping. Algorithm proposed in [16] is faster than algorithm proposed in [15] on random and DIMAX benchmark graphs [22]. However the order in which vertices are processed majorly affects the pruning strategy used in this algorithm.

Many algorithms to find maximum clique use vertex coloring to define upper bound on the maximum clique. MCQ algorithm [17] is one of the latest and popular methods which uses this idea. MaxCliqueDyn [23] is the improved version of MCQ with the variants MCQD and MCQD&CS. It uses computationally more expensive tighter upper bounds, which are applied on a part of search space. BBMC [24] is another enhanced version of MCQ which uses efficient methods to compute graph transitions and bounds. It uses bit strings to sort vertices in constant time.

3. MAXIMUM CLIQUE FINDER ALGORITHM (MCF)

In this section a new algorithm, MCF, is presented which overcomes the limitations of other algorithms mentioned earlier by the use of additional pruning strategies. Following notations are used in the algorithm. The graph G(V,E) contains n vertices as $\{v_1, v_2, \ldots\ldots v_n\}$. $Adj(v_i)$ is set of vertices adjacent to vertex v_i. And the cardinality of $Adj(v_i)$ i.e. degree of v_i is denoted by $deg(v_i)$. Degree of each vertex is computed once in the beginning of the algorithm.

Maximum clique in a graph can be found by enumerating the largest clique containing each vertex and then selecting the largest among these. Most significant point of our algorithm is that the search space is reduced by pruning the vertices which cannot form cliques larger than the current maximum clique in an incremental fashion. Algorithm 1 and algorithm 2 shown in figure 1 and figure 2 respectively outline this method. The variable max stores size of the largest clique found so far. Initially it is set as 0 or any other positive lower bound, if cliques smaller than the lower bound, are insignificant.

Maximum clique containing a vertex v_i cannot be larger than the degree of v_i, hence only the adjacent vertices of v_i are considered to obtain the largest clique containing v_i. The main procedure MCF therefore generates a set U⊆ $Adj(v_i)$ for each vertex v_i, which contains those neighbors of v_i which could survive the prunings. Subroutine Find_Clique is then invoked on U. Subroutine Find_Clique presented in Algorithm 2 enumerates every possible clique containing vertex v_i in a recursive fashion and returns the largest clique containing v_i. The size of clique found at any point of execution of Find_Clique is stored in 'size'. Initial value of size is set 1 as we start with a clique having just one vertex.

Algorithm 1: MCF(*G=(V,E)*)

```
/*max, size, C and M are global variables.*/
begin
        Sort all the vertices in V non-increasing order of
        degree./*pruning 1*/
  max ← 0;
  for i ← 1 to n /*to iterate n times*/
    size ← 1;
    C ← ϕ;
    v_i ← Select_First(V); /* v_i is highest degree node in V.*/
    V ← V\{v_i}; /*Pruning 2*/
    current_deg ← deg(v_i);
    C ← C ∪ {v_i}; /*C is current clique containing v_i*/
    U ← V ∩ Adj(v_i);
    Find_Clique(U,1,C);
    if size > max then
      max ← size;
      M ← C; /*M is max-clique found so far*/
    end if
    if max = current_deg then /*Pruning 3*/
      return M;
    end if
  end for
  return M;
end
```

Fig. 1 Algorithm for finding maximum clique

Algorithm 2: Find_Clique(*U, size, C*);

```
begin
  while U ≠ ϕ do
    if size + |U| ≤ max then /*Pruning 4*/
      return;
    end if
    select highest degree node u from U;
    U ← U\{u};
    C ← C ∪ {u};
    Find_Clique(U ∩ Adj(u),size+1,C);
  end while
end
```

Fig. 2 Subroutine for algorithm 1

There are four pruning steps in our algorithm. Probability of a vertex to be part of maximum clique is directly proportional to its degree. Therefore higher degree nodes have higher

Volume 7, Issue 3
June 2018

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

chances to be in max-clique. Pruning 1 emphasizes this idea as the vertices are already sorted in non-increasing order of their degrees. Lower degree nodes are pruned in very obvious manner. Pruning 2 avoids re-computation of already found cliques by including only those vertices in neighbors list of vertex v_i whose cliques are not yet found. Pruning 3 works on the same analogy used in pruning 1. If the largest clique found so far is of size k, then any vertex of degree smaller than or equal to k cannot form a clique of size greater than k. Therefore at this point all the vertices having degree less than or equal to k are ignored for further search of the largest clique. Pruning 4 says if all the vertices of U were added to get the clique, its size cannot be more than the size of largest clique found so far (max). Pruning 4 is most frequent pruning, pruning 1 and 2 are moderate and pruning 3 is used just once.

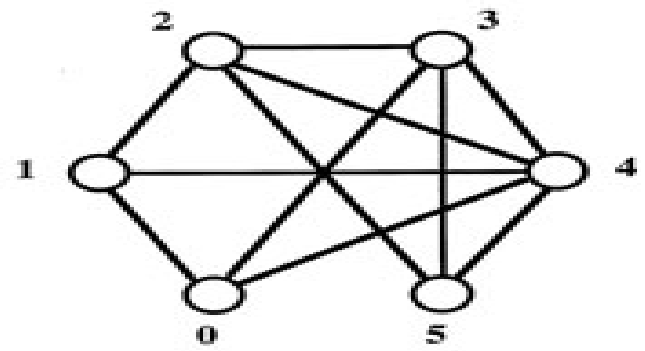

Fig. 3. Example Graph

Demonstration of MCF is presented as follows for graph shown in figure 3.

```
V = {4, 2, 3, 0, 1, 5} /*Sorted in
non-decreasing order of their degrees*/
max = 0
size = 1
C = {4} /*highest degree vertex 4 is
selected*/
Find_Clique({2, 3, 0, 1, 5}, 1, {4})
   C = {4, 2} /*vertex 2 is selected*/
   size = 2
   U = {3, 0, 1, 5}∩{1, 3, 4, 5}
   Find_Clique({3, 1, 5}, 2, {4, 2})
      C = {4, 2, 3} /*vertex 3 is selected*/
      size = 3
      U = {1, 5}∩{2, 0, 5, 4}
      Find_Clique({{5}, 3, {4, 2, 3})
         C = {4, 2, 3, 5} /*vertex 5 is
selected*/
         size=4
         U = ∅∩{2, 3, 4}
         Find_Clique(∅, 4, {4, 2, 3, 5})
            return./*Main Function MCF*/
max = 4
M = {4, 2, 3, 5}
max = deg(2) /*Pruning 3 prunes rest of the
vertices*/
return {4, 2, 3, 5} /*Max Clique*/
```

It is clearly evident from the above demonstration that, the maximum clique found by MCF is correct. MCF finds maximum clique in single iteration of the main algorithm, due to pruning 3 which is extremely effective and gives MCF upper hand as compared to other exact algorithms. We shall prove this fact with the help of experimental results in next section.

4. EXPERIMENTS AND RESULT ANALYSIS

In this section we present comparison of performance of our algorithm with other exact algorithms. Our experiments were performed on 64 bit windows 7 Home Basic with 2.3 GHz Intel Core i3 with 32 GB of main memory. Implementation is done in C compiled using NeuTroN DoS-C++ version 0.77.0.0. Our implementation uses a simple adjacency list representation for graph. This is done by maintaining a reference array of size |V|, which contains references for |V| lists of vertices, each corresponding to a particular vertex. Figure 4 shows our representation.

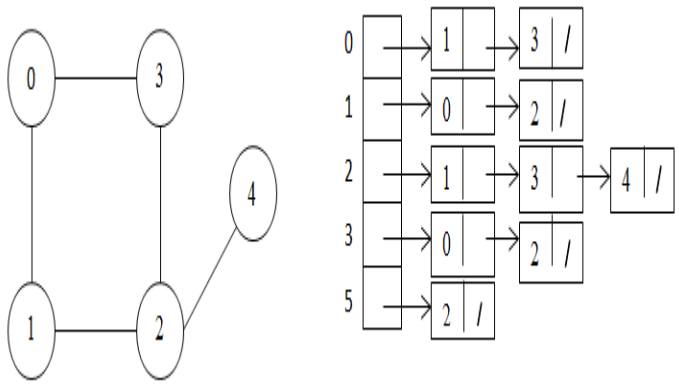

Fig. 4. The Adjacency List data structure.

We have considered graphs from two categories. First category includes graphs originated from real world applications. Table 1 gives brief description of those graphs. Second category includes graphs from DIMACS Implementation Challenge [22]. Table 2 represents structural properties of the graphs.

TABLE 1 DESCRIPTION OF REAL WORLD GRAPHS USED IN EXPERIMENT.

Graph	Description
Email_enron [25]	Communication network of E-mail Exchange
dictionary28 [20]	Network of words
code-mat-2003 [26]	Collaboration network of scientists
foldoc [27]	Dictionary for computing related terms
web-Google [28]	Web graph released as part of Google programming contest in 2002
soc-wiki-vote [29]	Wikipedia page vote network

Volume 7, Issue 3
June 2018

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

daysall [30]	Network of Reuter terror news obtained from CRA networks

TABLE 2 STRUCTURAL CHARACTERISTICS OF GRAPHS

Graph	|V|	|E|	Edge Density	Max Degree
Email_enron	36692	183831	0.00027	1482
dictionary28	52652	89038	0.00006	38
code-mat-2003	31163	120029	0.00025	202
foldoc	13356	91470	0.00103	728
web-Google	916428	4322051	0.00001	6332
soc-wiki-vote	8297	100762	0.00293	1065
daysall	13308	148035	0.00167	2265
hamingo6-4	64	704	0.34921	22
c2000.5	2000	999836	0.50010	1074
c4000.5	4000	4000268	0.50000	2123
Johnson8-94	70	1855	0.76812	53
keller4	171	9435	0.64912	124
le450_25	450	17343	0.17100	179
le450_25d	450	17425	0.17200	157
c-fat200-5	200	8473	0.42578	86
brock200-2	200	9876	0.49628	114
r250-5	250	14849	0.47700	191
r1000-1	1000	485090	0.97100	991

TABLE 3 COMPARISON OF RUNTIMES OF VARIOUS ALGORITHMS. Δ IS SIZE OF MAX-CLIQUE FOUND. T_{CP}, $T_{Ostergard}$, T_{MCQD} IS TIME TAKEN BY CARRAGHAN PARDALOS, OSTERGARD ALGORITHM AND MCQD RESPECTIVELY. T_{MCF} IS TIME TAKEN BY OUR ALGORITHM.

G	Δ	T_{CP}	$T_{Ostergard}$	T_{MCQD}	T_{MCF}
Email_enron	20	6.940	16.210	4.010	1.001
dictionary28	26	7.154	28.254	8.102	0.100
code-mat-2003	25	8.010	12.003	1.960	0.021
foldoc	9	*	3.210	0.620	0.070
web-Google	44	*	11005.070	*	1.203
soc-wiki-vote	17	0.720	0.940	0.310	2.014
daysall	28	8.440	10.010	1.002	0.210
hamingo6-4	4	<0.01	<0.01	<0.01	<0.01
c2000.5	184	40.810	25.354	8.207	32.112
c4000.5	247	378.24	101.245	24.200	87.024
Johnson8-94	14	0.201	<0.01	0.011	0.310
keller4	11	25.610	0.220	0.031	0.010
le450_25	25	0.120	0.100	0.100	<0.01
le450_25d	25	0.120	0.100	0.100	<0.01
c-fat200-5	58	0.710	0.345	0.010	0.556
brock200-2	12	1.022	0.032	0.010	0.540
r250-5	35	0.010	0.010	0.010	0.010
r1000-1	291	2.214	0.997	0.445	0.210

Three significant exact algorithms are considered for result analysis i.e. Carraghan Pardalos [15], Ostergard Algorithm [16] and MCQD [23]. For Carraghan Pardalos we have used our own implementation. For Ostergard Algorithm, we have used cliquer source code [31] that is publically available. For MCQD also we have used publically available source code available at http://insilab.org/maxclique/. Table 3 represents performance comparison of various algorithms. We have set an upper limit of 1800 seconds on runtime. The program is forcefully aborted if it fails to terminate in 1800 seconds. It is shown by an Asterisk (*) in the table. Figure 3 represents comparison of normalized runtimes of various algorithms. Figure 4 represents comparison of runtimes of various algorithms against edge density in the graphs.

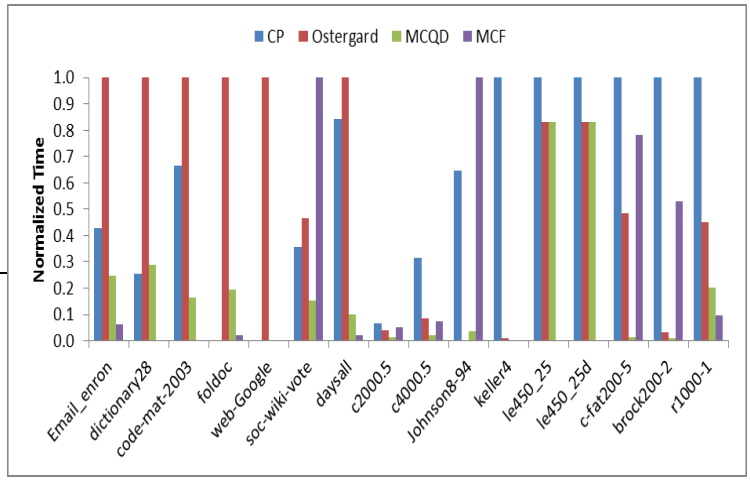

Fig. 3. Runtime (normalized to slowest algorithm) comparison of various algorithms.

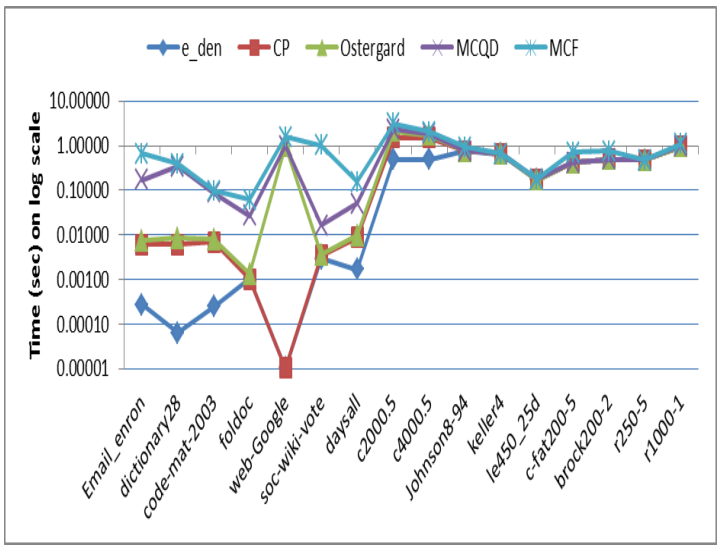

Fig. 4. Runtimes of various algorithms vs edge density.

From figure 4 we can conclude that MCF performs excellent if the graph is sparse. Maximum clique problem has major applications in community detection in social graphs. Almost every social graph follows power law of degree distribution [1], which implies that there are small number of higher degree vertices and large number of lower degree vertices. Pruning 3 in our algorithm utilizes this characteristic and most of the lower degree vertices are pruned in initial iterations of our algorithm. For DIMACS graphs also our algorithm gives very good results.

5. CONCLUSION

A novel exact algorithm, MCF, to find maximum clique has been presented in this paper. The algorithm has been tested on real world graphs and DIMACS benchmark graphs. The results are compared against performance of other significant exact algorithms such as algorithm proposed by Carraghan and Pardalos [15], Ostergard algorithm [16], MCQ [17] and MCQD [24]. The results show that proposed algorithm performed tremendously well with real world graphs. For DIMACS benchmark dense graphs, our algorithm brings slight improvement over Carraghan and Pardalos algorithm whereas cliquer [16] performs better. For sparse real world graphs our algorithm performed significantly better than any other algorithm. Maximum clique problem falls into category of NP-Hard problems. Due to this reason, maximum clique detection is avoided in several applications. But the results shown in this research establish the fact that maximum cliques can be found in feasible time in significantly large and sparse real world graphs.

REFERENCES

[1] Atul Srivastava, Anuradha and Dimple Juneja Gupta, "Social Network Analysis: Hardly Easy", *IEEE International Conference on Reliability, Optimization and Information Technology, pg. 128-135, 6, February 2014.*

[2] S. Fortunato. "Community Detection in Graphs." Physics Reports 486:3 (2010), 75–174.

[3] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. "Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society." Nature 435 (2005), pp. 814–818.

[4] S. Sadi, S. Oguducu, and A. S. Uyar. "An Efficient Community Detection Method Using Paralle Clique Finding Ants." In Proceedings of the IEEE Congress on Evolutionary Computation, pp.1–7. IEEE, 2010.

[5] T. Matsunaga, C. Yonemori, E. Tomita, and M. Muramatsu. "Clique-Based Data Mining for Related Genes in a Biomedical Database." BMC Bioinformatics 10 (2010), 205.

[6] J. G. Augustson and J. Minker. "An Analysis of Some Graph Theoretical Cluster Techniques." Journal of the ACM 17:4 (1970), 571–588.

[7] L. Wang, L. Zhou, J. Lu, and J. Yip. "An Order-Clique-Based Approach for Mining Maximal Colocations." Information Sciences 179:19 (2009), 3370–3382.

[8] R. E. Bonner. "On Some Clustering Techniques." IBM Journal of Research and Development 8:1(1964), 22–32.

[9] R. Horaud and T. Skordas. "Stereo Correspondence Through Feature Grouping and Maximal Cliques." IEEE Transactions on Pattern Analysis and Machine Intelligence 11:11(1989), 1168–1180.

[10] A. E. Brouwer, J. B. Shearer, N. J. A. Sloane, and W. D. Smith. "A New Table of Constant Weight Codes." IEEE Transactions on Information Theory 36:6 (1990), 1334–1380.

[11] M. Pavan and M. Pelillo. "A New Graph-Theoretic Approach to Clustering and Segmentation." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03), pp. 145–152. Washington, DC, USA: IEEE Computer Society, 2003.

[12] G. Gutin, J. Gross, J. Yellen. "Discrete Mathematics & Its Applications." Handbook of Graph Theory. CRC Press, 2004.

[13] P. M. Pardalos and J. Xue. "The Maximum Clique Problem." Journal of Global Optimization 4 (1994), 301–328.

[14] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP Completeness. New York, NY, USA: W. H. Freeman & Co., 1979.

[15] R. Carraghan and P. Pardalos. "An Exact Algorithm for the Maximum Clique Problem." Operations Research Letters 9:6 (1990), 375–382.

[16] P. R. J. Ostergard. "A Fast Algorithm for the Maximum Clique Problem." Discrete Applied Mathematics 120:1-3 (2002), 197–207.

[17] E. Tomita and T. Seki. "An Efficient Branch-and-Bound Algorithm for Finding a Maximum Clique." In Proceedings of the 4th International Conference on Discrete Mathematics and Theoretical Computer Science, pp. 278–289. Berlin, Heidelberg: Springer-Verlag, 2003.

[18] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. "The Maximum Clique Problem." In Hand.

[19] V. Boginski, S. Butenko, and P. M. Pardalos. "Statistical Analysis of Financial Networks." Computational Statistics & Data Analysis 48:2 (2005), 431–443.

[20] V. Batagelj and A. Mrvar, Pajek Datasets. Available online (http://vlado.fmf.uni-lj.si/pub/networks/data/), 2006.

[21] P. Prosser. "Exact Algorithms for Maximum Clique: A Computational Study." Algorithms 5:4 (2012), 545–587.

[22] D. Johnson and M. A. Trick, Editors, Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge. In DIMACS Series on Discrete Mathematics and Theoretical Computer Science 26, 1996.

[23] J. Konc and D. Janezic. "An Improved Branch and Bound Algorithm for the Maximum Clique Problem." MATCH

Communications in Mathematical Computer Chemistry 58 (2007), 569–590.

[24] P. San Segundo, D. Rodr´ıguez-Losada, and A. Jim´enez. “An Exact Bit-Parallel Algorithm for the Maximum Clique Problem.” Computers & Operations Research 38:2 (2011), 571–581.

[25] J. Leskovec, J. Kleinberg, and C. Faloutsos . “Graphs Over Time: Densification Laws, Shrinking Diameters and Possible Explanations.” In Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD’05), pp. 177–187. New York, NY, USA: ACM, 2005.

[26] M. E. J. Newman. “Co-authorship Networks and Patterns of Scientific Collaboration.” In Proceedings of the National Academy of Sciences of the United States of America 101(2004), 5200–5205.

[27] D. Howe. “Foldoc: Free On-Line Dictionary of Computing.” Available online (http://foldoc.org/).

[28] Google programming contest. Available online (http://www.google.com/programming-contest/).

[29] J. Leskovec, D. Huttenlocher, and J. Kleinberg. “Predicting Positive and Negative Links in Online Social Networks.” In Proceedings of the 19th International Conference on World Wide Web (WWW’10), pp. 641–650. New York, NY, USA: ACM, 2010.

[30] S. R. Corman, T. Kuhn, R. D. Mcphee, and K. J. Dooley. “Studying Complex Discursive Systems: Centering Resonance Analysis of Communication.” Human Communication Research 28:2 (2002), 157–206.

[31] S. Niskanen and P. R. J. Ostergard. “Cliquer user’s guide, version 1.0.” Technical Report T48, Communications Laboratory, Helsinki University of Technology, Espoo, Finland, 2003.

AUTHOR BIOGRAPHIES

Atul Srivastava received the B. Tech. degree in Information Technology from the UP Technical University, Lucknow, India, in 2008, and M.Tech. degree in Information Technology from YMCA University of Science and Technology, Faridabad, India in 2012. He is a research student of YMCA University of Science and Technology, Faridabad, India. He is currently Assistant Professor at Pranveer Singh Institute of Technology, Kanpur, India.

Dr. Anuradha Pillai has received her M. Tech and PhD in Computer Engineering from MD University Rohtak, in the years 2004 and 2011 respectively. She has published 30 research papers in various International journals and conferences. She has more than 13 years of teaching experience. Presently she is serving as Assistant Professor at YMCA University of Science & Technology, A State Govt. University, Faridabad Haryana. Her research interests include Web Mining, Data Structures and Algorithms, Databases. Research Papers listed in database.

Dr. Dimple Juneja Gupta is working as Faculty at Department of Computer Applications at NIT Kurukshetra, India.