Clique Listing Algorithms and Characteristics of Cliques in Random Graphics

Sonal Patel
Ryerson University

CLIQUE LISTING ALGORITHMS AND CHARACTERISTICS OF CLIQUES IN RANDOM GRAPHS

by

Sonal Patel

B.E in Information Technology, Nirma University, India, 2006

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Computer Science

Toronto, Ontario, Canada, 2010

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

SONAL PATEL

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

SONAL PATEL

BORROWER'S PAGE

Name	Signature	Address	Date

CLIQUE LISTING ALGORITHMS AND CHARACTERISTICS OF CLIQUES IN RANDOM GRAPHS

Sonal Patel
Master of Science, Computer Science, 2010
Ryerson University

ABSTRACT

In this thesis we address three main problems in clique detection in the area of Graph Theory. *i)* Most of current methods for clique detection are time consuming (can take exponential time to the size of input graph), so there is a practical limit on size of input graph. In this thesis we propose three different methods for estimating the number of cliques. We examine these methods for various graphs and conclude that they efficiently find the number of cliques within 5% error typically. *ii)* We compare various versions of the Bron-Kerbosch (BK) clique listing algorithm to discover a method of combining the best features of different versions. We test our new versions of BK for various inputs. *iii)* We study the characteristics of cliques in random graphs as a function of size and density.

ACKNOWLEDGEMENTS

I am very happy and thankful as I write down this section of my thesis that allows me to express my heartfelt gratitude to all the prime people who helped me come this far. I am deeply grateful to my supervisor Dr. Eric Harley whose guidance and support have made the foundational work for my thesis possible. His constant support, thoughtful ideas, encouraging words, and the valuable time given to me steered me towards the finish line of this thesis work. He constantly shared with me his never-ending trail of intellectual thoughts. He was always available to meet with me and help me in spite of his very busy schedule. Words are not enough to express my sincere thanks to Dr. Harley in helping me through everything possible to, achieve better.

I would like to thank my parents Ms. Heena and Mr. Kantilal for their prayers and support that gave me courage to fulfil my goals. I would like to express my profound gratitude to my husband Gaurav who has been a constant source of encouragement for me throughout my studies. To him, I dedicate this thesis.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Graph Terminology

An arbitrary graph is denoted by G(V,E), where V denotes the set of vertices and E stands for set of edges for the graph. A clique is a subgraph where each vertex is connected to every other vertex in the subgraph. A maximal clique is a clique which is not a proper subgraph of another clique. A maximum clique is a clique with the largest size in a given graph. Cliques are a basic concept of graph theory and are used in many other mathematical problems as well as in Data Mining [1], Web Mining [14] and Bioinformatics [7, 11, 17]. Finding and enumerating cliques from a graph is considered as NP-hard problem [8] so it is very difficult to obtain exact solution. Basically the word "clique" comes from Luce and Perry [16], who used complete subgraphs in social network to model cliques of people. In a social network, a clique represents a group of people all of whom know each other. This is just one of many real life problems where, vertices of cliques represent entities which are more closely related to each other and an edge represents a relation between two entities. We will be using the following notation to describe graphs throughout this thesis:

G(n,p)= a graph with n vertices and density of edges p;

n= |V|= number of vertices;

p= density of edges in a graph= (*100* m*) / *C (n, 2);*

m=|E|= number of edges in a graph;

μ=number of maximal cliques.

Figure 1 represents a graph G(n,p) having n=6 vertices and μ=3 maximal cliques. The maximal cliques are shown in form of subgraphs as well as in form of sets.

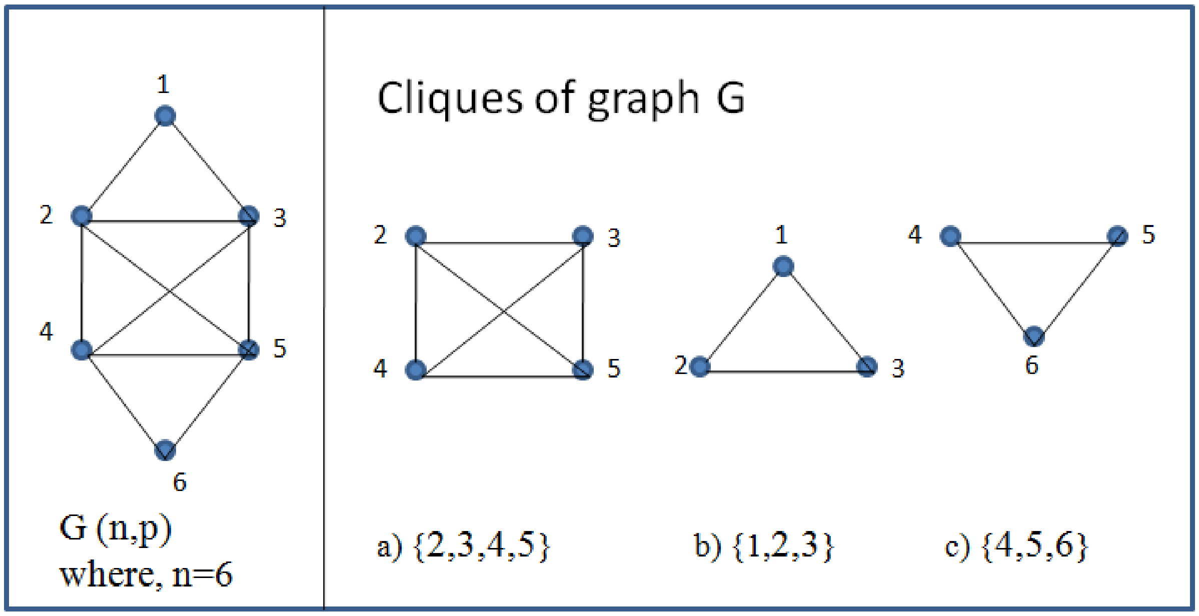
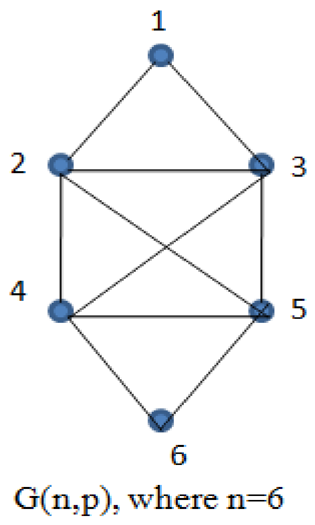
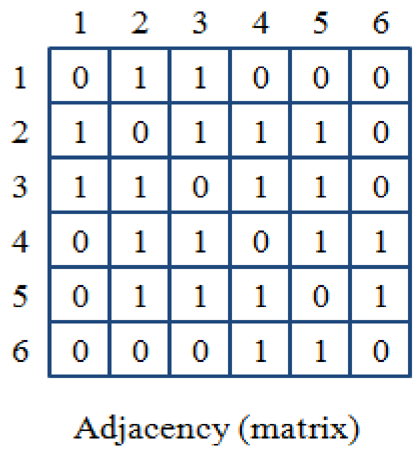


Figure 1 Graph and cliques of graph

In the program that is used for clique finding, graphs can be represented in the form of two dimensional arrays having size n x n where n is size of graph as shown in the Figure 2. As shown in the matrix, the value at position [i,j] indicates the connectivity of vertices *i* and *j* where, 1 represents the vertices are connected and 0 indicates no connection. The matrix representation of adjacency information provides fast execution but has the disadvantage of requiring O(n^2) memory space. Another way to represent adjacency information is in form of linked list as shown in the Figure 3, where vertices adjacent to vertex *i* are represented by a linked list attached to node *i*. To find the connectivity of vertices *i* and *j*, the program starts from *i* and goes over the list until it finds *j* or a higher index then *j* or the end of the list (Null). The linked list data structure requires less memory than a matrix when the graph is sparse. It is useful for very large graphs where the computer has insufficient memory for the matrix. The disadvantage of the structure is that it consumes more time than the matrix to identify connectivity of vertices.





	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	1	1	1	0
3	1	1	0	1	1	0
4	0	1	1	0	1	1
5	0	1	1	1	0	1
6	0	0	0	1	1	0

Adjacency (matrix)

Figure 2 Adjacency representation for the graph G in form of matrix.

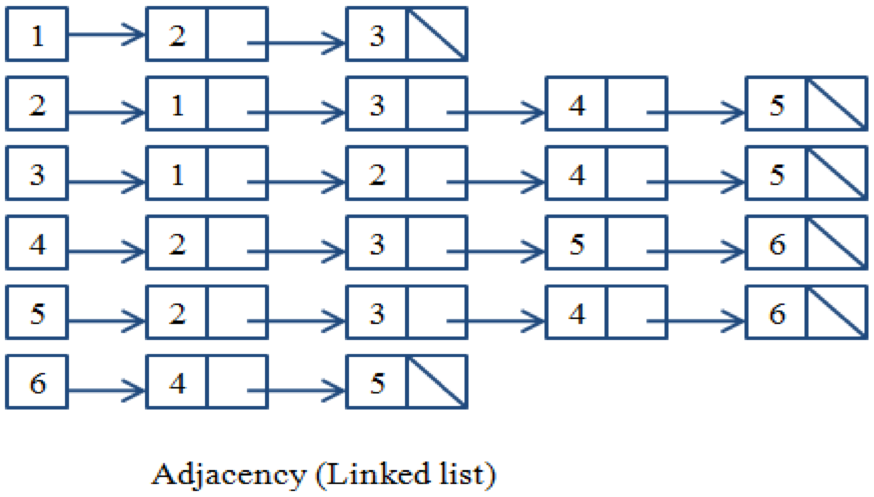


Figure 3 Adjacency representation for the graph in form of linked list.

1.2 Motivation

Many researchers point out that the relationships in nature and society can be depicted in terms of graphs. Here the graphs present an enormous network between entities that they are made of e.g. the internet, the network of WWW and a social network. The internet is made of a number of computers and servers connected with a physical link. The network of WWW is made of a number of websites connected with hyperlinks. A social network is where people are represented as nodes and the relations are represented as edges. Meanwhile, advanced computerized systems have been developed to store large size of network data having millions of nodes. This provision encourages the researchers to discover various patterns in the network.

Among all the different patterns studied in graph theory, maximal clique is most famous pattern and it features in many applications in various domains. In graph theory, identifying maximal cliques is considered as an NP-hard problem which helps in solving several other problems. Maximum clique is the clique having largest number of vertices. An independent set is a set of vertices where not a single pair of vertices is adjacent. Finding an independent set from the input graph is the same as finding a clique in the compliment of the input graph. So the problem of finding cliques and independent sets from a graph G can be solved by an algorithm developed to find and list cliques. A clique in social network can represent a group of people having the same behaviour or interest. Assembling the trends of these communities with common interest helps in the creation of new business strategies and also helps in making decisions that are helpful for the economic growth of the business. In the domain of bioinformatics, a clique can model a group of proteins with similarity in structure. Maximal clique can be used in many other applications, in fact wherever clustering is needed. All these applications motivate the clique finding algorithms.

The need to find the number of cliques in a random graph having size *n* and different edge density has become imperative. Since the existing algorithms used to find the number of cliques has an exponential time complexity, there is a need to estimate cliques efficiently while optimizing its execution time.

Moreover, the interesting features of cliques inspire the study the clique behaviour. This further draws interest to do various experiments to find out characteristics of cliques.

1.3 Problem Statement

Of all current clique detection methods, the BK method (developed by Bron and Kerbosch) [6] is most popular one. MBK (Modified BK) [9] is one of many modifications to the BK algorithm which have improved performance on a particular type of graph. MBK is proven to be more efficiently efficient than BK for large sparse graphs (low density). There is a need to develop a method which serves as most efficient method for all types of random graphs following the idea of BK method. Could an algorithm be developed which would apply either BK or MBK, depending on the density of the graph?

Second, many real life problems can be modeled using graphs containing cliques. As a preliminary step, one might want to estimate the number of cliques and their sizes. Can this estimation be done quickly, avoiding the high cost of full enumeration?

Third, the following questions concerning the characteristics of cliques are of interest:

- How does the size of cliques differ while increasing graph size?
- What is the time dependency of the rate of clique detection?
- How does the number of cliques vary with increasing size or density of the graph?
- How is the total program execution time distributed between clique detection and clique listing?

1.4 Contribution

The primary goal of this thesis is to improve existing methods of clique enumeration and estimation. We modify the clique detection method developed by Bron and Kerbosch (BK) by combining the best features of different versions of BK method, in order to improve the execution time. We propose methodologies based on sampling, probability and curve fitting to estimate the number of cliques while minimizing the execution time. For example, given a random graph with n=3200 vertices and edge density p=30% (some experiments in Section 4.3 are done for graph G(1000,30)), we can estimate the number of cliques to be 1.62E20. If we allot 1 microsecond to detect a clique then using 1000 parallel processors, exact counting with BK would take 5000 years. We compared these three methods of estimation accuracy and efficiency. These methods are useful in cases where a quick estimate of the number of cliques is required. In addition, we study the size distribution of cliques for various graphs having different sizes and densities.

1.5 Thesis Outline

The remainder of this thesis is organized as follows:

Chapter 2 - Literature Review: This Chapter comprehensively reviews the related research efforts under the area of clique detection and enumeration. It discusses various methods developed in the area of clique detection. In addition, it broadly covers the importance of cliques along with an overview of current applications use cliques to model difficult problems. Finally, the significance of clique in various areas provides the inspiration for the proposed approaches of this thesis.

Chapter 3 –Methodology: In this Chapter we discuss our methods. The Chapter is divided in to three sections. In the first section, we compare various versions of the BK method and propose a new method by combining the best features of two. In the second section, we discuss the design of experiments to detect characteristics of cliques. In the last section, we propose methods for clique estimation.

Chapter 4 -Experiments and Analysis: This Chapter is also divided in three sections, where each section describes the experiments performed to evaluate the methodology discussed in respective section of Chapter 3.

Chapter 5 - Conclusion: This Chapter concludes by summarizing the work done in this thesis and provides recommendations for potential future work based on the current contributions.

CHAPTER 2
LITERATURE REVIEW

In this Chapter we examine work in the area of clique detection as it relates to this thesis. We describe the most popular method known as BK in detail since in some of the experiments described in Chapter 3, we made small modifications to BK. We also discuss the importance of clique enumeration in various fields, including theory and real life. This serves as an inspiration for the experiments and work that are presented in Chapters 3 and 4.

2.1 Clique Listing Algorithms

The study of algorithms related to finding cliques dates back to at least 1963. The algorithms developed by Bierstone [3] and Bonner [5] provided cluster analysis for large data sets. In 1970, Augustson and Minker [2] tested these two algorithms and came to the conclusion that the algorithm developed by Bierstone [3] performed better than the one by Bonner [5]. In 1973, Bron and Kerbosch [6] developed a back tracking algorithm (sometimes known as BK) using a branch and bound technique to cut off branches of the search tree which do not lead to clique. This algorithm tends to produce larger cliques first and tends to generate cliques in a sequence where they share large common intersection. Bron and Kerbosch showed that their algorithm was faster than that of Bierstone. In 1975, Johnston [12] developed and tested some variations of the Bron and Kerbosch algorithm, some of which improved performance for certain densities of edges. In 1977, Tsukiyama [23] proposed an algorithm for generating all maximal independent sets (MIS) of a graph. The algorithm runs with $O(nm\partial)$ time and in $O(n+m)$ space, where n stands for number of vertices (n=|V|), m stands for number of edges (m=|E|) and ∂ stands for the number of maximal independent sets. In 1981, Loukakis and Tsouros [15] came up with a depth first search algorithm (LTMIS) to generate maximal independent sets in lexicographic order.

They tested the new algorithm on the various sizes of graphs from n=20 to n=1000 having density from 20% to 90%. According to their results, the algorithm LTMIS is two to fifteen times faster than the algorithm developed by Bron and Kerbosch and at least three times faster than the method developed by Tsukiyama. However, others have found BK to be faster than LTMIS [10]. The algorithm LTMIS makes use of four one-dimensional arrays of size n, which is less memory than the memory required by other methods. In 2001, Koch [13] presented several variations to the Bron and Kerbosch algorithm. Koch tested these variants on various kinds of graphs and found that their variants are not faster than the original Bron and Kerbosch algorithm as they required an expensive search for some specific vertex. However, in the case of graphs representing protein structure these variants proved to be faster than some other clique finding algorithms including BK. In 2003, Harley [9] proposed a variation of BK called "Modified Bron and Kerbosch" (MBK) which improved the performance of BK on a type of graphs used in mapping genomes. MBK was shown to be faster than several other algorithms when the input graphs are "overlap" graphs, which are basically noisy interval graphs [10]. In 2004 Mokino and Uno [17] proposed two algorithms for enumerating all maximal bipartite cliques. The first of their algorithms makes use of matrix multiplication and runs with O(M(n)) time delay (i.e. time between any two consecutive outputs) and in $O(n^2)$ space, where M(n) denotes the time needed to multiply two $n \times n$ matrices, which is currently $O(n^{2.376})$ [25]. The second algorithm runs in $O(\Delta^4)$ time delay and $O(n+m)$ space, where Δ stands for maximum degree of the graph G. This algorithm requires $O(nm)$ pre-processing time before listing first maximal clique. They also implemented Tsukiyama algorithm and compared with their two algorithms by taking various size graphs as input. According to experimental results they concluded that their algorithms are much faster than the algorithm developed by Tsukiyama.

In 2006, Tomita et al. [22] suggested a variation to BK which prints cliques in the form of a tree. This saves memory and time. Printing time is important since in our experience it consumes almost half the total time to find and print cliques. (There are a few experiments presented in Chapter 4 that separate searching and printing time of cliques). Tomita et al. also proved that the modified algorithms has worst case time complexity of $O(3^{n/3})$, where *n* is the number of vertices, which is optimal considering the Moon-Moser result [19]. In 2006, Wan et al. [24] proposed a depth first search algorithm for clique enumeration in a complex network. A complex network is a network with patterns of connection among their elements that are neither purely regular nor purely random. According to Wan et al. the actual network of telecom users does not conform to a random graph model but rather looks like a complex network with short average path length and having high clustering coefficient. "The clustering coefficient is a measure of the number of closed triplets over the total number of triplets" [24]. Here a triplet is three nodes that are connected by either two (open triplet) or three (closed triplet) edges. Using this definition it is easy to calculate the clustering coefficient for a vertex in the graph. "If vertex v has d_v neighbours, then vertex v and its neighbour can construct $m= d_v*(d_v-1)/2$ triangles at most, but the actual number of triangles constructed by v and the neighbours is H, so the clustering coefficient of v is $G_v=2*H/d_v*(d_v-1)$" [24]. Wan et al. presented an algorithm to enumerate the cliques in a complex network. It runs with $O(d^2\mu S)$ time delay and in $O(n+m)$ space and requires $O(nD^2)$ time for pre-processing, where D denotes maximum degree, μ denotes number of maximal cliques, S denotes the size of a maximum clique, d is the number of triangles containing a vertex with degree D, $d=CT$, where C is clustering coefficient and T stands for number of triangles that can be constructed from the vertices with degree D. They applied this algorithm for customer churn analysis. "Customer churn is an interesting problem in the telecommunication industry”, since effective prediction of which customers are likely to leave or join the brand may allow adjustments that result in more customers and more profit. Here the maximal cliques represent groups of customers closely related to each other. The experimental results show that, by introducing maximal cliques as a property of customer improves churn prediction. Results are satisfactory especially for the VIP customers who are most important for telecommunication industry. Wan et al. also compared their algorithm with BK by applying BK on complex networks generated from telecommunication call data of a month in a city and concluded that their algorithm performs more efficiently on complex networks than BK.

2.2 BK Algorithm

There exist many clustering techniques, used to solve problems in informatics. Here we talk about one of most popular methods, known as BK (Bron & Kerbosch) [6]. In this section, we discuss three versions of BK, and how they differ in the process of clique listing and their fundamental logic.

2.2.1 Description

In 1973, Bron and Kerbosch [6] came up with a backtracking algorithm which uses a search tree such that a path from root node to leaf node gives a solution (clique). They used a branch and bound strategy to avoid exploring branches which cannot lead to a clique. To understand the procedure of this method, we need to understand their use of the following three sets, COMPSUB, CANDIDATES and NOT.

The set COMPSUB contains vertices which will form a clique at the end of travel to leaf node of search tree. Size of COMPSUB is extended by one through traveling to leaf node and decreased by one going back up a level in the search tree. It behaves like a stack if we think about its structure in the algorithm. The points which are selected to extend COMPSUB must be connected to all other points of COMPSUB.

The set CANDIDATES contains vertices of input graph that qualified to serve an extension of the set COMPSUB. In the beginning of BK program CANDIDATES contains all vertices of the graph so it has the same size as the size of the graph. Throughout the search process as the cliques are detected, it decreases the size of the CANDIDATE set (if we ignore the size of the new CANDIDATES set in the process of internal recursion caused by an EXTEND function discussed below).

The set NOT contains points, which have already served as an extension of the set COMPSUB. These are the points that will no longer be considered in the future for extension of COMPSUB.

The sets NOT and CANDIDATES are maintained in a one dimensional global array in the program and maintained by two parameters, ne and ce, which indicate the position of last member of each of these sets. The array has following layout.

```
Global Array: | NOT|CANDIDATES   |
Indices:      |1....ne|........................ce|
Size:         |(ne)   |     (ce-ne)              |
```

There is a function called EXTEND, which is recursively applied on these three sets mentioned above. This function is responsible for extension of COMPSUB. It also checks the qualification of the member of CANDIDATES which is proposed for extension of COMPSUB. The qualified member (responsible for extension of COMPSUB) must be connected to all other points of COMPSUB.

The mechanism of BK is illustrated in Figure 4 for a graph having six vertices as shown in same figure.

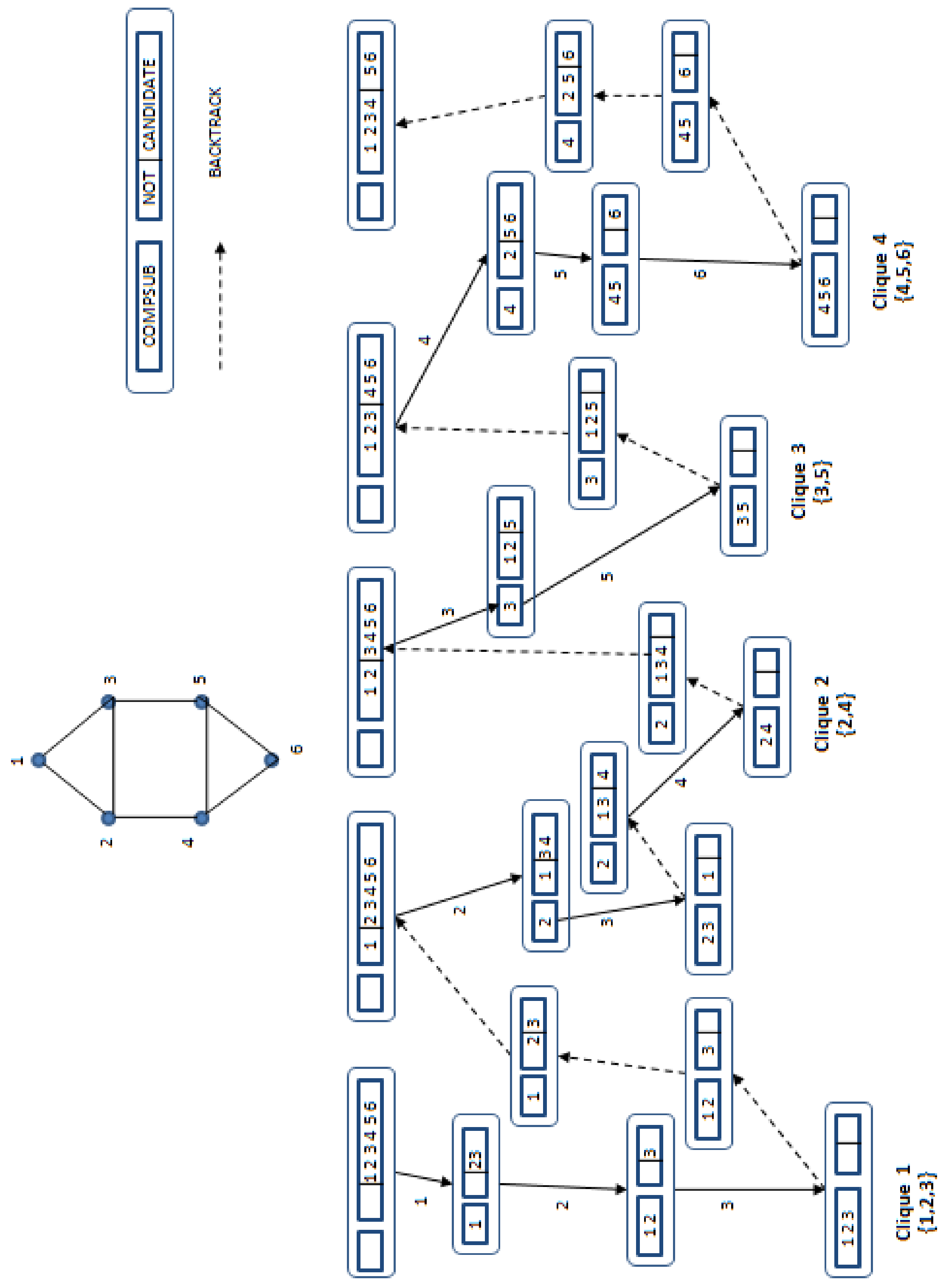


Figure 4 Mechanism of BK program (version 1)

In the beginning the sets NOT and COMPSUB are empty whereas, CANDIDATES holds the indices of all the vertices of given graph. The first step of process is to select a candidate and add the selected candidate to COMPSUB. The function EXTEND generates new sets NOT and CANDIDATES by removing all the points which are not connected to selected candidate. The function runs recursively until the set CANDIDATES becomes empty. The set COMPSUB contains a clique when the sets CANDIDATES and NOT are empty. The backtrack cycle is influenced by the following two conditions, as described in [6].

1) CANDIDATES is empty but NOT is not empty.
2) The set NOT contains a point, which is connected to all points of the set CANDIDATES.

These two situations indicate that further extension of the set COMSUB will not turn out to be a clique so further extension by calling EXTEND would be a waste of time. Stopping recursive calls at this point amounts to pruning the branches of search tree which do not lead to a maximal clique. The backtrack cycle runs until the set COMPSUB becomes empty. The backtracking process removes the points from COMPSUB and adds them to the set NOT. At the termination of backtrack cycle, the set NOT contains points which have already served as an extension of COMPSUB and the cliques having any of these points been already detected.

The algorithm described above is the basic mechanism of BK and is what Bron and Kerbosch called *version 1*. In the same paper [6] Bron and Kerbosch presented an improvement they called *version 2*. To make BK method faster for sparse graphs there is another variation known as MBK (*version 3*) [9]. Let’s see how these three versions of BK are different from each other.

In the *version 1* of the BK method, there is no internal shuffling of the set CANDIDATES while COMPSUB is extended by taking an element of CANDIDATES. Elements are selected in the sequence of their index value. *Version 1* finds cliques in lexicographical order exactly as shown in the Figure 4.

Version 2 is different from *version 1* in terms of selection process of elements from CANDIDATES. *Version 2* does not select the first candidate according to index, but instead it provides internal shuffling to find most suitable element from the set CANDIDATES. Here most suitable element is the one having minimum number of disconnections among rest of other elements of CANDIDATES. Therefore *version 2* tends to find cliques in descending order to the size of clique rather than lexicographical order.

Version 3 can be considered as combination of *version 1* and *version 2, as* it finds cliques in order of their associated index similar to *version 1* and also does internal shuffling to select candidate like *version 2*. According to methodology of *version 3*, it selects the first element from CANDIDATES then finds a subgraph for selected element. The subgraph contains selected element plus all elements connected to the selected element. Then it applies *version 2* and finds all cliques among the subgraph. *Version 3* is known as (Modified Bron and Kerbosch) MBK program [9].

2.3 Moon –Moser Graph

Moon and Moser [19] designed a graph to answer the question "what kind of graph having *n* vertices gives the maximum number of cliques?" This kind of graph is very easy to construct. Figure 5 shows the graph G having $n=9$. G is not the Moon-Moser graph but rather its compliment G' is the Moon-Moser graph as shown in the Figure 6. The graph G' has exactly $3^{n/3}$ maximal cliques, which is 27 maximal cliques.

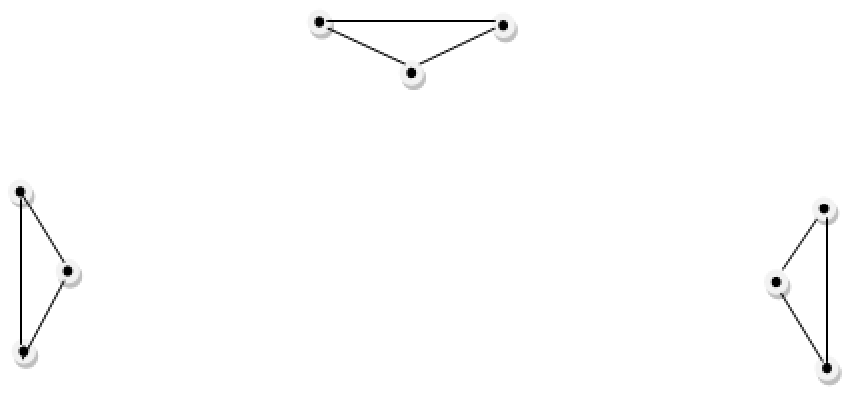


Figure 5 Graph G, where *n=9*

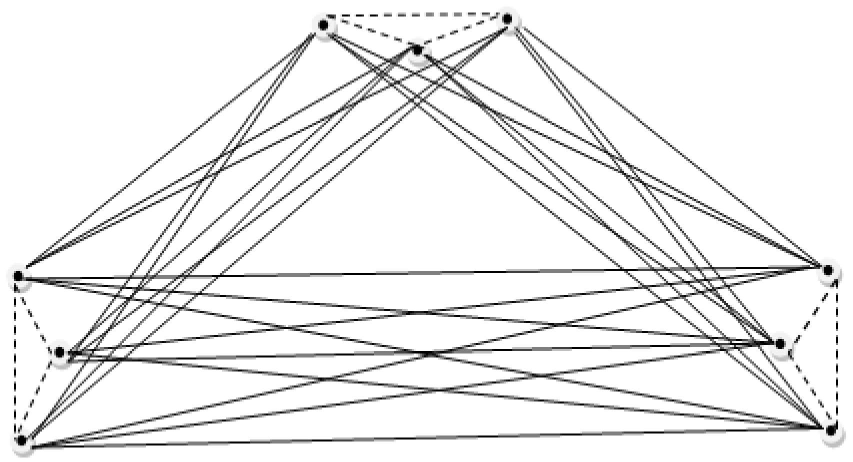


Figure 6 Graph G’, where *k=3*

2.4 Maximum Clique Problem

Maximum clique of graph G is a clique of G having largest number of vertices. Finding maximum clique from graph is proven to be NP- hard problem [8] so it is very difficult to find even an approximate solution. Branch and bound is most popular technique to reduce the search space efficiency. In 2007, Tomita developed a branch and bound algorithm [20] to find maximum cliques based on coloring and ordering vertices. This algorithm is known as MCR and they proved that it is faster than other algorithms they compared with for finding maximum clique. According to another paper they have published in 2009 [21], MCR is good but still it is not fast enough for complex and large graphs. To overcome on this weakness they introduced a new algorithm known as MSC by introducing new ordering and coloring techniques to MCR. The experimental results presented in the paper prove that MCS is faster than MCR for the graphs with high density and for very large and sparse graphs.

2.5 Applications Supported by Clique Finding

Finding and enumerating subgraphs from a single graph is becoming a hot-spot current research, especially in bioinformatics, data mining, web mining and knowledge discovery.

2.5.1 Bioinformatics

Here we talk about usefulness of clique finding approach in the bioinformatics. We can relate maximal cliques in a graph with virtual probes in genomes. A *virtual probe* is the region of the genome shared by a maximal set of fragments of the genome, used in sequencing. Finding a virtual probe is similar to finding the maximal cliques in random graph where a vertex represents a fragment and an edge represents the overlap in sequence. Harley et al. [11] presented a new way to examine overlap data using the clique finding approach.

In the field of computational biology, prediction of protein structure from amino acid sequence is one of the most important tasks. There are three different methods for prediction based on amino acid sequence of newly generated proteins as discussed in [7]. If the new protein is found to have high similarity with a known 3D structure then “homology modeling” is very useful method. Secondly, when the newly generated protein does not show any similarity with known protein structure then “ab-initio prediction” is applied. In the third case, when new protein is found to have weak homology with known structures then “protein threading” is considered as the most important approach. Bahadur et al. [7] presented three different algorithms for “protein threading” using the concept of maximum edge-weight clique. First algorithm is known as CLIQUETHREAD for protein threading. It is not very efficient for large proteins. To improve the efficiency of CLIQUETHREAD, they have generated FTHREAD, which works for large scale protein structure prediction. They have also generated algorithm NTHREAD for protein threading. These entire three algorithms are using the same concept of maximum edge clique as base.

Information related to genes is compiled in many books and in databases such as OMIM (Online Mendelian Inheritance In Man). Researchers and physicians might need to know about a gene family or gene pathway. To satisfy this need they usually do a lot of reading. Researchers and physicians are interested in some better way to recognize groups of functionally related genes rather than simply reading documents. Matsunaga et al. [18] describes a graph based approach to fulfill their requirement. A biomedical relational graph is formed by nodes that represent biological entities and edges represent relation between these entities. They have constructed a biological graph whose nodes are pages of genes or diseases and edges are hyperlink connections between pages. They experiment on the OMIM database having more than 10,000 entities. Set of related genes detected by enumerating maximal cliques are utilized to understand the mechanism of complex diseases.

2.5.2 Data Mining and Web Mining

Finding associations rules is one of the main approaches in database mining. The paper [1] discusses the problem of discovering association rules between items in sales transaction database. Successful marketing firms are much interested to view such a database and to come up with new attractive marketing strategy. An example of such a rule might be 98% of customers who purchase tires and auto accessories also get automotive service done [1]. Finding such rules are important for cross marketing. The authors have proposed two new algorithms to discover association rules using the concept of graph theory. They also discussed how these two algorithms can be combined for best performance. Discovery of association rules is also considered as central task of knowledge discovery. The algorithm [14] uses novel item set clustering techniques to approximate the set of potentially maximal frequent items. Experimental results for this method indicate improvement over existing algorithms. Traditional data mining is about inferring association rules based on support and confidence whereas one application of web mining is finding groups of individuals who share a common interest. These communities can be identified using web graph (the directed graph induced by the hyperlinks between Web pages) [14]. Instead of using resource gathering algorithms that look for pages related some specific topic here, they scan through a web and identify instances of graph structure that are community of people share a common interest.

CHAPTER 3
METHODOLOGY

This Chapter covers methodologies related to the main contributions of the thesis, which are threefold. (1) We compare various versions of Bron and Kerbosch algorithms for clique detection in terms of required execution time to list all maximal cliques. The purpose of this comparison is to come up with some idea to improve existing Bron and Kerbosch method. (2) We determine characteristics of cliques in terms of various sizes, density of graph, time distribution for clique detection and clique frequency. (3) We propose effective methods to predict number of cliques for a random graph based on number of vertices and density of edges.

3.1 Clique Listing Algorithms

Finding maximal cliques of a random graph can take exponential time considering the exponential number of cliques in the Moon-Moser graph described in Chapter 2. Various algorithms have been developed to find and list cliques. The best known method is that of by Bron and Kerbosch [6]. We will refer this method as the BK algorithm throughout this Chapter. The BK method uses a branch and bound technique to cut off the branches of search tree which do not lead to successful end point. We use two versions of this algorithm known as BK [6] and MBK [9], written in C. There are two different programs for each version based on different data structures: BK (list) and MBK(list) - where adjacency information is stored in a linked list to reduce memory allocation, BK(matrix) and MBK(matrix) - where adjacency information stored in matrix (faster than the link list structure). According to the working of BK [Section 2.2], the arrays CANDIDATES and NOT scanned each time the EXTEND function is called. CANDIDATES stores candidate vertices and is scanned to find next vertex which has “minimum number of disconnections” to extend COMPSUB. This searching process for most suitable vertex, requires much time for very large graphs. To overcome this expense of time, a new program was developed called modified Bron and Kerbosch (MBK) [9]. This modification reduces the scanning time of CANDIDATES and NOT every time to search for proper vertex. According to the modification the array CANDIDATES is not scanned and shuffled every time to find the most suitable vertex when the EXTEND function called. When looking for suitable candidate it selects candidates according to their index from the set CANDIDATES. Consider v_1 has index 1 (CANDIDATES [1]). To find cliques for vertex v_1, MBK selects vertices connected to v_1 and makes a subgraph called Si_1. When the search for all maximal cliques containing v_1 is done, v_1 added to the list called *done*. The list *done* is added to list NOT before it moves to next subgraph. In this section, we compare these two different programs, BK and MBK, in terms of execution time required for various densities. Based on comparison results we will also discuss our idea to improve existing programs.

3.1.1 Comparison of BK and MBK

In Chapter 4, we compare BK and MBK in terms of program execution time required to find and list the maximal cliques. We choose input graphs with various sizes and densities. We generate three random versions of each graph to get a better estimation. We supply these graphs as input to programs BK (matrix) and MBK (matrix). Let *t(BK)* be the execution time using BK program and *t(MBK)* be the execution time using MBK program. We plot the ratio *t(BK)* / *t(MBK)* (using average execution time for three random graphs) against the size of graphs.

According to the experimental results, MBK appears better than BK for low densities (< 30%) and BK seems better than MBK for high densities (≥ 30%). We will discuss this experiment further in Chapter 4.

3.1.2 Combining BK and MBK

In the previous section, we mentioned that for lower density MBK works better than BK and for higher density BK is better. Though the difference is not remarkable, we create a program that works better for all densities. This new program is a combination of BK and MBK methods called MBK_2. It uses BK for density equal to or higher than 30% and uses MBK for input graphs with density less than 30%. We will discuss the experimental results in the next Chapter.

3.1.3 Improvement to BK

Section 2.2 explains that the EXTEND function in the BK method is responsible to detect the proper candidate to extend COMPSUB to form a clique. When the EXTEND function is called, it first scans the list of CANDIDATES to find the most suitable candidate having minimum number of disconnections. This process takes time which is in quadratic in the size of CANDIDATES. We modify this method so that it finds the first suitable candidate having minimum number of disconnections in linear rather than quadratic time. To make this change, we calculate the number of disconnections for each vertex when we generate the adjacency matrix for the input graph. This modification should reduce the execution time for very large and dense graphs. We will see how effective this approach is compared to BK in Chapter 4.

3.1.4 Improvement to MBK

The MBK method improves performance of BK for sparse graphs. Applying this method, we can get cliques in order of their associated index whereas BK method tends to find cliques in descending order of their sizes. The modification we suggest is, rearrange the list of vertices prior to applying MBK. We have generated a new program which sorts the vertices from high to low degree, then applies MBK for clique detection. Here in this new program, we are using a linked list to hold adjacency information. We will compare our new program with MBK (list) to get idea how better/worse the new approach is for clique detection.

3.2 Characteristics of Cliques in Random Graph

In this section, we discuss the various kinds of behaviours and patterns of cliques for certain inputs. We will discuss the attributes of cliques in terms of their size, their detection rate in certain graph, and time distribution in clique detection process among clique printing and clique detection processes. We will also talk about modal size of cliques and their distribution for various densities. Lastly, we will discuss the distribution of cliques in terms of the number of cliques vs. density and the number of cliques vs. number of vertices.

3.2.1 Clique Detection Rate

Clique detection rate indicates number of cliques found in fixed definite intervals of time. To examine clique detection rate we are using MBK method. MBK program gives total number of cliques and total amount of time required to detect all cliques for input graph. This experiment shows the average rate of finding cliques as function of time. It will answer question such as "where in the process is the clique finding rate highest, or is it constant?"

We modified the MBK program in a way that it prints number of cliques after each interval of six seconds. It also prints the selected vertex at the moment which is responsible to create a subgraph. We plot the experiment results as average rate of finding cliques vs. time. We will discuss more about this experiment and results in Chapter 4.

3.2.2 Time Distribution of clique listing for the Graph G(n,p)

Our next observation is about time required to find total number of cliques for various size graphs. The clique detection methods we have seen so far, print details of cliques in separate file and count all number of cliques. Sometimes a person is only interested in number of cliques rather than details of cliques. This experiment indicated the amount of time we can save to detect number of cliques for input graph if the clique detail is not important. We are also calculating time to print each clique to a file for various size graphs. Here, we are taking various size graphs for density 20% as input and find clique printing time required for each graph. We have also calculated the percentage amount of time required to print all the cliques in separate file from total required time to find and print cliques.

Based on this observation, we can say that printing time is almost 50% of total time, so printing time increases rapidly as we increase number of vertices. If user is not interested in clique detail then it is better to avoid clique printing, so by doing this, person can save 50% of total time.

3.2.3 Size of Cliques

This experiment examines the distribution of the size of cliques. We adjust the MBK program to count the number of cliques of each size. We applied various sizes of graphs as input to updated program. We plot the number of cliques against their size. Results of this experiment shows the size of the most frequent clique and the dependence of clique size on graph size or graph density.

3.2.4 Modal Size Clique

The modal size for the cliques is the most frequent clique size in the graph. In this section, we try to find modal size for various densities for specific size graph. To find modal size clique, we are using same program, used to find clique frequency. To analyze this experiment, we plot the results in the way that the X-axis shows various densities and the Y-axis shows the modal size for various densities. These results show the variation of modal size as we increase density. We will also discuss how the modal size of cliques differs for various sizes of graphs for fixed density.

To find modal size, we need to find all cliques and their sizes in the graph, and then distribute all cliques based on their sizes. We choose modal size of cliques which is most frequent clique size among all. This process is time consuming for large graphs. So here we tried to predict modal size of cliques for various densities based on sampling. For sampling process, we select a few subgraphs from the given graph, and then we find the modal size for cliques for those subgraphs. We defined sample modal size as most frequent size of cliques among the cliques of subgraphs. We will also compare actual modal size and predicted modal size based on sampling process.

3.2.5 More on Number of cliques vs. Number of vertices and Density

When density of input graph is fixed, increasing number of vertices yields rapid increase in number of cliques. When number of vertices is fixed, what happens as we increase density? For what density is number of cliques maximum or minimum? To answer such questions we did a few experiments using MBK for input graphs with different sizes and densities (we will discuss the experimental results in Chapter 4).

3.3 Predicting the Number of Cliques in G(n,p)

We mentioned earlier that there are many algorithms developed to enumerate the cliques in a random graph. This enumeration problem is exponential in complexity as discussed earlier, so there are practical limits on the size of graphs for which enumeration of cliques is possible. For very large graphs these methods can take eons. In this section, we propose three methods that could be used to predict the number of cliques in a random graph. One method is based on curve fitting, a second method is based on sampling and third method is based on calculation using probability arguments. We will compare all these three methods in Chapter 4 in terms of accuracy and efficiency.

3.3.1 Curve Fitting.

In the curve fitting approach, we try to find a functional relationship between the number of cliques and the number of vertices. We use MATLAB to fit a curve to data representing the number of cliques and number of vertices for the graph G(n,p). Suppose *f(n)* represent function relating number of vertices n to number of cliques, as obtained by least square fitting in MATLAB. Let μ_i represent the numbers of cliques for the graph G(n_i,p) and let $y_i = f(n_i)$, be the estimated number of cliques based on curve fitting. Using MATLAB we also determined a coefficient called “norm of residuals”, which is a measure of the deviation between the actual and the predicted value. The lower the norm of residuals, the more closely the equation fits the data. Formally, the norm of residual is defined as follows.

$$Norm\ of\ Residual = \sqrt{\sum_{i=1}^{n}(di^2)}$$

Where $d_i = \mu_i - y_i$ is the residual, i.e., the difference between actual and predicted values.

In MATLAB we can try various degrees of polynomial curves to fit this relation *f(n)*. The maximum degree allowed is 10. We tried various polynomial formulas and compared fitted curves with actual results using BK for same graph.

3.3.2 Sampling

In the sampling approach, we estimate the number of cliques in a random graph by selecting a few subgraphs and applying the BK method to find the number of cliques for each sample. We use the sample counts to estimate the number of cliques for the entire input graph G(n,p).

The accuracy of the estimation depends on the selection of subgraphs and number of samples. We first order the vertices in descending order according to their number of adjacent vertices (degree) for better accuracy. We then select vertices at regular intervals, starting with the vertex of highest degree. Each selected vertex and its neighbours (adjacent vertices) define a subgraph.

We apply the BK method to count the number of cliques for each subgraph. Using MATLAB, we produce a graph for the number of cliques vs. index of the vertex defining the subgraph. The area under this curve is our estimate of the total number of cliques for the input graph G(n,p).

3.3.3 Calculation

In this approach, we use probabilistic arguments to calculate the expected number of cliques in a random graph G(n,p). Let V be the set of n vertices of G, and let A_r be the set of subsets of V which have size *r*. The size of A_r is

$$C\ (n\ ,r) = n!\ /\ (n-r)!\ r!$$

This is the well-known formula for the number of ways of choosing m items from a set of n items without regard to order. The probability that any given subset in A_r is a clique is $p^{\ C\ (r,\ 2)}$, since the number of edges that are possible among r vertices is *C(r,2)* and each edge has probability p. Thus, the number of cliques (not necessarily maximal) of size *r* in A_r and thereby in G is expected to be

$$C\ (n,\ r) * p\ C\ (r,\ 2)$$

Consider one of the above cliques, K_r, and a vertex *v* not belonging to K_r. The probability that *v* together with K_r form a clique is p^r, since every vertex in K_r must connect to vertex v. Thus, the probability that *v* does not form a clique with K_r is $q = 1 - p^r$. The probability that K_r does not form a clique with any other vertex is $q^{\ (n-r)}$. Thus, the number of maximal cliques of size *r* is expected to be

$$C\ (n,\ r) * p^{\ C\ (r,\ 2)} * (1 - p^r)^{\ (n-r)}$$

The number of maximal cliques in G(n,p) is therefore

$$\sum_{r=1}^{n}\ (C\ (n,\ r) * p^{\ C\ (r,\ 2)} * (1 - p^r)^{(n-r)}) \qquad (1)$$

For the purpose of this formula, we define C(1,2) as 0. Alternatively, if one assumes the graph is connected, then the summation can start with r = 2. In practice the summation is carried out from r = 2 up to the value of r where $C(n, r) * p^{C(r, 2)}$ becomes zero, since at that point no clique (maximal or otherwise) is expected. This approach also serves requirement to know the maximum size and number of maximum size cliques available in the input graph. After deriving this equation we searched the literature to see if it is new. We found that Bollobas and Erdos have published this equation in 1976 [4].

3.4 Chapter Summary

This Chapter has discussed different kinds of patterns and behaviour of cliques. We have re-discovered an equation for clique prediction using probability theory (calculation) and we propose two more different directions for clique prediction. We have also proposed various ideas to improve clique detection methods BK and MBK.

CHAPTER 4

EXPERIMENTS AND ANALYSIS

4.1 Clique Listing Algorithms

This Chapter shows experimental results for the experiments discussed in Chapter 3.

4.1.1 Comparison of BK and MBK

We compare BK and MBK with respect to execution time required to find and list all maximal cliques of a graph. We use input graphs having various densities and sizes. For better precision, we generate three random versions of each input graph. We calculate the average of $t_{(BK)}/ t_{(MBK)}$, where $t_{(BK)}$ and $t_{(MBK)}$ are program execution times for BK and MBK, respectively. Figure 7 represents straight line least square fits to the data points calculated for each density. In this figure and throughout this Chapter, p and n stand for density and size of input graph, respectively. The figure shows that at low densities (5%-20%) MBK takes roughly 5% to 10% less time than the BK method, whereas at high densities (30%-60%) the reverse is true.

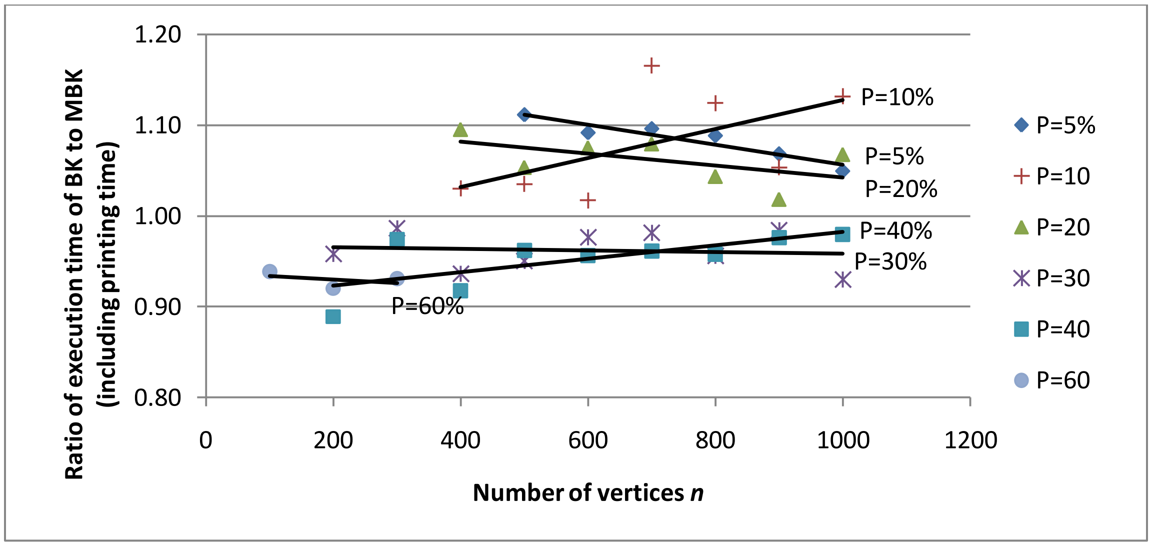


Figure 7 Comparison of BK and MBK

4.1.2 Combining BK and MBK

In the previous Section 4.1.1, we compared the BK method with the MBK in terms of their execution time. Insipired by the reseult of this comparision we produce a new method MBK$_2$ which switches between BK and MBK based on density. Here, we are comparing our new method MBK$_2$ with BK and MBK. Table 1 shows the execution time for all these three methods in milliseconds for the densities 10% to 50% for n=500. The ratio of BK to MBK$_2$ plotted in the Figure 8 shows that MBK$_2$ works like BK(ratio is nearly equal to 1) for higher density ($\geq$30) and behaves same as MBK for lower density (<30%). The ratio of MBK to MBK$_2$ indicates that MBK$_2$ performs like BK for high density and the same as MBK for low density. However, MBK$_2$ is the best for all densities and therefore the program combining both BK and MBK proves to the best fit.

Table 1 Execution time in milliseconds for BK, MBK and MBK$_2$ for n=500

Density p	t (BK)	t (MBK)	t (MBK$_2$)
10	565	495	501
20	6245	6096	6014
30	37716	38690	37495
40	460996	475250	460216
50	12837143	13073630	12651357

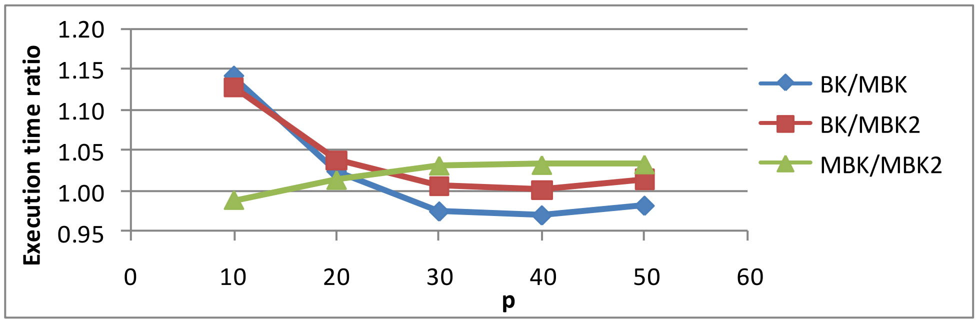


Figure 8 Comparisons of BK, MBK and MBK$_2$

4.1.3 Improvement to BK

As discussed in Chapter 2, the EXTEND function of BK looks for a proper candidate having "least number of disconnections" from the list of candidates every time after completion of BACKTRACK cycle. It really becomes time consuming process for very dense and large graphs especially when the proper candidate is at the end of list. It might be a good idea to save the time to find first proper candidate from the list, which might improve the performance of BK method for large graphs. To implement this change, we modified existing BK to MBK_3. Here we are comparing the improvement of MBK_3 in terms of execution time. Table 2 represents execution time in milliseconds required by BK and MBK_3. For proper estimation, we derived three random versions of each graph and took average of their execution time as result shown in Table 2. We calculated the improvement in MBK_3 over BK in percentage. Though the improvement is very negligible (around 2%) for small and dense graphs, we notice the jump in the improvement for large sparse graphs.

Table 2 Comparison of BK and MBK_3 with respect to their execution time (ms)

n	*p*	t (BK)	t (MBK3)	Improvement over BK
80	80	5760	5650	2%
90	80	18500	18130	2%
100	80	59100	57700	2%
200	60	100365	97955	2%
1000	10	6060	5560	8%
2000	5	24360	22650	7%

4.1.4 Improvement to MBK

We discussed the internal workings of MBK in Chapter 3. The strategy used in MBK generates cliques in order of their associated index. We generated a new program MBK_4 which rearranges the list of candidates in descending order according to their degree and then applied the clique detection strategy used in MBK. This change might improve performance over MBK. MBK_4 generates cliques in random order unlike MBK. Here we examine the new generated method for noisy interval graphs and random graphs having various sizes and densities. As previous experiments, we generated three replica of each graph and calculate average execution time required by MBK and MBK_4. Table 3 shows execution time for MBK and MBK_4 and also presents the improvement gain by MBK_4 over MBK. Here the input graphs are noisy interval graphs (shaped like overlap graphs). The table shows that the improvement of MBK_4 is up to 36% over MBK.

Table 3 Comparison of execution time (ms) of MBK and MBK_4

n	*p*	t (MBK)	t (MBK_4)	Improvement over MBK
1000	0.9	110	70	36%
2000	0.45	130	100	23%
4000	0.23	140	100	29%
8000	0.11	200	180	10%
16000	0.06	340	280	18%
32000	0.03	530	430	19%

In the Table 4, we examine random graphs rather than interval graphs. The results show that the performance of MBK_4 is worse by up to 103% over MBK. The conclusion therefore is that MBK_4 shows an improvement only when used with interval graphs having very low density.

Table 4 Comparison of execution times (ms) of MBK and MBK4

n	*p*	t(MBK)	t(MBK_4)	Declination over MBK
80	80	6172	7768	26%
90	80	19327	25767	33%
100	80	60935	80348	32%
200	60	106278	153501	44%
1000	10	5408	7202	33%
1000	0.9	32	65	103%
2000	45	35	60	71%
2000	5	21826	34252	57%
4000	5	382266	534183	40%
4000	0.23	75	91	21%

4.2 Characteristics of Cliques

In this section, we determine different features of cliques by doing various experiments. Here we modified MBK (matrix) program to determine each characteristic. For many characteristics, the BK would serve the purpose as well.

4.2.1 Clique Detection Rate

In this section, we examine the rate of clique detection using MBK algorithm. To do this, we printed the number of cliques detected during regular intervals.

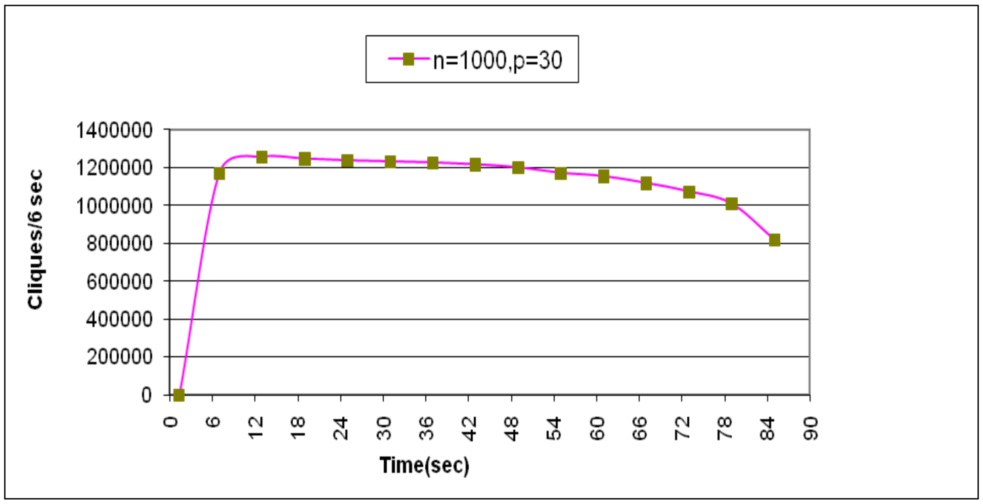


Figure 9 Clique detection rate per 6 sec

Figure 9 plots number of cliques detected per 6 seconds. The X axis specifies time with intervals of 6 seconds, and the Y axis shows cliques detected in each interval.

Based on the figure, it is clear that clique detection rate is relatively constant for most of the process.

4.2.2 Size of Cliques

In this section, we determine the distribution of the sizes of cliques. The following tables (5 to 10) represent the number of cliques found for various sizes. Each table presents clique size, number of cliques for corresponding clique size and percentage of cliques found at that size. Here μ is stands for number of cliques.

Table 5 Distribution of clique size for G (1000, 1)

Clique_size	μ	% of cliques found
2	4.02E+03	91.67
3	3.64E+02	8.31
4	1.00E+00	0.02
Total:	**4.38E+03**	

Table 6 Distribution of clique size for G (1000, 5)

Clique_size	μ	% of cliques found
2	1.97E+02	0.38
3	4.38E+04	85.46
4	7.21E+03	14.07
5	4.60E+01	0.09
Total:	**5.12E+04**	

Table 7 Distribution of clique size for G (1000, 10)

Clique_size	μ	% of cliques found
2	1.00E+00	0.00
3	5.53E+04	42.70
4	7.14E+04	55.12
5	2.81E+03	2.17
6	9.00E+00	0.01
Total	**1.29E+05**	

Table 8 Distribution of clique size for G (1000, 20)

Clique_size	μ	% of cliques found
3	4.39E+02	0.03
4	5.16E+05	40.43
5	7.03E+05	55.07
6	5.64E+04	4.42
7	6.76E+02	0.05
8	1.00E+00	0.00
Total :	**1.28E+06**	

Table 9 Distribution of clique size for G (1000, 30)

Clique_size	μ	% of cliques found
4	8.65E+03	0.05
5	4.11E+06	25.38
6	1.01E+07	62.13
7	1.94E+06	11.99
8	7.10E+04	0.44
9	5.87E+02	0.00
10	2.00E+00	0.00
Total:	**1.62E+07**	

Table 10 Distribution of clique size for G (1000, 40)

Clique_size	μ	% of cliques found
5	3.06E+04	0.01
6	2.35E+07	7.72
7	1.70E+08	55.80
8	9.90E+07	32.58
9	1.14E+07	3.77
10	3.81E+05	0.13
11	4.28E+03	0.00
12	1.40E+01	0.00
Total :	**3.04E+08**	

Figure 10 represents the percentage of cliques for the corresponding clique size for the graphs G(n,p) where n = 1000 and p = 1, 5, 10, 20, 30 and 40. The X axis shows various clique sizes and the Y axis shows the percentage of cliques with the indicated size.

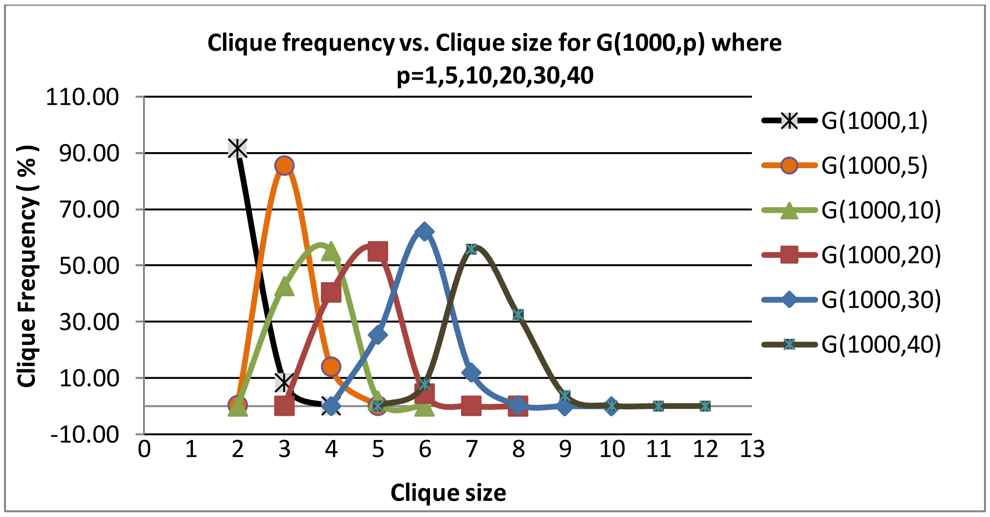


Figure 10 Clique frequency vs. Clique size

As we increase density for a fixed number of vertices, the size of cliques increases. For example, the modal size of cliques increases steadily from 3 to 7 as density increases from 5% to 40%. We did not measure beyond p=40% since as the density increases the time to find all the cliques increases. At p=40% and n=1000, the time is about 4.5 hours.

Figure 11 shows clique size distributions for graphs G(n,p) where n=500 vertices and density increases from p=10% to p=50%. This figure confirms our observation from Figure 10 since the most frequent clique size increases steadily from 3 to 9 as the density increases from 10% to 50%.

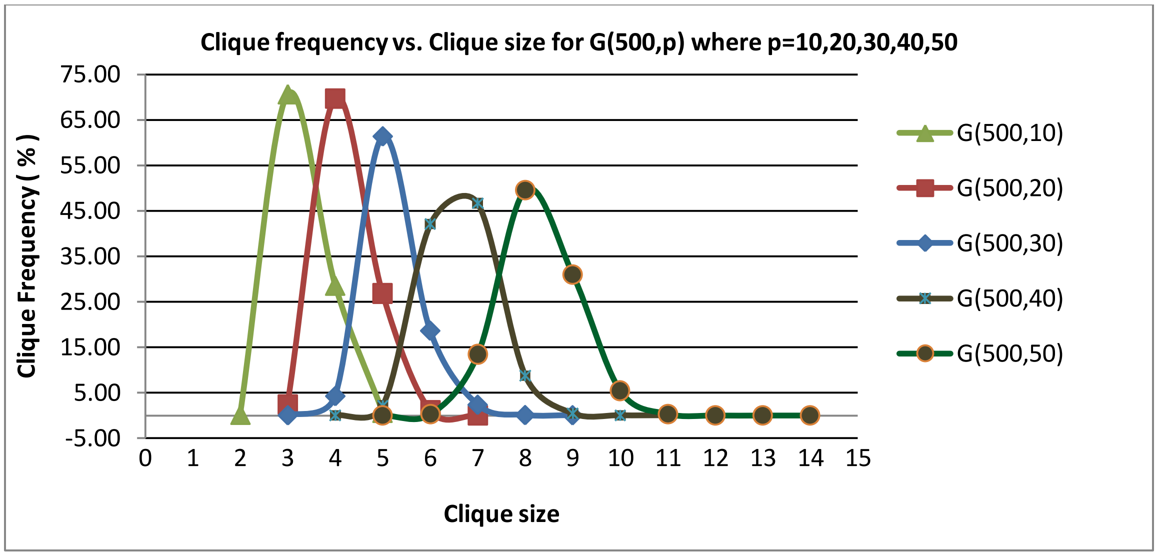


Figure 11 Clique frequency vs. Clique size

Figure 12 shows a similar experiment with smaller size and wider range of densities. Here, n=100 and p=10%, 30%, 50%, 70%, 90%, 95%, 97% and 100%. As we increase the density of edges for a fixed no of vertices, the modal size of cliques increases steadily to its maximum possible value. The maximum size occurs at 100% density where, there is only one clique, which has the same size as the graph.

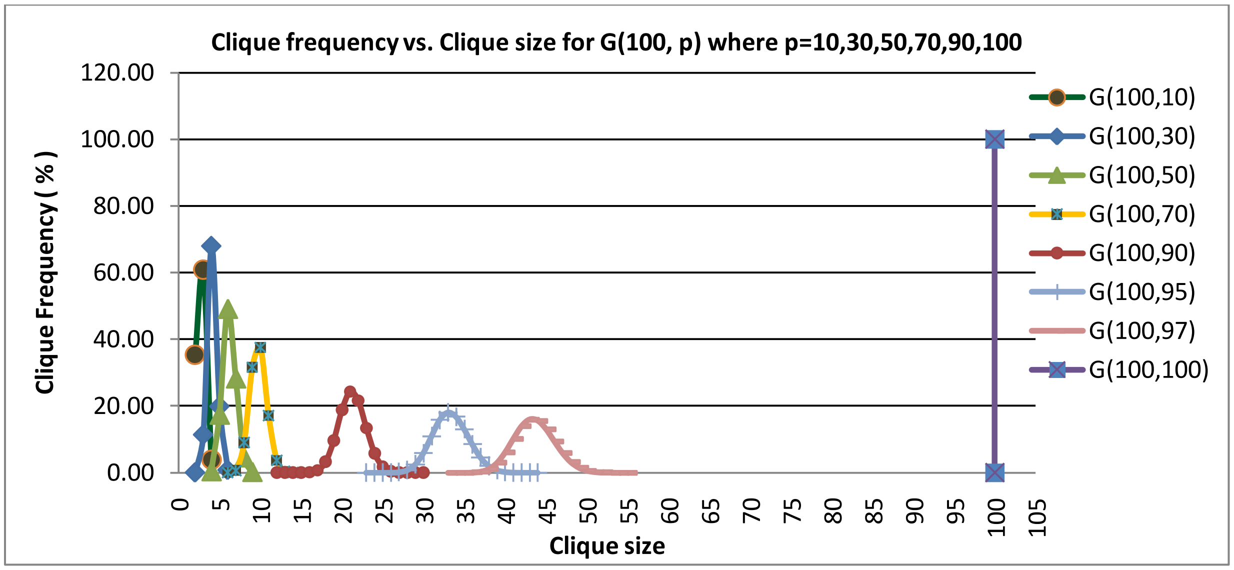


Figure 12 Clique frequency vs. Clique size

In the above experiments, we fixed the graph size and varied the density.

Figure 13 represents number of cliques for various densities for graph of size n= 100. It is clear that number of cliques increases steadily with density to its maximum possible value. The maximum number of cliques occurred at 95% density. Table 11 gives details of Figure 13.

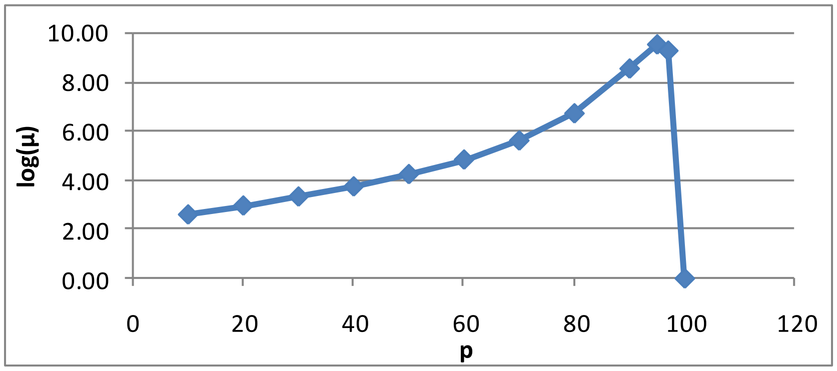


Figure 13 log(μ) vs. density.

Table 11 Distribution of number of cliques for various densities for graph G(100,p)

p	μ	log (μ)
10	3.93E+02	2.59
20	9.02E+02	2.96
30	2.09E+03	3.32
40	5.38E+03	3.73
50	1.70E+04	4.23
60	6.59E+04	4.82
70	3.90E+05	5.59
80	4.92E+06	6.69
90	3.21E+08	8.51
95	3.05E+09	9.48
97	1.70E+09	9.23
100	1.00E+00	0.00

In the next experiment, we fix the density and vary the number of vertices. Tables from 12 to 17 represent clique-size, number of cliques and percentage of clique for number of vertices respectively 500, 1000, 2000, 4000, 8000 and 16000 having density =5%.

Table 12 Distribution of clique size for G (500, 5)

Clique_size	μ	% of cliques found
2	5.93E+02	7.29
3	6.95E+03	85.40
4	5.92E+02	7.28
5	3.00E+00	0.04
Total:	**8.14E+03**	

Table 13 Distribution of clique size for G (1000, 5)

Clique_size	μ	% of cliques found
2	1.97E+02	0.38
3	4.38E+04	85.46
4	7.21E+03	14.07
5	4.60E+01	0.09
Total:	**5.12E+04**	

Table 14 Distribution of clique size for G (2000, 5)

Clique_size	μ	% of cliques found
2	3.00E+00	0.00
3	2.33E+05	69.27
4	1.02E+05	30.32
5	1.39E+03	0.41
Total:	**3.37E+05**	

Table 15 Distribution of clique size for G (4000, 5)

Clique_size	μ	% of cliques found
3	8.53E+05	35.99
4	1.48E+06	62.34
5	3.97E+04	1.68
6	5.30E+01	0.00
Total:	**2.37E+06**	

Table 16 Distribution of clique size for G (8000, 5)

Clique_size	μ	% of cliques found
3	1.48E+06	6.40
4	2.05E+07	88.52
5	1.17E+06	5.06
6	3.56E+03	0.02
7	1.00E+00	0.00
Total:	**2.31E+07**	

Table 17 Distribution of clique size for G (16000, 5)

Clique_size	μ	% of cliques found
3	5.58E+05	0.19
4	2.59E+08	87.60
5	3.59E+07	12.14
6	2.06E+05	0.07
7	8.50E+01	0.00
Total:	**2.96E+08**	

Chart in Figure 14 represents distribution of clique with respect to size of clique for various numbers of vertices. X-axis in chart shows size of clique and Y axis in chart represents percentages of clique corresponding clique size.

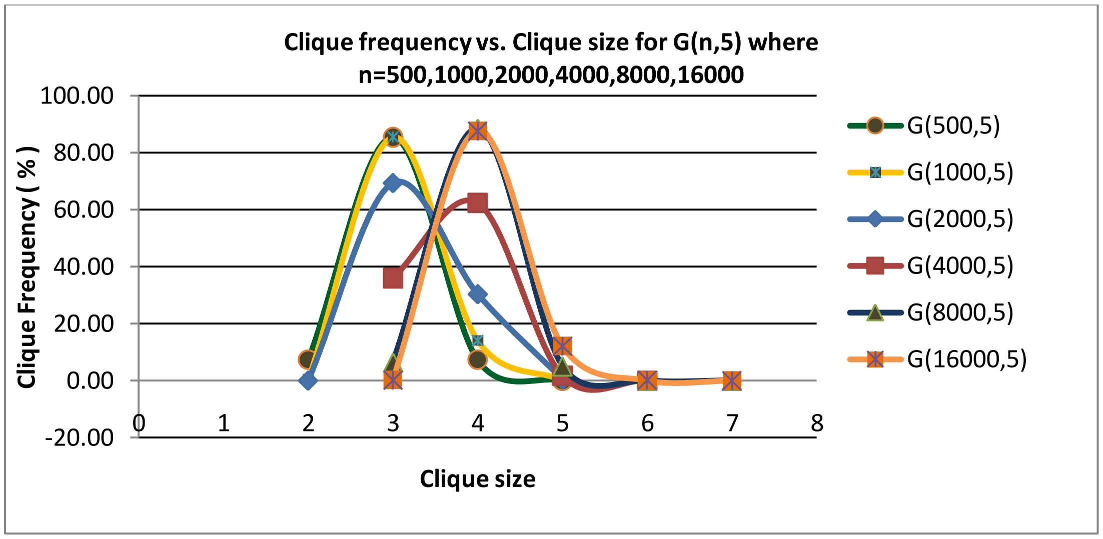


Figure 14 Clique frequency vs. clique size

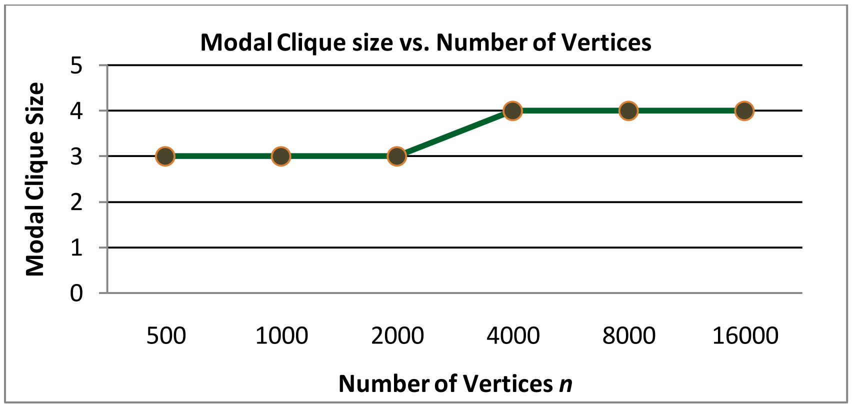


Figure 15 Modal Clique Size vs. Number of Vertices for *p*=5%

These results show that the modal size of cliques increases very slowly with the number of vertices at density p=5%. For better analysis, we added one more plot for Modal size vs. number of vertices as shown in Figure 15. For example, the graph G (500, 5) has maximum frequency for cliques with size of 3, similarly for graph G (1000, 5) and graph G (2000, 5) have maximum number of cliques with same size 3. Graph G(4000,5) has maximum frequency for cliques of size 4, as do G(8000,5) and G(16000,5). Figure 16 shows a similar experiment on the graph G(n, 20) where n=100, 200, 500, 1000 and 2000. Figure 16 represents experiment results in form of clique frequency vs. clique size. Here this result supports same conclusion as we have obtained previously, i.e. the modal size of clique increases slowly with n. Figure 17 shows modal size for number of vertices which makes clear picture of modal size clique behaviour. For example, graph G(100,20) has maximum cliques with size 3, graphs G(200,20), G(500,20) have maximum cliques with size 4 and graphs G(n,20) where n=1000 to 2000 have maximum cliques with size 5. So as we increase size of graphs for same density, modal size cliques increasing very slowly for p=5% or p=20%.

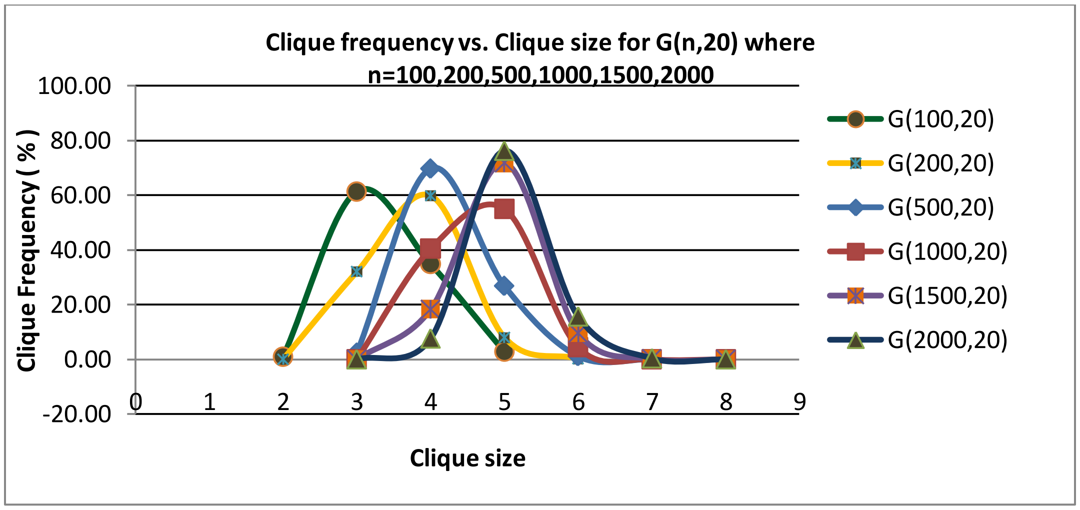


Figure 16 Clique frequency vs. clique size

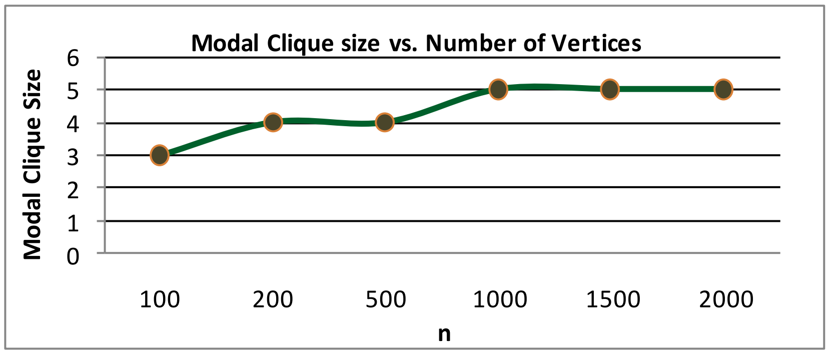


Figure 17 Modal Clique Size vs. Number of Vertices for *p*=20%

4.2.3 Modal Size Clique

In this section, we examine the modal size of cliques for graphs with various densities and numbers of vertices. Table 18 represents modal size of cliques for graphs G(1000,p), where p= 5 to 40% .

Table 18 Modal size of cliques for various densities for G (1000, p).

p	Modal size
5	3
10	4
20	5
30	6
40	7

Figure 18 represents the data from Table 18 as graph. It clearly shows that as we increase density, modal size of cliques increases. In this experiment, each increase of 10% in density shifts the modal size by one unit.

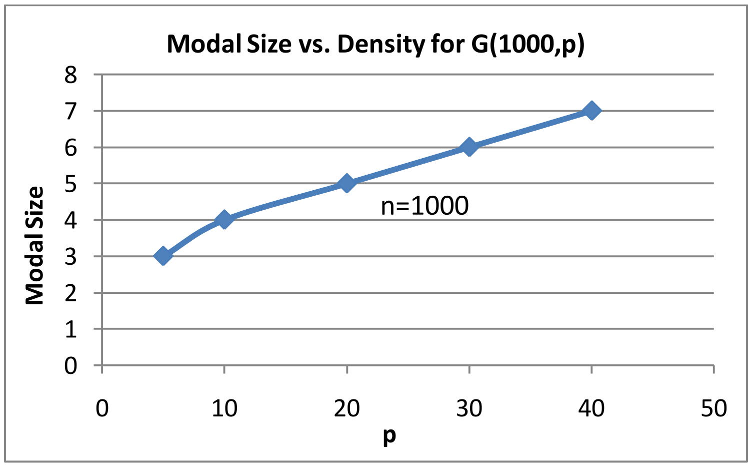


Figure 18 Modal size of cliques vs. density for G (1000, p).

Table 19 and Figure 19 show similar results for graphs G(100,p).

Table 19 Modal size of cliques for various density for G(100,p).

p	Modal size
10	3
20	3
30	4
40	5
50	6
60	7
70	10
80	14
90	21
100	100

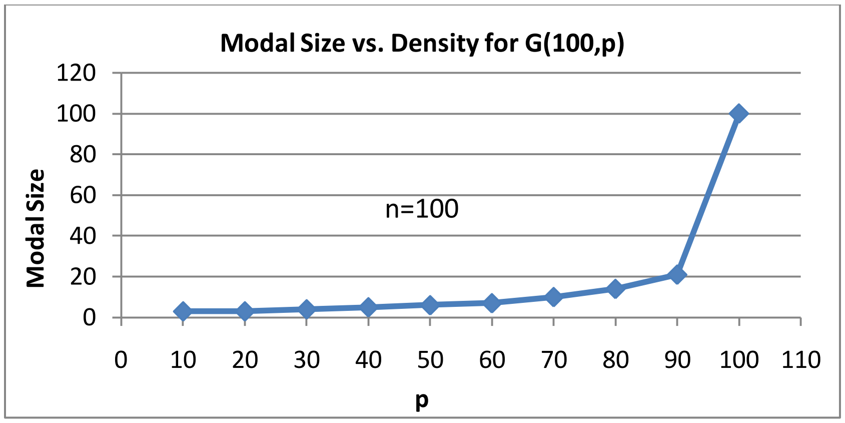


Figure 19 Modal size of cliques vs. density for G(100,p).

Clearly, the modal size of cliques increases with density, as one would expect. Figure 19 shows that the rate of increase in the size of cliques is low for low densities but becomes rapid at high densities.

To find the modal size of cliques, we find all cliques in the graph and then count them according to size. The modal size of cliques is the most frequent clique size. This process is time consuming for large graphs. Here we tried to predict the modal size of cliques for various densities based on sampling. For sampling process, we select few subgraphs from given graph G(n,p) then find modal size for cliques for subgraphs. The Table 20 represents sample modal size of cliques for the graph G(1000,p). Our approach for selection of subgraphs is to first select vertices after regular interval starting from index 1. Consider here our selected vertices are v_1,

v_{100}, v_{200}, v_{300}…, v_{900} and v_{1000}. We define subgraphs for each selected vertex which is the selected vertex and vertices adjacent to selected vertex. We derived sample modal size as most frequent size of cliques among the cliques of sub graphs.

Table 20 Sample Modal size of cliques for various densities for G (1000, p).

p	Sample Modal size	Actual Modal Size
5	3	3
10	4	4
20	5	5
30	6	6
40	7	7
50	9	NOT DONE
60	12	NOT DONE
70	15	NOT DONE

Figure 20 plots sample modal size of cliques vs. density for the graph G (1000, p). We plot actual modal size again for comparison of this chart with the chart in Figure 18. By sampling we can derive same results as experimental results without sampling up to p=40. Sampling process helps us to find modal size cliques for higher density (here for G(1000,p) up to 70%). Without sampling, time becomes prohibitive.

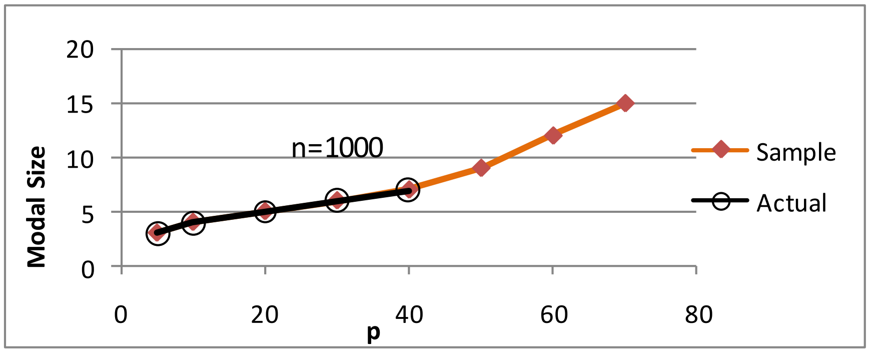


Figure 20 Sample modal size of cliques vs. density for G (1000, p).

We also compared modal sizes of cliques as determined by sampling with the sizes obtained by complete counting using the graph G (100, p). Here we chose vertices v_1, v_{10}, v_{20}, v_{30}…, v_{90} and

v_{100} to obtain subgraphs. Table 21 shows sample modal size and actual modal size for the graph G(100,p).

Table 21 Sample Modal size of cliques for various density for G (100, p)

p	Sample Modal size	Actual Modal Size
10	3	3
20	3	3
30	4	4
40	5	5
50	6	6
60	7	7
70	10	10
80	14	14
90	21	21
100	100	100

Figure 21 plots sample modal size vs. density for the graph G(100, p). We plot the actual modal size again for comparison of this chart with the chart in Figure 19. It represents the same results as we have in Figure 19.

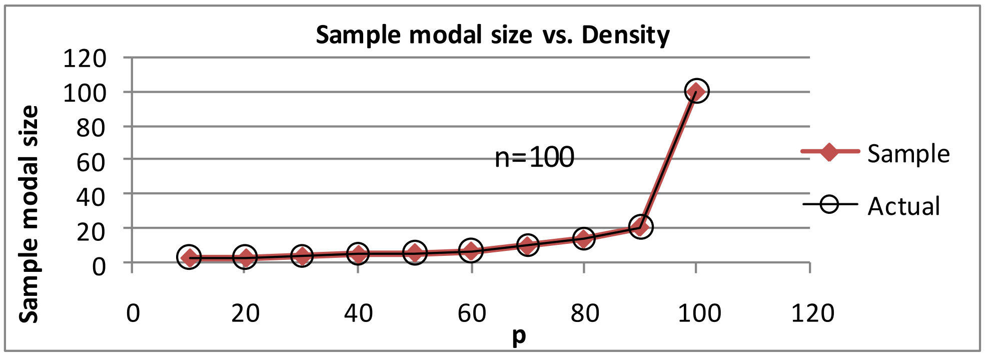


Figure 21 Sample modal size of cliques vs. density for G (100, p).

We have seen how modal size of cliques varies for various densities for a fixed size of graph. Now we will see how the modal size of cliques differs for various sizes of graphs where the density remains constant.

Figure 22 plots modal size of clique for various sizes of graphs. This Figure shows that modal size of cliques is same (i.e., modal size = 3) for graphs with 5% density in the range tested (n = 50 to 1000). As we increase density, we find some variation in modal size of clique, e.g. for 20%

density, the modal clique size is 3 for n =50 to 200 vertices, whereas for n=500 and 1000, the modal sizes are 4 and 5 respectively. For higher density like 40% the modal size of cliques increases as we increase the number of vertices up to n=500. The modal size was the same at n=1000 as at n=500.

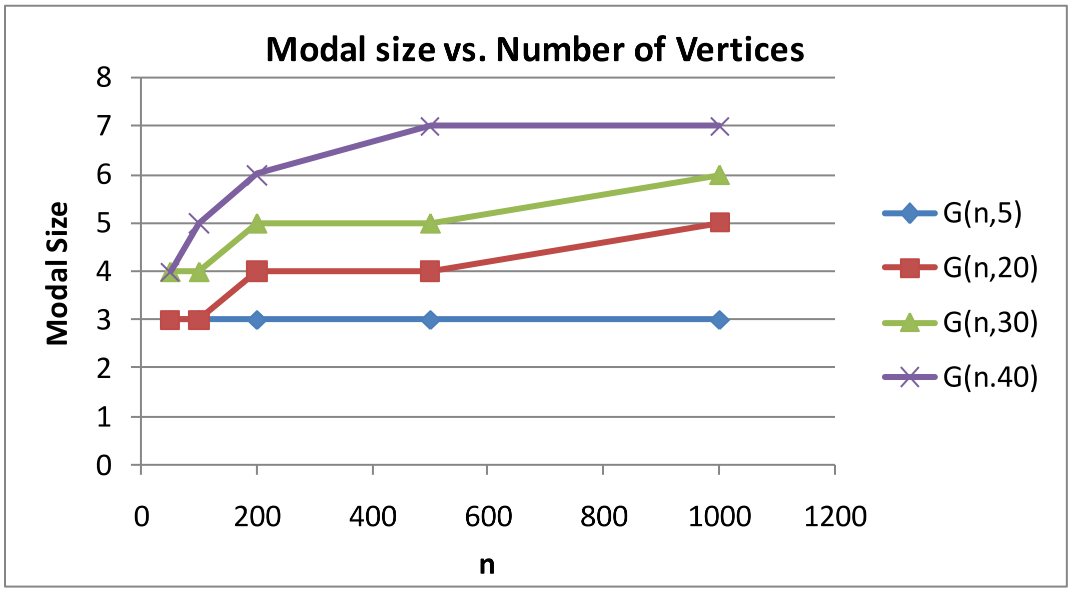


Figure 22 Modal sizes of cliques vs. N.O.V.

4.2.4 Time Distribution for the Graph G(n,p)

Our next observation concerns the time required to find cliques vs. the time required to print them to a file. We use various size graphs G(n, p) where p=20%. Table 22 shows the times, where

t1 = Time required to find all cliques and print them to a file.

t2 = Time required to find all cliques but not to print them.

t3 = Time required to print all cliques to a file.

The last column represents average time required to print a clique for related graph.

Table 22 -Time to find and list cliques for G (n, 20) where n=50, 100,200,500,1000,1500,2000

n	Time t1 (msec)	Time t2(msec)	Time t3 (msec)	Printing time	µ	Printing Time/µ (ms/clique)
50	7.00E+00	6.00E+00	1.00E+00	14%	1.61E+02	6.21E-03
100	1.40E+01	1.20E+01	2.00E+00	14%	9.02E+02	2.22E-03
200	6.10E+01	2.20E+01	3.90E+01	64%	5.74E+03	6.80E-03
500	5.68E+02	3.23E+02	2.45E+02	43%	1.08E+05	2.27E-03
1000	6.17E+03	3.03E+03	3.14E+03	51%	1.28E+06	2.46E-03
1500	3.24E+04	1.49E+04	1.75E+04	54%	6.16E+06	2.84E-03
2000	1.11E+05	5.85E+04	5.29E+04	47%	2.04E+07	2.59E-03

Chart in Figure 23 represents comparison of time t1, time t2 and printing time. X axis in the chart stands for number of vertices and Y axis expresses time in millisecond. Based on this observation, we can say that printing time is almost 50% of total time (time t1). If user is not interested in clique detail then it is better to avoid clique printing as it consume 50% of total time.

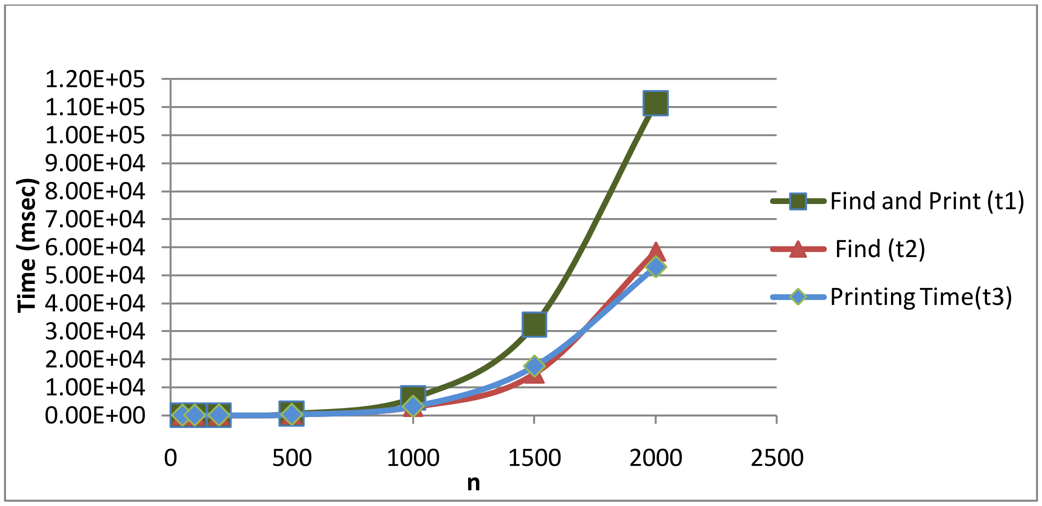


Figure 23 Time to find and list cliques vs. number of vertices

4.2.5 More on Number of cliques vs. Density and Number of vertices

Table 23 shows the logarithm (base 10) of the number of cliques for various size graphs and various densities.

Table 23 Distribution of number of cliques for number of vertices for various densities

Density(P)	log _10(μ)					
	n=50	n=80	n=100	n=200	n=500	n=1000
5	1.96	2.29	2.45	3.01	3.91	4.71
10	1.97	2.37	2.59	3.27	4.29	5.11
20	2.21	2.69	2.96	3.76	5.04	6.11
30	2.47	3.02	3.32	4.32	5.87	7.21
40	2.71	3.39	3.73	4.94	6.83	8.48
50	3.01	3.80	4.22	5.70	8.01	
60	3.29	4.29	4.82	6.64		
80	4.05	5.79	6.69	9.86		

The data in Table 23 can be plotted in two ways, as shown in Figures 24 and 25. Figure 24 plots number of cliques vs. density, whereas Figure 25 plots number of cliques vs. size of graph. Note that the graph G(80, 80) has approximately 5 times more cliques then the graph G(1000, 10).

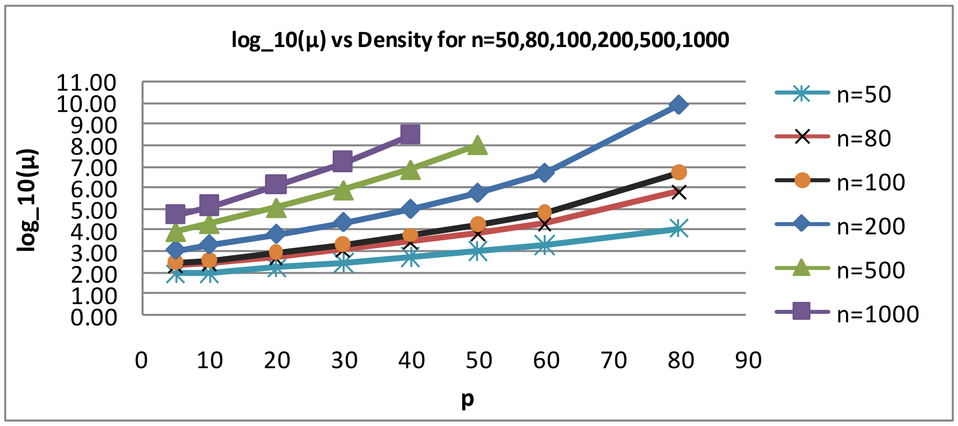


Figure 24 log_10 (number of cliques) vs. Density, for various sizes of graph.

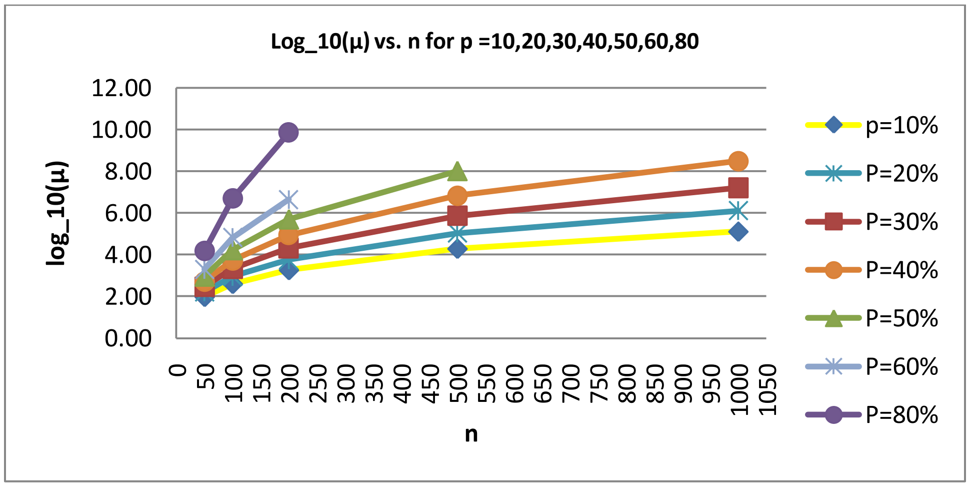


Figure 25 log_10 (number of cliques) vs. number of vertices, for various densities

Also, the graph G(200, 80) has approximately 450 times more cliques then the graph G (1000, 30). Clearly a small graph with high density can have many more cliques than a large graph with low density.

4.3 Estimating The Number of Cliques for The Graph G(n,p)

In this section, we show some experimental results for estimating the number of cliques in a random graph G(n,p) using the three methods discussed in Chapter 3. For most of the experiments that we show here, the graph is G(1000, 30), meaning n = 1000 vertices and density p = 30%.

4.3.1 Sampling

For the sampling experiment we have selected 11 vertices from the graph G(1000,30), (v_1, v_{100}, v_{200}, v_{300}…, v_{900} and v_{1000}) and used BK to count the associated cliques. Table 24 shows number of cliques calculated for the vertices selected. The number of cliques decreases swiftly with increasing vertex index. For each vertex we consider only cliques with vertices of higher index to avoid clique repetition.

Table 24 number of cliques for each sample of the graph G(1000,30)

Index of Vertex	Selected Vertex	μ
1	569	1.81E+05
100	925	5.23E+04
200	205	2.03E+04
300	410	1.07E+04
400	438	2.85E+03
500	923	1.29E+03
600	918	3.02E+02
700	187	1.52E+02
800	794	1.70E+01
900	275	1.00E+00
1000	82	0.00E+00

Figure 26 plots number of cliques vs. index of selected vertices. We calculate the area under the curve and use that value for the estimation for the number of cliques.

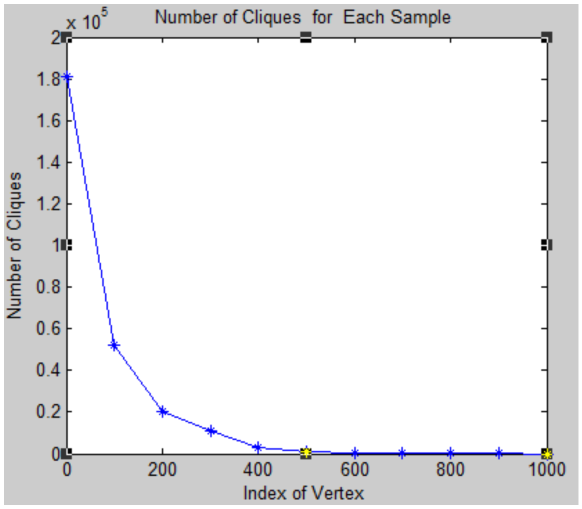


Figure 26 Number of cliques per subgraph

To determine the accuracy of the sampling process, we did the following.

First, we chose various sizes of graphs. Second, for each graph we generated three random versions. Third, for each random version we calculated the number of cliques using sampling method. Fourth, we took various numbers of samples for each random version to determine the error variation with respect to the number of samples. Table 25 shows the average percentage of errors from three random versions of the selected graphs respectively and the corresponding error variations with respect to number of samples. When the number of samples is 20, the error is 3% to 8% in the estimation of the number of cliques.

Table 25 Accuracy of the sampling method corresponding number of samples

	Error in Clique prediction		
	10 samples	**20 samples**	**50 samples**
G(100,70)	24±17 %	8±5 %	9±3 %
G(500,40)	23±5 %	5±2 %	3±2 %
G(1000,30)	18±5 %	8±2 %	3±2 %
G(2000,20)	17±5 %	7±2 %	5±3 %
G(10000,3)	11±2 %	3±0 %	3±0 %

We also did experiments where the number of samples is a percentage of the size of the graph. Table 26 represents error in clique prediction where the number of samples is a given percent of the number of vertices. When the number of subgraph is 5% of the number vertices the error is high for G(100,70) but low (3%) for G(500,40), G(1000,30) and G(10000,3). The execution time required for the sampling process increases with the percentage of samples. For example, when the number of samples is 0.5% of the number of vertices then the execution time is roughly 0.5% of the full counting time.

Table 26 Accuracy of the sampling method corresponding percentage of samples

	Error in Clique prediction			
	0.5% samples	**1% samples**	**5% samples**	**10% samples**
G(100,70)	NA	1042±212 %	128±38 %	24±17%
G(500,40)	165±15%	59±8 %	3±1 %	3±2 %
G(1000,30	62±6 %	18±5 %	3±2 %	1±1 %
G(10000,3	3±1 %	1±0 %	0±0 %	0±0 %

4.3.2 Curve Fitting

We have generated three random graphs with edge density 30% for graphs from 20 vertices to 1000 vertices. Using the BK program, we counted the number of cliques for each random graph. Figure 27 plots the average number of cliques for three random graphs of each size.

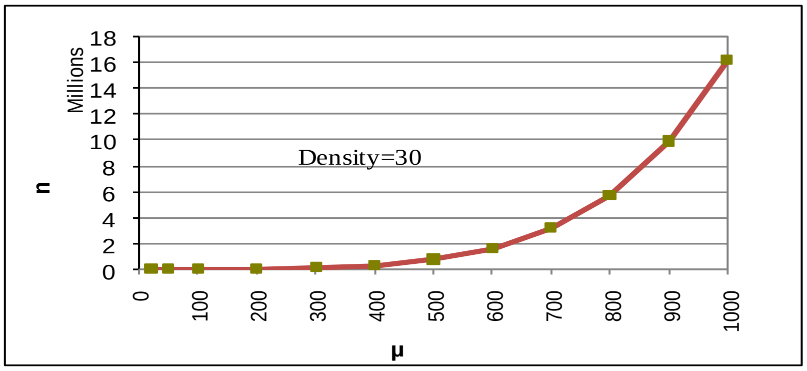


Figure 27 Number of vertices vs. Number of cliques

Using MATLAB we fit polynomial formulas to the data of Figure 27, thus inducing the functional relation between number of cliques and number of vertices. We started from linear polynomials and continued our search for best fit until the maximum degree available for curve fitting in MATLAB which is a 10th degree polynomial. Figure 28 shows the behaviour of quadratic and 10th degree polynomial formulas. The graph shows that the 10th degree polynomial fits the data quite accurately whereas the quadratic and linear are more approximate. Table 27 shows percent errors in fitted curves at each data point. According to Table 27 the formulas give large percent error in clique prediction for smaller n. Clearly, this is an artefact of the least square process and the fact that the values at high n are orders of magnitude larger than the values at low n.

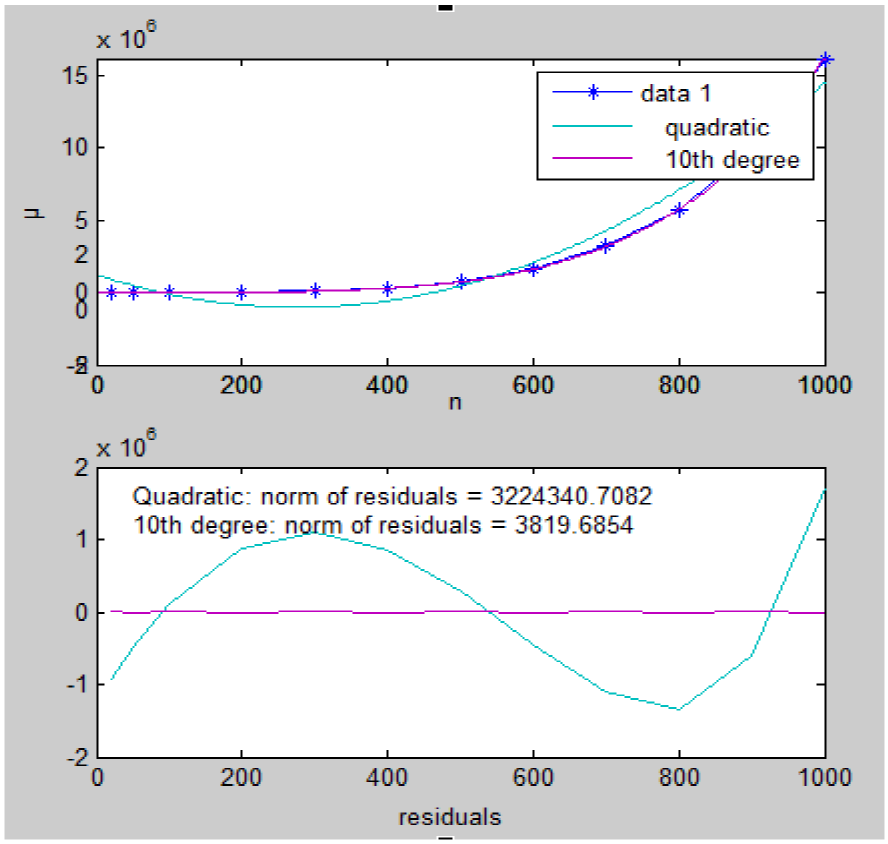


Figure 28 Curve fitting: number of cliques vs. number of vertices

Table 27 Curve fitting: number of cliques y_i (where *i* indicates degree of polynomial equation) and percent error | μ-y |*100 /μ vs. number of vertices n (density p=30%).

		10th Degree Polynomial		Quadratic	
n	μ	y1	Percent Error	y2	Percent Error
20	3.50E+01	-1.64E+02	569%	9.17E+05	2619900%
50	2.92E+02	9.00E+02	208%	4.90E+05	167708%
100	2.09E+03	1.29E+03	38%	-1.04E+05	5069%
200	2.11E+04	2.21E+04	5%	-8.48E+05	4124%
300	9.40E+04	9.23E+04	2%	-1.00E+06	1164%
400	2.93E+05	2.95E+05	1%	-5.65E+05	293%
500	7.38E+05	7.36E+05	0%	4.63E+05	37%
600	1.62E+06	1.62E+06	0%	2.80E+06	73%
700	3.20E+06	3.20E+06	0%	4.29E+06	34%
800	5.74E+06	5.74E+06	0%	7.09E+06	24%
900	9.89E+06	9.89E+06	0%	1.05E+07	6%
1000	1.62E+07	1.62E+07	0%	1.45E+07	10%

To overcome this, we fit the polynomial to the logarithm of number of cliques vs. n. Figure 29 shows curves of quadratic and 10th degree polynomial to the data. The chart in bottom of Figure 29 shows residuals as an indication of the goodness of correlation. These coefficients indicate that the higher degree polynomial gives better fit.

Here is the 10th degree formula we derived using MATLAB.

$y_{10} = p_1x^{10} + p_2x^9 + p_3x^8 + p_4x^7 + p_5x^6 + p_6x^5 + p_7x^4 + p_8x^3 + p_9x^2 + p_{10}x + p_{11}$

Coefficients: p_1 = -9.8298e-027, p_2 = 5.066e-023, p_3 = -1.1289e-019, p_4 = 1.4256e-016, p_5 = -1.1242e-013, p_6 = 5.7517e-011, p_7 = -1.9261e-008, p_9 = -0.00058565, p_{10} = 0.058723, p_{11} = 0.56914

Here is the quadratic formula we derived using MATLAB.

$y_2 = p_1x^2 + p_2x + p_3$, Coefficients: p_1 = -6.1538e-006, p_2 = 0.011174, p_3 = 1.9258

Here is the linear formula we derived (since we do not want y to decrease for large x because of having negative coefficient for the quadratic term) using MATLAB.

y_1 = p1*x + p2, Coefficients: p1 = 12519, p2 = -2.6633e+006

We do not show the graphs for linear equation, since the fit is not good. In these formulas, x stands for number of vertices and y_1, y_2 and y_{10} indicate logarithm of estimated number of cliques. Table 28 shows the percent error which is up to 2% and 483% respectively for y_{10} and y_2.

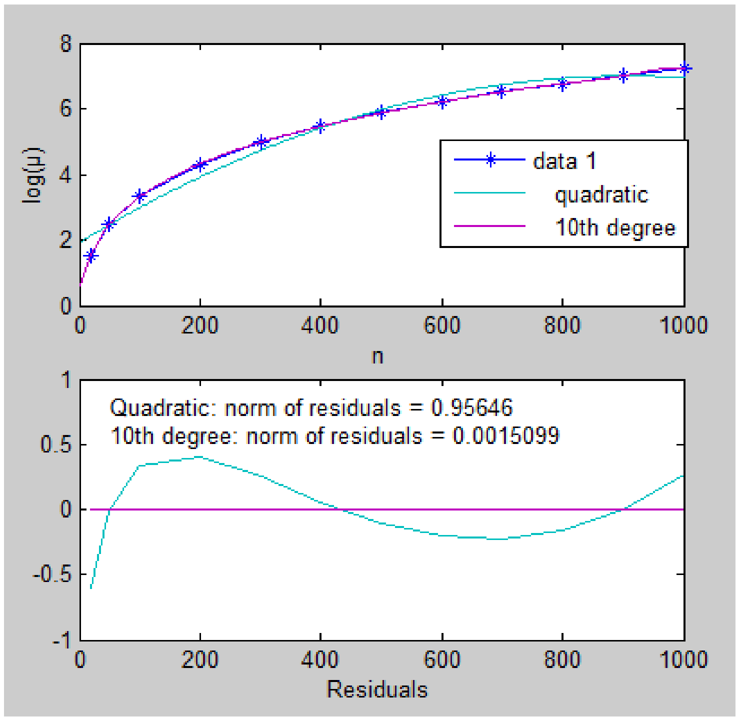


Figure 29 Curve fitting: log number of cliques vs. n.

Table 28 Estimated number of cliques y_i (where *i* indicates degree of polynomial equation) and percent error | μ-y |*100 /μ vs. number of vertices n (density p=30%) using log scale for curve fitting.

		10th Degree Polynomial		Quadratic	
n	μ	y10	Percent Error	y2	Percent Error
20	35	3.50E+01	0%	2.04E+02	483%
50	292	2.88E+02	1%	2.95E+02	1%
100	2093	2.09E+03	0%	9.55E+02	54%
200	21073	2.09E+04	1%	8.13E+03	61%
300	93989	9.55E+04	2%	5.25E+04	44%
400	292619	2.88E+05	1%	2.57E+05	12%
500	737523	7.41E+05	1%	9.33E+05	27%
600	1616656	1.62E+06	0%	2.57E+06	59%
700	3196690	3.24E+06	1%	5.37E+06	68%
800	5740164	5.75E+06	0%	8.51E+06	48%
900	9890770	1.00E+07	1%	1.00E+07	1%
1000	16180870	1.62E+07	0%	8.91E+06	45%

Consider using the derived equations (y_1, y_2 and y_{10}) to predict the number of cliques for a graph with 1500 vertices where, actual number of cliques (μ) is 1.16E+08.

	y_1	y_2	y_{10}
The Estimated values (log μ)	10.5	4.84	-4.91E03
Predicted number of cliques	$10^{(10.5)} \cong$ 3.16E+10	$10^{(4.84)} \cong 69183$	$10^{(-4.91E03)} \cong 0$
((Actual-Predicted)/ Actual)*100	27181%	100%	100%

Thus, although the quadratic polynomial estimate is better than the 10th degree polynomial, the error we got is 100% for both equations. The linear fit gives much higher error by the calculation shown, but its prediction is within a factor of 100, whereas the quadratic estimate is 1000 times too low. So based on this experiment we can deduce that the curve fitting method is good for interpolation but not for extrapolation.

4.3.3 Calculation

In the Section 3.3.3, we derived the following formula.

$$\text{Number of cliques in } G(n,p) = \sum_{r=1}^{n} (C(n,r) * p^{C(r,2)} * (1-p^r)^{(n-r)})$$

We did some experiments to test this probabilistic formula. This Equation gives not only the expected total number of maximal cliques, but also the expected number for each size *r*. We generated three random graphs G(1000, 30) and counted the number of maximal cliques of various sizes to compare with the expected numbers. Table 29 shows that the expected number of cliques for each size is roughly correct. Summing the predicted values, as indicated by the Equation, gives a value for the total number of maximal cliques which is within a few percent of the experimentally determined value. Figure 30 shows the total number of maximal cliques for G(n, p) for a wide range of values of n along with curve going through the predicted values. The experimental values lie very close to the predicted curves, supporting the validity of this Equation.

Table 29 Average number of maximal cliques of various sizes and the predicted numbers for the Graph G(1000,30).

Clique Size *r*	**Average ± SD**	**Expected**	**Percent Error**
2	(0±0)	0	0%
3	(0±0)	0	0%
4	(8.88±0.04)E+03	9.16E+03	3%
5	(4.12±0.02)E+06	4.33E+06	5%
6	(1.00±0.004)E+07	9.51E+06	5%
7	(1.92±0.02)E+06	1.64E+06	15%
8	(6.85±0.15)E+04	5.17E+04	25%
9	(6.15±0.45)E+02	3.91E+02	36%
10	(1±0)	0	100%
11	(0±0)	0	0%
	(1.61±0.01)E+07	**1.55E+07**	**4%**

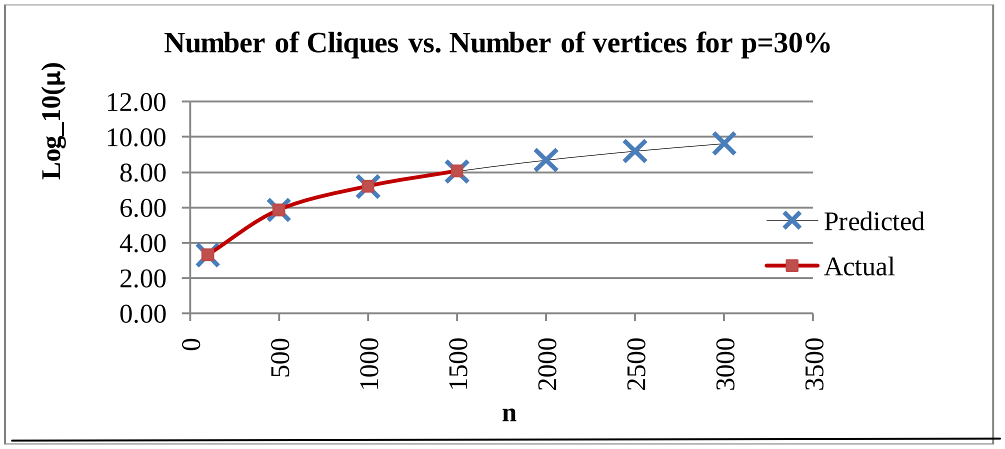


Figure 30 Log of number of maximal cliques vs. number of vertices

To check the correctness of our formula, we applied the same formula to a few more graphs. Table 30 shows predicted number of cliques and percent error in prediction for various graphs having various sizes and densities. Here also we applied the same approach of taking the average of the number of cliques in three random graphs to check correctness of our formula.

Table 30 Expected number of maximal cliques

	Predicted μ	**Prediction Error**
G(100,70)	3.92E+05	1%
G(500,40)	6.55E+06	4%
G(1000,30)	1.55E+07	4%
G(2000,20)	1.87E+07	3%
G(10000,3)	3.74E+06	1%

4.4 Chapter Summary

In this Chapter, we presented experiments conducted to test methods that we discussed to estimate the numbers and size of cliques in random graphs. Detailed explanation of all experiments was provided along with the graphical representation of results with various scenarios used. We also compared and evaluated several versions of the BK algorithm outlined and discussed the type of graph best suited for each version. Further discussion is in Chapter 5.

CHAPTER 5
CONCLUSION

5.1 Summary

In this thesis, we presented our proposal to improve the BK method, motivated by noticing possible weaknesses in this algorithm. We proposed three different algorithms MBK_2, MBK_3 and MBK_4, showing how each one differs from BK and MBK. We compared the BK and MBK methods for random graphs having various densities and concluded that the BK is preferable for dense graphs and MBK is better for sparse graphs. MBK_2 is efficient for all densities since it switches between BK and MBK based on density. MBK_3 signifies in BK's method of searching for the first candidate having maximum degree. The improvement achieved by the MBK_3 is small around (2%-8%). Another method MBK_4 is derived by modifying the MBK method in order to select the subgraph having maximum size for clique detection first, by arranging the input array (CANDIDATE) in descending order according to their degree. Though the MBK_4 method is better than MBK for interval graphs having very low density, it is worse for random graphs of high densities.

We also have identified the characteristics of cliques such as by doing various experiments. To detect each characteristic we modified the MBK program, although for most experiments the BK program would also serve the purpose. We obtained following information as clique characteristics.

- We showed that the clique detection rate is almost constant for each time interval (here we took 6 second intervals).

- We determined the frequency of the sizes of cliques and showed how it changes while increasing the density or size of input graphs.

- The modal size of clique is the size of most frequent cliques. There are a few where experiments we find the behaviour of modal size while increasing the input graph size or density. These experiments show that the modal size increases rapidly while increasing graph density but very slowly while increasing graph size.

- In one of our experiments we show that a small graph having high density gives more cliques than a large graph having low density.

We designed and tested three methods for estimating the number of maximal cliques in a random graph: sampling, calculation and curve fitting. Of these three, the method which appears to be most quick and accurate is the calculation based on expected value. This method also has the advantage that it takes negligible time compared to the others, since it involves no counting of cliques. In all the cases we tested, from n = 100 to 10,000 and p = 3% to 70%, the calculated number of cliques was within 5% of the measured average value (Note however, that we tested only a few graphs in this range). This approach also gives a good approximation of the number of maximal cliques of various sizes. The sampling approach has the advantage that it gives an estimate based on the particular graph at hand, which conceivably might not be typical of a random graph. The accuracy here increases with the number of samples, but of course so does the amount of time invested. The method based on curve fitting to collected data, requires negligible time for new predictions after the curve fitting is done, but it seems to be good only for interpolation, not for extrapolation. The result here of course would be better if a better function were used for the curve fitting.

5.2 Future Work

By utilizing the findings and results of this thesis, it is feasible to use modified algorithms for clique detection according to their sizes and densities. It is also preferable to use clique estimation techniques where detail of cliques is not required. However, by considering important observations made during the experiments we observed, a few directions for potential future work along these lines would be pointed out.

MBK_3 modifies BK in order to save time when finding first vertex having maximum degree, whereas MBK_4 modifies MBK by re-arranging the vertices of input graph. Both changes improve the performance for some of the graphs. It might be a good idea to combine MBK_4 and MBK_3 to improve performance against MBK_2.

In addition to the first suggestion described above, we can also add the functionality suggested by Tomita for clique printing. Perhaps the whole combination (MBK_3 and Tomita's strategy for clique printing) serves better performance against all other modifications.

We studied the characteristics of cliques in random graphs. In the future, it would be of interest to apply similar techniques on noisy interval graphs.

In future work, we will determine the limitations of the formula derived by probabilistic arguments. It would be interesting to answer the question "How many vertices are required for a given graph so that the estimation error is in particular range?"

REFERENCES

[1] R. Agrawal and R. Shrikant, “Fast Algorithms for Mining Association Rules in Large Databases”, *Proceedings of the 20th International Conference on Very Large Data Bases,* pp 487 – 499, 1994.

[2] G. Augustson and J. Minker, “An analysis of Some Graph Theoretical cluster Techniques”, *Journal of the association for Computing Machinery*, Vol 17, No.4, pp 571-588, 1970.

[3] E. Bierstone, “Cliques and generalized cliques in a finite linear graph”, Unpublished report.

[4] B. Bollobas and P. Erdos, “Cliques in random graphs”, *Math. Proc. Camb. Phil. Soc. 80*, pp 419-427, 1976.

[5] R.E. Bonner, “ On Some Clustering Techniques”, *On Some Clustering Techniques.*Vol 8, pp 22-32 ,1964.

[6] C. Bron. and J. Kerbosch, “Finding All Cliques of an Undirected Graph”. Communications of the ACM, Vol 16, pp 575-577, 1973.

[7] K.C. Dukka Bahadur , E. Tomita , J. Suzuki, K. Horimoto and T. Akutsu, “ clique based algorithms for protein threading with profiles and constraints”, *Proceedings of the 3rd Asia-pacific Bioinformatics Conference”,* pp 51-64, 2004.

[8] M.R. Garey and D.S. Johnson, “Computers and Intractability: A Guide to the Theory of NP-Completeness”, Freeman, San Francisco, 1979.

[9] E.R. Harley, “Graph Algorithms for Assembling Integrated Genome Maps”, *Ph.D thesis*, University of Toronto, 2003.

[10] E.R. Harley, "Comparison of Clique-Listing Algorithms", *In Proceedings of the International Conference on Modeling, Simulation and Visualization Methods (MSV'04),* Las Vegas, Nevada, USA, pp 433-438, 2004.

[11] E.R. Harley, A. Bonner, and N. Goodman, "Uniform Integration of Genome Mapping Data Using Intersection Graphs", *Bioinformatics* Vol. 17, pp 487-494, 2001.

[12] H.C. Johnston , "Cliques of a graph-variations on the Bron-Kerbosch algorithm", *International Journal of Parallel Programming,"* pp 209-238, 1975.

[13] I. Koch, "Enumerating all connected maximal common subgraphs in two graphs", Theoretical Computer Science 250, pp 1–30, 2001.

[14] R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins, " Trawling the Web for emerging cyber-communities**"** *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Vol 31 , pp 1481-1493, 1999.

[15] E. Loukakis and C. Tsouros, "A Depth First Search Algorithm to Generate the Family of Maximal Independent Sets of a Graph Lexicographically", *Computing*, Vol 27, No 4, pp 349-366, 1981.

[16] R. D. Luce and A.D. Perry, "A Method of Matrix Analysis of Group Structure", *Psychometrika* Vol 14, pp 95–116,1949.

[17] K. Makino and T. Uno, "New algorithms for enumerating all maximal cliques", *Proc. 9th Scand. Worksh. Algorithm Theory* (SWAT 2004), pp. 260.272, 2004.

[18] T. Matsunaga, C. Yonemori, E. Tomita and M. Muramatsu, "Clique-based Data Mining for Related Genes in a Biomedical database", *BMC Bioinformatics 2009,* Vol 10, 2009.

[19] J.W. Moon and L. Moser, “On cliques in graphs", *Israel Journal of Mathematics* 3,pp 23–28,1965.

[20] E. Tomita, and T. Kameda, “An Efficient Branch and Bound Algorithm for Finding a Maximum Clique with Computational Experiments”, *Journal of Global Optimization,* Vol 37, pp 95-111, 2007.

[21] E. Tomita, Y. Sutani, T. Higashi, S. Takahashi and M. Wakatsuki, “A Simple and Faster Branch-and-Bound Algorithm for Finding a Maximum Clique”, *WALCOM: Algorithms and Computation,* pp 191-203, 2010.

[22] E. Tomita, A. Tanaka, and H. Takahashi, “The worst-case time complexity for generating all maximal cliques and computational experiments", *Theoretical Computer Science 363,* pp 28–42, 2006.

[23] S. Tsukiyama, M. Ide, H. Ariyoshi and I. Shirakawa, “A New Algorithm for Generating all the Maximal Independent Sets”, *SIAM J. Comput.*, Vol 6, pp 505–517, 1977.

[24] L. Wan, B. Wu, N. Du, Q. Ye and P. Chen, “A New Algorithm for Enumerating All Maximal Cliques in Complex Network”, *Advanced Data Mining and Applications,* pp 606-617, 2006.

[25] S. Winogard and D. Coppersmith , “Matrix Multiplication via arithmetic progression”, *Journal of Symbolic Computation* Vol 9, pp 251-280, 1990.