

Finding the Maximum Edge K -Clique over a K -Partite Graph: A Stochastic Approach using Dynamic Programming

by

Faria Tasnim Tanim (181014062)

*Thesis report (CSE499) submitted in partial fulfillment of the
requirements for the degree of*

Bachelor of Science in Computer Science and Engineering

Under the supervision of

Dr. Nafees Mansoor



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY OF LIBERAL ARTS BANGLADESH**

FALL 2022

© *Faria Tasnim Tanim*
All rights reserved

DECLARATION

Project Title	Finding the Maximum Edge K -Clique over a K -Partite Graph: A Stochastic Approach using Dynamic Programming
Author	<i>Faria Tasnim Tanima</i>
Student ID	181014062
Supervisor	Dr. Nafees Mansoor

I declare that this thesis report entitled *Finding the Maximum Edge K -Clique over a K -Partite Graph: A Stochastic Approach using Dynamic Programming* is the result of my own work except as cited in the references. The thesis report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Faria Tasnim Tanima
181014062

Department of Computer Science and Engineering
University of Liberal Arts Bangladesh

Date: February 13, 2023



Department of Computer Science and Engineering
University of Liberal Arts Bangladesh
Mohammadpur, Dhaka - 1207

CERTIFICATE

This is to certify that the thesis report entitled **Finding the Maximum Edge K -Clique over a K -Partite Graph: A Stochastic Approach using Dynamic Programming**, submitted by **Faria Tasnim Tanim** (Student ID: 181014062) is a undergraduate student of the **Department of Computer Science and Engineering** has been examined. Upon recommendation by the examination committee, we hereby accord our approval of it as the presented work and submitted report fulfill the requirements for its acceptance in partial fulfillment for the degree of *Bachelor of Science in Computer Science and Engineering*.

Dr. Nafees Mansoor, Associate Professor

Dept. of CSE, University of Liberal Arts Bangladesh

Bijan Paul, Senior Lecturer

Dept. of CSE, University of Liberal Arts Bangladesh

Sabrina Jesmin, Lecturer

Dept. of CSE, University of Liberal Arts Bangladesh

Niaz Ashraf Khan, Lecturer

Dept. of CSE, University of Liberal Arts Bangladesh

Place: Dhaka

Date: February 13, 2023

ACKNOWLEDGEMENTS

It is my privilege that I have completed my research under the supervision of *Dr. Nafees Mansoor*. His invaluable energy and direction throughout my journey is incomparable. His confidence in my work and guidance in every phase have left an enduring belief in me which will help me to go a long way to accomplish the goal of my life. I could not have achieved this without his supervision as he showed different methods and working steps for accomplishing the task till the end. Therefore, I am honored and would like to convey my sincerest appreciation towards him.

I am grateful to my respected faculty members of the University of Liberal Arts Bangladesh and all the instructors and teachers who have guided me throughout my academic journey. I have grown and been better as a person due to their invaluable teaching and advice in different fields.

My parents deserve the most credit for their empathy, sacrifices and never ending kind support for my education and preparing me for creating a successful career. My elder sister has always guided me to choose the right path and make a successful life. Without the support of my family I would not be able to come this far.

Finally, unconditional support and kindness from the people of surrounding helps a person a lot to progress and gain more in life, this is why, I want to convey my heartfelt gratitude to everyone who have helped me either directly or indirectly to accomplish the research work.

Faria Tasnim Tanima

University of Liberal Arts Bangladesh

Date: February 13, 2023

Dedicated to
my loving parents and elder sister
– *Faria Tasnim Tanim*

ABSTRACT

This study presents a theoretical development of a stochastic algorithm using the features of dynamic programming for finding maximum edge k -clique over a k partite graph. Dynamic programming technique includes breaking down a larger problem into smaller sub-problems, storing it for getting the required optimal solution and reducing the repetitive computation. The algorithm has been developed based on major four considerations which are sort, divide, compare and conquer. It has been developed focusing on the bi-clique solve first which are the sub-problems. Storing the result of bi-clique or sub-graphs over the given k partite graph helps to form a maximum edge k -clique. By using the stored result of the bi-clique which are the divided problems, the optimal solution of k -clique can be reached as the solutions of bi-cliques are the subset of the ultimate formed k -clique. The new algorithm uses bi-clique solve function over two different sets of vertices forming the bi-partite graph from the given k partite graph; each time starting from the smallest set of vertices and in an ascending order considering the cardinality of different sets. Memoization enables a new form of forming a maximum edge k -clique by using the sub-solutions of bi-clique.

Keywords: clique; sub-graphs; maximum edges; graph theory; dynamic programming; k partite graph; k -clique; bi-cliques; bipartite graph

Contents

1	Introduction	1
1.1	Background Study & Motivation	1
1.2	Problem statement	4
1.3	Objectives	4
1.4	Significance of the Study	4
1.5	Scope and Limitations of the Study	4
2	Literature Review and Analysis	6
2.1	Literature Review	6
2.2	Literature Analysis	8
3	Methodology	13
3.1	Development Methodology	14
3.2	Development Stages	15
4	Proposed Algorithm	17
4.1	Context of Dynamic Programming	17
4.2	Preliminaries	18
4.2.1	k Partite Graph	18
4.2.2	k -Clique	18
4.2.3	Maximum Edge k -Clique	18
4.3	Algorithm	18
4.3.1	Algorithmic Fundamentals	19
4.3.2	The k .CLQ Algorithm	19
4.3.3	Algorithmic Description	21
4.3.4	Graphical Representation	23
4.3.5	Observations	25

5	Result analysis	26
6	Conclusion	28
6.1	Summary	28
6.2	Future Works	29
	References	30

List of Figures

1.1	Bi-partite Graph; ($k=2$)	2
1.2	Bi-Clique	3
1.3	Different Field of Application	5
3.1	Traditional Waterfall model improved upon iterative feedback of Royce's	13
3.2	Development Steps	14
4.1	k -Partite Graph; $k=3$	23
4.2	Bi-Clique Solve(Remark 1)	23
4.3	Bi-Clique Solve(Remark 1)	24
4.4	Bi-Clique Solve(Remark 3)	24
4.5	Maximum Edge k Clique(Remark 4)	24
5.1	k -Partite Graph; $k=3$	27

List of Tables

2.1	Previous Algorithms in the field of Clique Problem . . .	7
-----	--	---

Chapter 1

Introduction

This research aims to accomplish a theoretical development of a new algorithm to find maximum edge k -clique over a k partite graph. The subject of identifying k -cliques consisting maximum edges from a k -partite graphs has numerous applications in the field of science and engineering [Mirghorbani & Krokhmal \(2013\)](#). Many cross-field applications can be derived from an efficient sub-optimal maximal edge k -clique method. The clique finding algorithms are very significant and are used in different fields which consist different instances and parameters where finding out the common instances carrying maximum connections with other instances helps to take major decisions and conclude significant results.

1.1 Background Study & Motivation

The Maximum Clique Problem (MCP) is stated as one of the fundamental problems. Major fields where MCP have been used are discrete mathematics, bio-informatics, computer science, data mining, operations research, and related subjects that involves finding the maximum clique within a graph [Mirghorbani & Krokhmal \(2013\)](#).

Many real-world problems from numerous domains, including biology, assembly line management, and supply chain management, can be represented using k -partite graphing. In fact, many multivariate or multi-parameter decision problems may be analyzed and optimized using k -partite graphing, especially by attempting to solve the problem's maximal k -clique. One of the powerful unique challenges in computer science is the k -clique problem which plays a significant role in parameterized complexity [Vassilevska \(2009\)](#).

From a larger graph, if some vertices and edges form another graph then the later

is called a sub-graph of the prior graph which is larger in size. Basically, sub-graph consists vertices and edges which are subset of a bigger graph. A clique is the graph which is formed over a graph and represents another complete sub-graph where the subset of vertices are all adjacent to each other. For a partite graph, the subset of vertices are all connected with each other in different vertices set but does not have any adjacency with the vertices in the same set.

Given a k partite graph where k is a positive integer denotes the partition number of disjoint vertices, a k -clique is complete k partite sub-graph. The set of vertices which are disjoint have adjacency with other sets but no vertices within the same set have any adjacency.

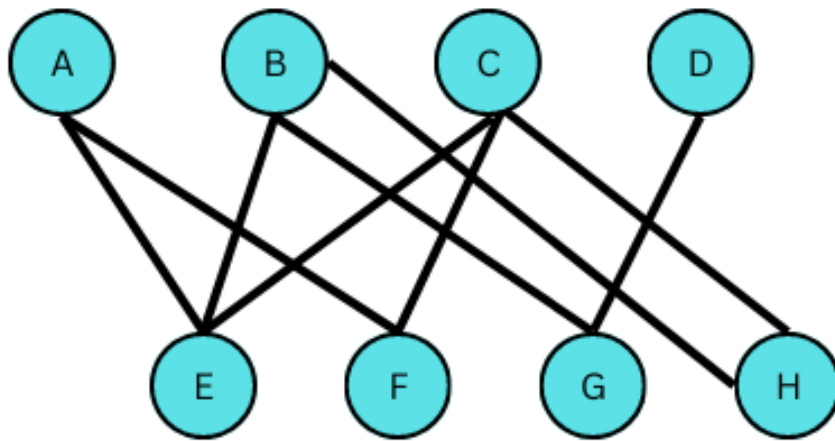


Figure 1.1: Bi-partite Graph; ($k=2$)

Figure 1.1 shows a k -partite graph where $k=2$ and thus the graph is called a bi-partite graph. In this graph $G = (U \cup V, E)$, the two sets of disjoint vertices are $U = \{A, B, C, D\}$ and $V = \{E, F, G, H\}$ where there is adjacency within the vertices of U and V . However, there is no adjacent vertices within a same set.

K-Partite Graph: A graph G having k disjoint vertices set denoting the number of sets the vertices are partitioned into. Thus, $G = (V_1 \cup V_2 \cup V_3 \cup \dots V_k, E)$.

Figure 1.2 shows the corresponding bi-clique of the bi-partite graph of Figure 1.1. Here the maximum edge bi-clique is $(\{A, E\}, \{A, F\}, \{C, E\}, \{C, F\})$ where each

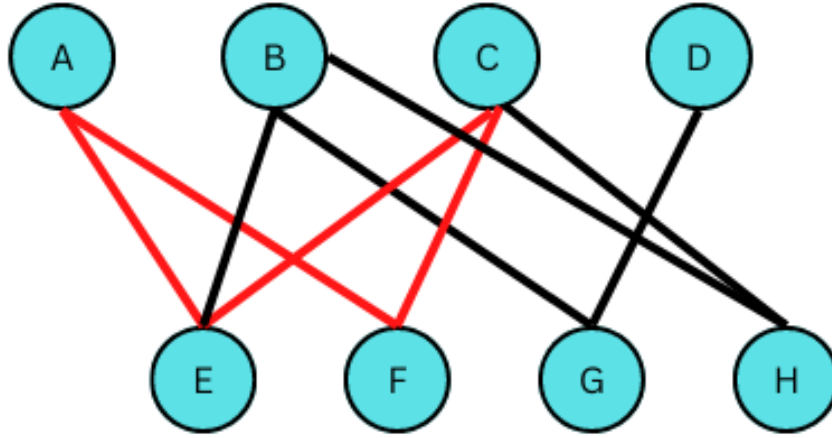


Figure 1.2: Bi-Clique

vertices from set U has adjacency with the set V .

Clique: A clique C is the subset of vertices set V i.e. $C \subseteq V$ from an undirected graph $G(V, E)$ in which every vertex of C is connected with every other vertex of C [Grünert et al. \(2002\)](#).

However, listing k -clique is often insoluble for when the size of k is greater as the number of k -cliques in a graph is exponential in the size k [Li et al. \(2020\)](#). Furthermore, in practice, the problem is exceedingly difficult to solve optimally. Problems with more than 500 nodes exceed the capability of the finest optimization techniques, depending on the nature of the graph structure [Grünert et al. \(2002\)](#). Problems of bigger and dense graphs can only be approached via heuristics, which do not guarantee the finding of the maximum number of cliques [Grünert et al. \(2002\)](#).

The wide range of applications using clique finding algorithms has been a great inspiration of this research to work particularly with developing new algorithm for clique finding in a partite graph. Moreover, as this problem is considered as one of the fundamental problems, this study tries to merge the context of dynamic programming which gives it a new dimension. Different approaches on solving fundamental problems helps to use the solution in real world data set for creating new systems and taking decisions.

1.2 Problem statement

Maximum edge k -cliques finding from a k -partite graph which is a NP-Hard problem.

1.3 Objectives

This section defines precise purposes behind this research work which are need to be achieved after the development completion and throughout the development phase.

- To find an efficient algorithm for solving k -clique problems in a k -partite graph.
- To propose an algorithm which can form k -clique over a k partite graph following divide and conquer method of dynamic programming.
- To analyze the existing algorithms & identifying strengths and limitations.
- To publish a scholarly article is the ultimate goal utilizing the proposed algorithm.

1.4 Significance of the Study

This study overviews several previous proposed algorithms as well as applications of those in real world data sets. Moreover, different constraints and categories for listing out different kind of sub-graphs has helped to define and develop what new features are needed to be taken under consideration. The proposed algorithm using a complete new concept in Chapter 4 which have not been considered in previous research works which adds nobility to this research work.

1.5 Scope and Limitations of the Study

Solving for sub-graphs from a graph helps to determine many significant roles of different characteristics from different fields. A maximal sub-graph contains those important parameters and relations which denotes significant patterns between

different objects which helps to understand network characteristics whether it is of a communication system or protein structural network.

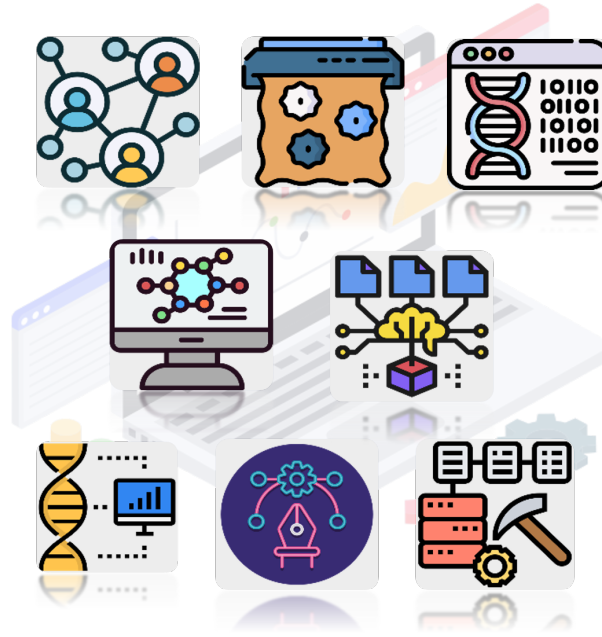


Figure 1.3: Different Field of Application

Real worlds applications of clique solving problem i.e. networking, data mining, discrete mathematics, bio-informatics etc. has already shown how clique solving algorithms can be used in major fields. The major concern is listing out cliques is not easy to solve. There are many constraint to apply an efficient algorithm on the fundamental study of K -clique which includes extreme time complexity for larger graphs. To develop a new algorithm in the sector of maximum clique solving it needs rigorous comparison with different size graph instances which is the further scope. However, if a near to optimal algorithm can be developed many real world decision making on multi-variant environment is possible.

Chapter 2

Literature Review and Analysis

This section covers the overall summary of related works on clique problem solutions, i.e. bi-clique finding from partite graphs, finding k cliques and listing different sub-graphs. In the section 2.1, different applications on real life scenarios of clique finding has been discussed . Furthermore, different algorithms that has been proposed is summarized mentioning method and time complexity.

2.1 Literature Review

Significant application in the field of community finding and social network research consists of listing all K -cliques in a graph which is considered as a primary graph mining operator [Li et al. \(2020\)](#). Different social network connections, networks driven from biological data, communication networks show structure of a tenacious sub-graph which is very fundamental to extract to complete further network analysis in data mining communities [Li et al. \(2020\)](#). This is where the k -clique structure plays an important role as it is considered as the most primary cohesive substructure within a graph and has a wide range of applications. Several applications in the field of science and engineering has already been conducted using the subject of identifying k -cliques in k -partite graphs [Mirghorbani & Krokhmal \(2013\)](#). Many cross-field applications will derive from an efficient sub-optimal maximal edge k -clique method. Some of the significant applications as well as research have been done to find k -cliques in k -partite graphs which include the braiding problem from textile industry [Grünert et al. \(2002\)](#), protein structures identification [Liu et al. \(2009\)](#), and multidimensional assignment problem(MAP) of high range randomized variables [Krokhmal et al. \(2007\)](#).

Table 2.1: **Previous Algorithms in the field of Clique Problem**

Proposed Algorithm	Input	Output	Followed Method
Shaham et al. (2016)	Bi-Partite Graph	List of maximal bi-cliques	Based on Monte-Carlo Strategy a Bi-clique Subspace Clustering Method
Zhang et al. (2008)	Bi-Partite Graph	All Maximal Bi-cliques	Improved Maximal Bi-clique Enumeration Algorithm (iMBEA)
Tomita et al. (2006)	General Graph	Maximal Cliques	Maximal clique finding in general graphs
Makino & Uno (2004)	Bi-Partite Graph	Maximal Bi-cliques	Maximal clique finding in general graphs
Zaki & Hsiao (2002)	Bi-Partite Graph	Maximal Bi-cliques	Mining Method in closed item-set
Alexe et al. (2004a)	General Graph	Maximal Bi-cliques	A Consensus algorithm
Mushlin et al. (2007)	Bi-Partite Graph	Maximal Bi-cliques	Exhaustive search with a priority queue
Sanderson et al. (2003)	Bi-Partite Graph	Bounded Maximal Bi-cliques	Exhaustive search by iterative bi-clique building
Chiba & Nishizeki (1985)	General Graph	List all cliques and complete sub-graphs	Mining Method in closed item-set
Vassilevska (2009)	General Graph	Clique of Size k	Combinatorial Algorithm
Li et al. (2020)	Graph	Listing k -cliques	Depth-first Backtracking(MACE Algorithm)

From the previous research and findings different categories of proposing clique solving algorithm has been found.

There are different kind of sub-graphs which forms different geometrical shapes [Chiba & Nishizeki \(1985\)](#). In the field of communications network both fastest and densest connection needs to be identified which can be solve with clique solving approach. In the protein structure identification, each vertex represents a protein and the interaction of different proteins within themselves is presented as an edge [Liu et al. \(2009\)](#). Thus from this interaction networks protein cliques have been fetched. Most of the research has stated maximum clique problem as NP-hard problems [Wu & Hao \(2015\)](#). Depth first search, brand and bound, edge searching, node mining etc. different techniques have been used. Rather than finding maximum edge k clique, there are different algorithms on densest k sub-graphs [Feige et al. \(1997\)](#). The crucial fact about clique finding is that if an algorithm focuses on maximum clique finding it may has some inconsistent result as it is not definite that a graph will only have one maximum clique [Eblen et al. \(2011\)](#).

2.2 Literature Analysis

Different studies has considered different context of k -cliques. Moreover, majority of the studies are on solving bi-clique problem from a bi-partite graph. Solving for k - clique has been shown on some studies considering different parameter like arboricity, degeneracy of a graph, time bound on size, limiting the size of k etc. This section analyzes previous proposed algorithms from previous studies and the method that has been used. It is important to mention that majority of the algorithms are based on branch and bound and depth first backtracking and the further studies shows result of improved algorithm. Different features of the proposed algorithms and their complexity and performance are analyzed in this section.

A Review on Algorithms for Maximum Clique Problems

[Wu & Hao \(2015\)](#) has stated numerous real life application of clique problem which shows the significance of this particular field of graph theory. The problem is similar with combinatorial optimization problem. Most of the effective method has been designed using brand and bound technique. As the problem gets extremely complicated regarding space as well as time complexity with increasing size of

the graphs it becomes necessary to bound or limit some parameter of the graph. To at-least gain sub-optimal solution within an proper time the accepted methods are heuristic and meta-heuristic which shows good results for large graphs. The clique search method can be improved by merging above mentioned categories. Different algorithms which has been formed with heuristic approaches are local search with given size of k , greedy algorithm where nodes are iteratively added until no adjacency is found, fixed search space in local search with legal strategy.

Arboricity and Subgraph Listing Algorithms

Chiba & Nishizeki (1985) focused on solving certain kinds of sub-graph listing problems and proposed different algorithms for this. A clique can represent different kind of shape based sub-graph as well which looks like "triangle" having three vertex, "quadrangle" having four vertex and adjacency, "complete sub-graph of a fixed order" which is a subset of main graph and "clique" which is the maximal complete sub-graph. They introduced a novel simple and effective technique to edge various types of sub-graph listing problems.

Listing all the triangles from a Graph takes time of $\mathcal{O}(a(G)m)$ delay. In this time delay all the triangles is listed using their proposed algorithm. In this algorithm, $a(G)$ is the arboricity of the graph G . To find all the quadrangles their proposed algorithm takes $\mathcal{O}(a(G)m)$ time. They have also proposed an algorithm having $\mathcal{O}(la(G)^{l-2}m)$ time complexity which lists all the complete sub-graphs K_l of order l . Lastly, they dealt with cliques listing problems in which their algorithm achieved $\mathcal{O}(a(G)m)$ time per clique. The algorithms that have been considered for listing different sub-graphs takes linear space. In order to limit the running time of algorithms, they have employed the fairly strange graph invariant parameter $a(G)$. Arboricity means the disjoint edges of G of the minimum number of forests where G can be partitioned. The new strategy scans the edges of the selected vertex and fetched the sub-graph from selected vertex v in which all the sub-graphs are unduced of v .

Efficient Algorithms for Clique Problems

There have been different approaches to solving the k -clique problem. In a 2009 paper by Dr. Virginia Vassilevska, it was stated that combinatorial, non-algebraic algorithms which are proposed are considered more practical solutions than algorithms which are developed upon algebraic fast matrix multiplication, even though the latter provide theoretically better asymptotic gains. The proposed algorithm in

that paper has $\mathcal{O}(n^K / (\epsilon \log n)^{K-1})$ time for every fixed $k \geq 3$ and $\mathcal{O}(n^\epsilon)$ space, for any constant $\epsilon > 0$, independent of k [Vassilevska \(2009\)](#).

They studied a combinatorial method in small-space in a node weighted graph using the technique of max weight sum that discovers a k -clique. The task of finding maximum node-weighted $(k - 1)$ cliques have been simplified with the algorithm's core idea which finds large number of tiny $\mathcal{O}(\log n)$ size sub-graphs, and then aim to complete such cliques by adding an extra vertex. The method iterates so that each iteration reuses the space occupied by the preceding one. Their proposed algorithm is considered as space-efficient which outperforms the existing best combinatorial techniques on discovering k -clique, and it includes an algorithm within the same runtime which can identify maximum node-weighted k -clique.

There is the MACE (Maximal Clique Enumerator), which makes use of a depth first backtracking procedure for maximal k -cliques listing within a graph. To list a clique, this algorithm takes which at most $\mathcal{O}(n)$ time. They have classified existing algorithms in two different categories which are exact algorithms and approximation algorithms. the key idea of MACE has been formulated where to obtain k clique a recursive process is done adding vertex to a sub-clique i.e. $C \cup v$.

On Finding k -cliques in k -partite Graphs

The proposed algorithm 'BitCLQ' based on branch and bound algorithm, in this paper, is improved upon the 'FINDCLIQUE' algorithm of [Grünert et al. \(2002\)](#) which uses bit-sets and the main data structures bit parallelism to improve computational procedure for finding k cliques in a k partite graph.

Binary vectors or sequential bits employed as bit-sets are used in computer codes to effectively store adjacency matrices of graphs or sets. In different algorithms of computation string matching, combinatorial problems i.e. in SAT bit parallelism has been applied as a form of parallel computing to get improved result in computational complexity. The proposed method of the algorithm follows the branch and bound technique. In each step it achieves a partial solution which is added to a variable S or discarded from S . To achieve a faster backtracking and efficiency in memory storage bit-sets helped to develop their algorithm. The use of bit-sets and the accompanying bit parallelism reduces the computational cost in the branch-and-bound technique. Different other techniques are also mentioned in different studies.

On Finding the Maximum Edge Biclique in a Bipartite Graph: A Subspace Clustering Approach

Using Monte-carlo subspace clustering approach [Shaham et al. \(2016\)](#) have proposed a bi-clique subspace clustering algorithm based on heuristic method to find the maximum edge bi-clique. The paper stated 3 categories for the algorithms which are tackling the clique solving problem. These categories falls under characteristics bounding to relax the problem, exploiting some classes and domain reduction. Size of the bi-cliques can be bound upon first category as well as arboricity of the graph and bounding vertices degree. The second category which is exploitation of classes includes limitation to convex and chordal bipartite graphs. Maximum clique reduction and reducing frequent item sets falls in the last category. The method of the proposed algorithm initializes on a random selections from the adjacency matrix list which is a set of rows. Then a column is accumulated which complies the seeding row. This steps repeat till a probabilistic solution. The run time overall for the algorithm is $mn^{\mathcal{O}(1)}$ which is polynomial.

Consensus Algorithms for the Generation of All Maximal Bicliques

From the paper [Alexe et al. \(2004b\)](#) shows, the primary consensus algorithm run time is exponential in in the size of the input graph and in the number of its maximal bi-clique. Improved upon the basic consensus algorithm, the new version modular consensus algorithm which follows a simple structure of adding a maximal bi-clique to the current list C in each operation. Different version of consensus algorithms show polynomial run time. The study proposed an algorithm which can generate sub-graphs which are complete bipartite and maximal in size. They also stated the problem as NP-hard. The key features of their proposed algorithm are: (i) absorption which is based on choosing the bi-clique from the list of C which can absorb another bi-clique and (ii) the second criteria is consensus adjunction where the list of C is updated with different bi-cliques is those bi-cliques are not absorbed by another bi-cliques.

On Finding Bicliques in Bipartite graphs: A Novel Algorithm and its Application to the Integration of Diverse Biological Data Types

[Zhang et al. \(2008\)](#) stated that finding maximum bi-cliques in bipartite graphs can help with a number of biological problems. The vertex maximum bi-clique problem and the edge maximum bi-clique problem are two separate variations of the maximum bi-clique problem that have been shown. Maximum bi-clique with maximum

vertices can be solved in polynomial time and asks us to locate a bi-clique with the most vertices. The second one with the maximum edges is NP-complete and requires us to locate a bi-clique with the most edges. the worst-case time complexity of this $\mathcal{O}(2^n mn)$. The second variation is highly applicable in different fields of biological diverse problems as the bi-cliques found with maximum edges are more balanced considering connectivity. Maximum bi-clique can be enumerated in $\mathcal{O}(a^3 2^{2a})$.

On the Complexity of Fixed Parameter Clique and Dominating Set

[Eisenbrand & Grandoni \(2004\)](#) stated that, theory of complexity consists of problems like *clique* and *dominating set*. They proposed a new algorithm which have running time of $\mathcal{O}(n^{\omega(r,s,t)})$ which is of multiplication matrix. A clique and a dominating set having fixed parameter is possible to tract and based on reduction procedure many other parameterized problem is possible to tract. Both of the mentioned problem is stated as NP-complete as well in some studies it is shown as NP-hard. In parameterized complexity the complexity of the problems mainly generates from the input.

Using Symbolic Techniques to find the Maximum Clique in Very Large Sparse Graphs

[Corno et al. \(1995\)](#) stating the clique problem Np-complete said that with the increasing number of nodes or a graph already with bigger size can not adapt with maximum clique finding algorithms. The core function of the proposed algorithm uses Boolean function which can encode the connected nodes of a graph. Neither enumeration problems nor symbolic approaches are efficient for large graphs because the former faces problem with enumeration and the later faces encoding complexity. To solve the issue they proposed an algorithm merging BDDs and heuristic to resort the previous mixed symbolic explicit approach.

Chapter 3

Methodology

This thesis completion utilizes the water fall model model sequence which is improved by Royce.

The traditional Waterfall model is a linear process where one stage moves on to the other after completion without any reversibility. But an iterative model helps to revisit the stages based on the needs. To enhance the outcome of each stage and reduce design difficulties Royce [Ruparelia \(2010\)](#) improved this model by adding a feedback loop so that every stage can get back to the previous one if there are any complications so that final deployment does not include any complexity which can not be revisited. Due to this feedback loop stages can be revisited if the result is inadequate. In Fig 3.1, the red arrows denotes feed-backs from the current stage and the gray arrows denotes proceeding to next stage.

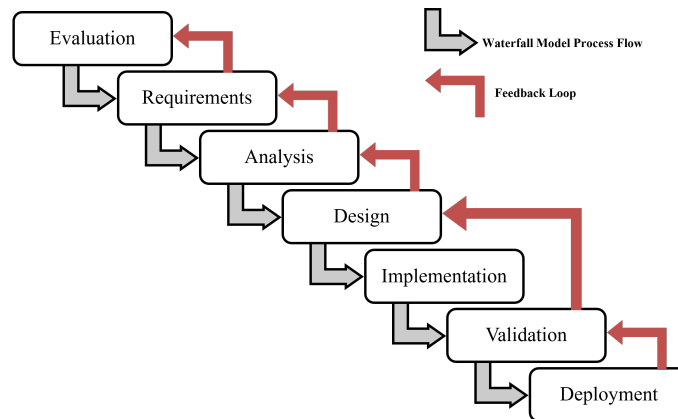


Figure 3.1: Traditional Waterfall model improved upon iterative feedback of Royce's [Ruparelia \(2010\)](#).

3.1 Development Methodology

Following Royce's iterative feedback on the waterfall model, the entire thesis model is developed, and the necessary steps till development are announced. The evaluation stage for this thesis is defining the problem and declaring the major problem statement that must be answered after deployment. The steps of requirement and analysis are described in the section on literature review and analysis. Ideate a solution is the stage in which a solution is designed based on prior insights from the literature 'review and analysis' stage. Implementation, performance analysis, and outcome analysis all seem to be combination of implementation and validation processes. The final stage is deploying the final result, and if the results are unsatisfactory, the prior stages can be reviewed. Due to this revisit facility, it becomes very easy to resolve inefficient parts of the development without removing the whole portion of progress. Throughout the whole process a report is generated and after the final delivery of the developed product overview through final report is necessary and the last step is to deploy the product and report.

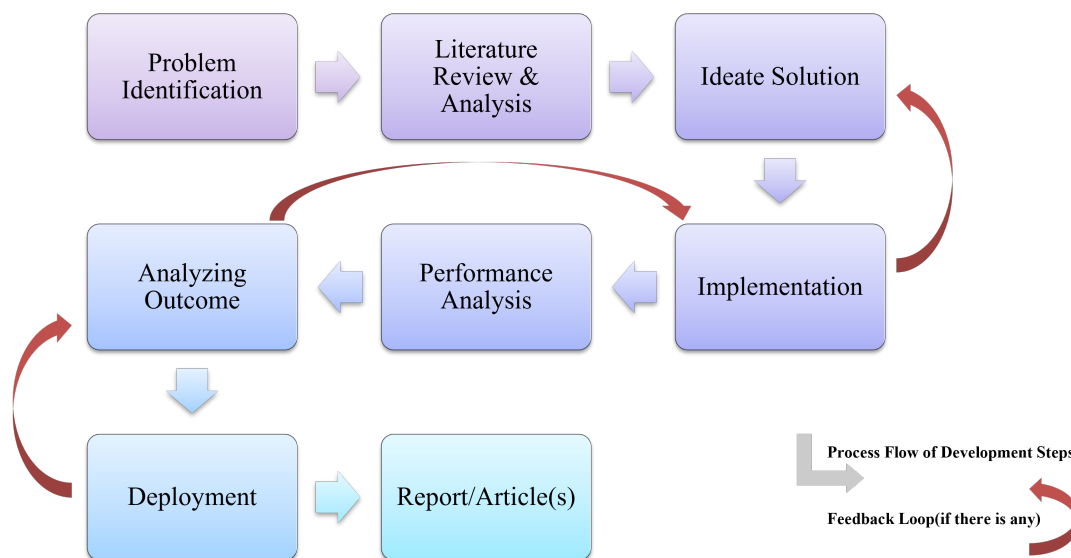


Figure 3.2: Development Steps

3.2 Development Stages

This section explains functionality of different stages throughout the completion from the figure 3.2.

- **Problem Identification:** Listing out cliques from a complete graph can help to solve many decision making problems. A complete sub-graph helps to give many efficient solutions on parameterized problem fields. Many bi-clique solving algorithms are there which are used in real world data sets. An efficient solution for finding k cliques will make a significant change in the sector of clique solving algorithms. This thesis aims to propose a new algorithm with different functionality to solve for maximum edge k clique.
- **Literature Review and Analysis:** Many research have been done on this topic. Moreover, different applications on the field of biological diversity, parameter based problems, filed of mathematics has used the fundamentals of clique problem. Based on the previous algorithms and procedures that have been stated earlier, this study focuses on proposing core functionality to find k -cliques.
- **Ideate Solution:** The main consideration is to find out maximum edge k -clique where the stochastic method is followed as the solution will not be prioritized based on any selected parameter or particular node. The choice of solution is random. There have been rigorous solutions for bi-clique finding rather than k -clique solutions. The approaches on solving k clique is minor. This paper proposes a new algorithm considering the functionality of dynamic programming to add a new dimension on clique solving approaches.
- **Implementation:** In this stage considering the pre-defined, this study develops a new algorithm to form K -clique. The proposed algorithm should be able to solve the declared problem statement. Moreover, the implementation phase should cover new concept and features which have not proposed before.
- **Performance Analysis:** Based upon the implementation stage, this step works to analyze if the algorithm is being able to solve the problem or not which will also include necessary feedback so that previous step can be revisited. Moreover, prior comparison result is also made in this phase.

- **Analyzing Outcomes** Based upon the performance result different outcome of the proposed algorithms is shown. As this study revolves around theoretical section, the outcome analyzing shows the discussion over proposed solution and probable outcomes for different scenarios.
- **Deployment & Report:** The final report is done after the overall steps have completed. After the deployment if there is no inadequate scenarios or elements the a scholar article can be written up.

Chapter 4

Proposed Algorithm

In this section, a thorough discussion has been covered on how the conceptual development of the algorithm is made. Later, the new algorithm referred as ' k_CLQ ' using dynamic programming is explained. By introducing dynamic programming and utilizing its features maximum edges k -clique will be formed.

4.1 Context of Dynamic Programming

Dynamic programming is an optimization technique that breaks down a complex problem into a group of smaller problems; its key feature is the multilevel structure of the optimization method and using the solution from different levels to generate required optimal solution. The solution which comes from subdivided problems helps to generate ultimate optimal solution. Dividing the main problem into different small problems makes it easier to solve the problem with lesser time and taking lesser complex.

A dynamic programming solution utilizing memoization and breaking the problem into smaller sub-problems, i.e., solving the maximal edge k -clique via solving the bi-cliques that comprise it, the order of which will be chosen through rank and size of said bi-cliques, will provide a sub-optimal efficient stochastic algorithm, a novel approach to tackling this problem. A dynamic programming solution lends itself well to the k -clique problems due to the nature of multipartite graphs leading to sometimes large k -cliques requiring large computational resources otherwise, which memoization helps combat, trading space complexity for time complexity.

4.2 Preliminaries

The graph that is considered for developing steps of algorithm is k - partite graph. The graph $G = (V_1 \cup V_2 \cup V_3 \cup \dots V_K, E)$ is undirected, unweighted, simple and each vertex has at least one connection to every other set of vertices.

4.2.1 k Partite Graph

A K -partite graph, is defined as a graph $G = (V, E)$, where V is the set of all vertex, and E is the set of all edges $E = (U, V)$; where V can be partitioned into K disjoint subsets $V_1, V_2, V_3, \dots V_K$ such that $G = (V_1 \cup V_2 \cup V_3 \cup \dots V_K, E)$ where every edge $(U, V) \in E$ is between different vertex sets, which is to say E is a set containing edges from V_i and V'_i , where i is any integer between 1 and K . It is because to make a partite graph minimum 2 sets of vertices are there which is a bi-partite graph where $K=2$ and $G = (U \cup V, E)$.

4.2.2 k -Clique

A K -clique can be defined as $K = (U_1 \cup U_2 \cup U_3 \cup \dots U_n)$ where $U_i \subseteq V_{i+m}, \forall i, 1 \leq i \leq n$, for any $n \geq 2$, and any $m \geq 0$, and for every $U_1 \in U_i$ and $U_2 \in U_i$, the edge $(U_1, U_2) \in E$.

4.2.3 Maximum Edge k -Clique

A maximum edge K -clique is denoted as $\omega(G)$ where $\omega \leq k$; and the graph induced by the ω is $G = (V_{1,2,\dots \leq k}, E)$ where every V has adjacency with every set.

4.3 Algorithm

This section presents the ' k_CLQ ' algorithm describing the algorithm for extracting a list of maximal bi-cliques. Approaching this problem through using the key features of dynamic programming, we have developed a stochastic method to solve the maximum edge k -clique problem. The problem is subdivided into a series of the more well understood bi-clique problem, and thereafter applying memoization, we solve the ranked bi-cliques ultimately leading to the solution of the k -clique.

The bi-cliques comprising the overarching k -clique are ranked based on common edges between them and other bi-cliques, and their size. This is to avoid the 'curse of dimensionality' as coined by Richard Bellman in his book Dynamic Programming,

which refers to sparsity of meaningful data in higher dimensional space.. It is stochastic in the sense that the initial starting point on where the algorithm is implemented determines the speed or efficiency of the result. The graph consists three key terms where partition number is k , the number of vertices in different vertices set is denoted by n , the number of edges is denoted by m .

The output of the ' k_CLQ ' Algorithm is a maximum sub-graph which forms a maximum edge k clique induced by all the sub-solution of bi-cliques and sub-graphs. The ' k_CLQ ' algorithm uses the stored solution of bi-cliques which is divided by forming bi-partite over the k -partite graph. This new concept in the clique solving algorithm uses iterative projection of finding bi-cliques starting form the smallest bi-partite graph. It proceeds by discarding the vertices which does not exist in the bi-clique results so in the next search for bi-clique becomes less dense for newly formed bi-partite.

4.3.1 Algorithmic Fundamentals

Let $G = (V_1 \cup V_2 \cup V_3 \cup \dots V_K, E)$ be a k -partite graph. ' k_CLQ ' operates with two sets of vertices in each iteration of bi-clique solve. Four dynamically changing sets of vertices throughout the algorithm are:

- L : set of vertices $\in U$; which contains all the common nodes which are adjacent of R , initially $L = U$
- R : set of vertices $\in V$; belongs to the current bi-clique and R is initially empty
- P : set of vertices $\in V$; that can be added to R where initially $P = V$
- Q : contains the elements which are not found common in the bi-cliques and are former candidates of bi-partite graph $G = (L \cup R, E)$.

4.3.2 The k_CLQ Algorithm

Algorithm 1 presents the pseudo-code which is a conceptual development of the algorithm. The structure of the Algorithm is divided into major 4 steps which are here explained below:

Algorithm 1 Finding Maximum Edge k Clique

```
1: bool VerticesSizeCompare(const vector < int > & first, const vector < int >
   & second)
2: return first.size() < second.size()
3:  $V_1.insert(V_1.end(), first.begin(), first.end());$ 
4:  $V_2.insert(V_2.end(), second.begin(), second.end());$ 
5: Biclique_solve( $G, L, R, P, Q$ );
6:  $G = (V_1 \cup V_2, E)$ 
7:  $L = V_1$ ;
8:  $R = V_2$ ;
9: for  $P \neq \emptyset$  do
10:    $x = P[i++]$ ;
11:    $R' = R \cup \{x\}$ ;
12:    $L' = \{u \in L \mid (u, x) \in E(G)\}$ ;
13:    $P' = \emptyset$ ;
14:    $Q = \emptyset$ ;
15:   Is_maximal = true;
16:   if Is_maximal = true then{
17:     for  $\forall v \in P, v \neq x$  do{
18:        $K[v] = \{u \in L' \mid (u, v) \in E(G)\}$ ;
19:       if  $|K[v]| = |L'|$  then
20:          $R' = R' \cup \{v\}$ ;
21:       else  $|K[v]| > 0$ 
22:          $P' = P' \cup \{v\}$ ;
23:       end if
24:       PRINT ( $L', R'$ );
25:     end for
26:   end if
27: end for
28: if  $P' = \emptyset$  then{
29:   Biclique_solve( $G, L', R', P', Q'$ );
30:    $Q = Q \cup C$ ;
31:    $P = P \setminus C$ ;
32: end if
33:  $memo[\{L, R\}] = \text{Biclique\_solve}$ ;
34: set_intersection(biclique1.begin(), biclique1.end(), biclique2.begin(), biclique2.end());
35: inserter(common_vertices, common_vertices.begin());
36: if (common_vertices.size() ==  $k$ ) then {
37:   k_cliques.insert(common_vertices);
38: end if
39: return k_cliques;
```

1. Line(1-4) is the snippet of how different set of vertices can be compared based on their cardinality. The function *VerticesSizeCompare* compares given vector sets. It returns the sets in ascending order based on the element each set contains. The sorted set of vertices will be next inserted gradually into the $V_1 \dots to V_k$.
2. From the line 5 bi-clique solve starts. As P contains the elements that can be added to R till it is not empty x gets the element from P and the new set R' gets compatible nodes. Initially the set Q is empty while no vertex is discarded from the bipartite graph. $K[v]$ stores maximum bi-clique formed with L' and R' . It returns the bi-clique. This process will repeat $(k - 1)(k/2)$ times because bi-clique solved will be required for each two partition set i.e if the number of partition $k = 5$ then the bi-partite graph that can be formed over this graph will be $[\{k[0], k[1]\}, [\{k[0], k[2]\}, [\{k[0], k[3]\}, [\{k[0], k[4]\}, [\{k[1], k[2]\}, [\{k[1], k[3]\}, [\{k[1], k[4]\}, [\{k[2], k[3]\}, [\{k[2], k[4]\}, [\{k[3], k[4]\}]$; considering that every vertex from every set have adjacency with every other set.
3. After each bi-clique solve it will be stored in *memo*. This stored results of bi-clique will do a intersection operation to return the common vertices among the adjacency.
4. The intersect result will form a definite sub-graph resulting maximum edge k -clique.

4.3.3 Algorithmic Description

The *Biclique_solve()* function finds: (i) maximum bi-cliques having pairwise adjacency, (ii) if not pairwise the node having the highest degree. The *Biclique_solve()* function uses depth-first search tree traversal. After each bi-clique find, the input graph is changed with new two sets with the neighbor list. The process stops when bi-partite is formed with every other set and all the possible bi-cliques are listed and P is empty.

After sorting different sets the new sets are in a array which will group new bipartite graphs sequentially according to the size. One of the major features of this step is it help to reduce the disconnected node from the bi-clique in the first phase of finding optimal solution. After the grouping when the bi clique solve function starts it first

looks for sub graphs which makes bi-clique if not the only nodes with maximum edges forming maximal sub graph is considered.

Lemma 1: Every sub-solution from the formed bi-partite graph is a maximal sub-graph.

Proof. Nodes in $G = (L \cup R, E)$ all subset of $G = (V_1 \cup V_2 \cup V_3 \cup \dots V_K, E)$. The divided graph from the former k -partite graph is considered to pick the bi-cliques that is maximal within the divided graph. When $[\{k[0], k[1]\}]$ is the input graph bi-cliques are picked and the nodes with less or no degree at all gets reduced and both $[\{k[0]\}]$ and $[\{k[1]\}]$ is updated. In the next search for bi-clique updated $[\{k[0]\}]$ and $[\{k[1]\}]$ is used. Thus, the remaining nodes forms new maximal sub-graph as the prior comparison already reduced the nodes with lesser degree.

Lemma 2: Updating the sets lessens the time complexity for finding new bi-cliques.

Proof. From each solution of bi-clique the compared partition sets gets updated(Figure 4.2 & 4.3). The less the number of nodes it's more space and time efficient to find bi-clique(Figure 4.4) as the unconnected nodes from the bi-clique is already reduced.

Theorem 1. ' k_CLQ ' list all maximal bi-cliques from a large graph.

Proof. ' k_CLQ ' traverse through the entire search space of all the bi-partite formed over the k -partite and store the bi-cliques by **Lemma 1**. It reduces the vertices that do not form any maximal bi-cliques within the formed bipartite graph. Therefore, upon each end of bi-clique solve, ' k_CLQ ' stores all maximal bi-cliques.

Theorem 2. C is a collection of all bi-cliques of the graph G which contains vertices that forms maximum edge k -clique.

Proof. The algorithm terminates after storing all the bi-cliques; every bi-clique of G is stored in *memo*. Further intersection among all the C returns the result of k which are in maximum edge k -clique

4.3.4 Graphical Representation

The Figures shown below presents different steps of the previously discussed algorithm and corresponding remarks.

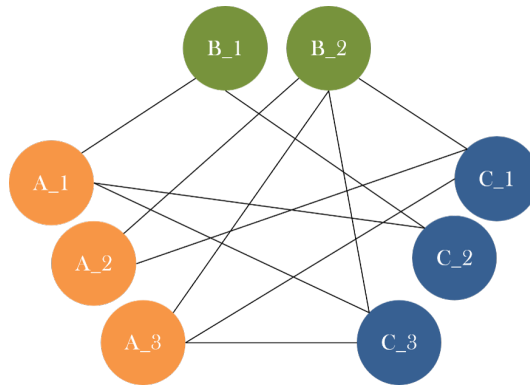


Figure 4.1: k -Partite Graph; $k=3$

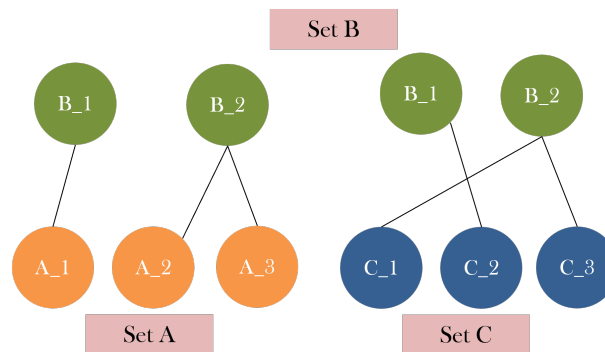


Figure 4.2: Bi-Clique Solve(Remark 1)

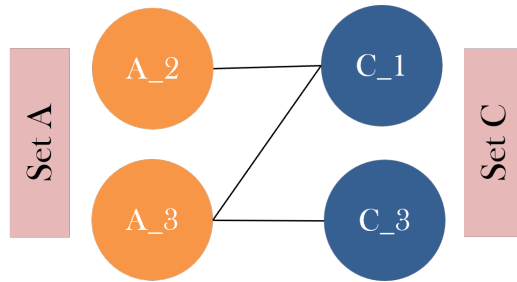


Figure 4.3: Bi-Clique Solve(Remark 1)

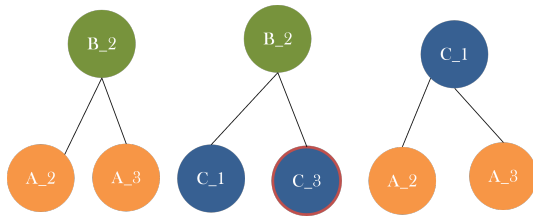


Figure 4.4: Bi-Clique Solve(Remark 3)

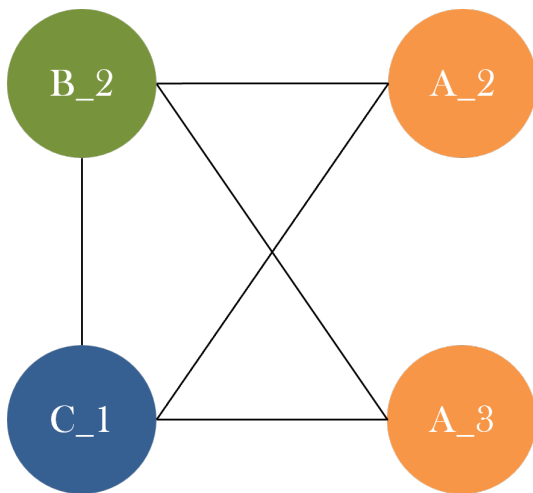


Figure 4.5: Maximum Edge k Clique(Remark 4)

4.3.5 Observations

This section now describes the observations from the proposed algorithm.

Remark 1: The result L', R' is of maximum bi-cliques from the bi-partite graph $G = (L \cup R, E)$, formed over the given k -partite graph.

The bi-clique solve function adds the nodes one by one where the returned result shows the vertices forming bi-clique and if not maximal further comparison occurs. The graph is considered undirected so the bi-clique $(\{L'\}, \{R'\})$ or $(\{R'\}, \{L'\})$ consists vertices with maximum adjacency if not any pairwise connection found.

Remark 2: Let $k[v]$ denote maximal bi-clique where candidate elements from P is gradually added.

While solving bi-clique, the comparison starts with the smallest set of vertices $k[0]$ with its neighbor $k[1]$ with its neighbor sets. $k[v]$ stores vertices that forms maximum bi-clique.

Remark 3: Set intersection of maximum bi-cliques returns common vertices which forms a new subgraph.

Chapter 5

Result analysis

This section presents the run time complexity of the proposed algorithm based on the fundamental algorithmic complexity expression.

$\mathcal{O}(n \log n)$ is the run time complexity on average for the proposed algorithm. The worst case scenario of the proposed algorithm's run time complexity is $\mathcal{O}((n^2) \log mn)$.

Example 1. Consider the graph(Figure 4.1) $G = (V_1 \cup V_2 \cup V_3 \cup \dots V_K, E)$ with vertex-set $V_1 = (A\{A_1, A_2, A_3\}; V_2 = (B\{B_1, B_2\}; V_3 = (C\{C_1, C_2, C_3\}$ edge edge set $E = \{(A_1; B_1); (A_1; C_2); (A_1; C_3)\}; \{(A_2; B_2); (A_2; C_1)\}; \{(A_3; B_2); (A_3; C_1); (A_3; C_3); \}$.

Let us consider the following bipartite graph formed from the $k - partite(k = 3)$, $G = (V_1 \cup V_2, E); G = (V_1 \cup V_3, E); G = (V_2 \cup V_3, E)$.

STEP 1: Sorting the set of vertices depending on their cardinality the algorithm takes $\mathcal{O}(n \log n)$ time where n is the number of elements in each set. For **Example 1** maximum number of elements is 3.

STEP 2: For forming every possible bipartite it takes $(k - 1)(k/2)$ time where k is the number of sets or the number of partitions.

STEP 3: The results of biclique will again go under intersection operation where each iteration will take $\mathcal{O}(\min(x, x'))$ where x is the size of first biclique and x' is for second biclique.

STEP 4: The insertion of common vertices to form k -clique operation has worst-case time complexity of $O(k \log k)$, where k is the size of the common vertices set. Overall.

STEP 5: The time complexity of forming k -clique using the stored result of biclique and after intersection is $\mathcal{O}(\min(m, n) + k \log k)$, and the space complexity is $\mathcal{O}(k)$.

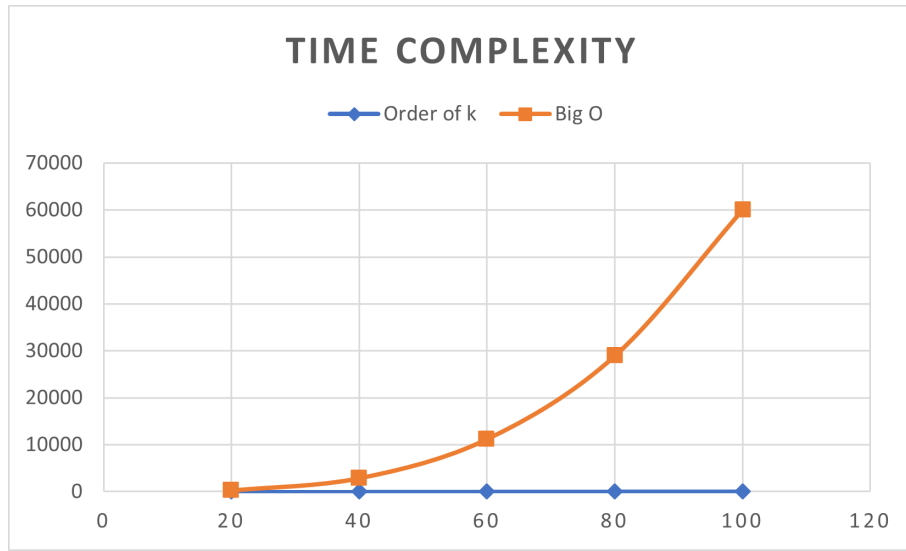


Figure 5.1: k -Partite Graph; $k=3$

The worst time complexity of the algorithm is $\mathcal{O}(\frac{(k^2-k)(n \log n)}{2})$ which is presented in the Figure 5.1 where the middle line shows the growth of complexity corresponding to input size of k ranged between 20 o 100.

Chapter 6

Conclusion

In the field of computer science and engineering the fundamental problem lists that falls under NP-complete and NP-Hard have huge attention as solving these type of fundamental problems with an optimal algorithm has numerous applications. Specially, listing different kind of sub-graphs and clique plays and significant role in real world fields where networking routes, dense network connections, common instances within multiple characteristics is important to be identified. This research adds a new category on how maximum edge k clique can be formed over the sub solutions of bi-clique.

6.1 Summary

This study is a theoretical development of an algorithmic concept which can from maximum edge k -clique. It focuses on the significance of the different application the clique solving algorithm has. Later previous related works and many algorithms on bi-clique and k -clique is listed and discussed on Chapter 2. Based on the findings the developments stages has been designed which is followed till the completion of the research. By using dynamic programming approach to solve the constituent bi-cliques in a k -partite graph, the proposed algorithm solves for the k -clique of said graph via sectioning it into sub-problems and using memoization of those results to refer back to provide an efficient, stochastic solution to the maximal k -clique problem. The key concept of developing the algorithm is considering the bi-clique from the smallest of bi-partite formed and picking the intersect vertices of the stored bi-cliques which can solve for k clique result from a k partite graph. The conceptual development that has been discussed overall on the previous chapters has common

features only in the biclique solving section but it does not only handle bi-cliques the aim is to find significant induced sub-graphs.

6.2 Future Works

An efficient solution for k cliques can create many real world applications opportunities. Clique solving algorithms has been used and many research has been done using real world data sets as well as many experimental result has been generated using artificial data set. From this study, the new concept which is developed for k clique solving will create an huge impact on clique studying sector. As the new concept of adding dynamic programming is a conceptual development idea and will have probable efficient implementation with real world data set. Considering the previous findings of the significance applications the reduced time complexity will make real world implementations easier, optimal and efficient. There are different patterns of graph theory and its features which is significant and has huge applications. The clique pattern and its usage can help to gain result from different dense network to make different crucial decision and find parameters and nodes which are significant to the given data-set.

References

- Alexe, G., Alexe, S., Crama, Y., Foldes, S., Hammer, P. L., & Simeone, B. (2004a). Consensus algorithms for the generation of all maximal bicliques. *Discrete Applied Mathematics*, 145(1), 11–21.
- Alexe, G., Alexe, S., Crama, Y., Foldes, S., Hammer, P. L., & Simeone, B. (2004b). Consensus algorithms for the generation of all maximal bicliques. *Discrete Applied Mathematics*, 145(1), 11–21.
- Chiba, N., & Nishizeki, T. (1985). Arboricity and subgraph listing algorithms. *SIAM Journal on computing*, 14(1), 210–223.
- Corno, F., Prinetto, P., & Reorda, M. S. (1995). Using symbolic techniques to find the maximum clique in very large sparse graphs. In *Proceedings the european design and test conference. ed&tc 1995* (pp. 320–324).
- Eblen, J. D., Phillips, C. A., Rogers, G. L., & Langston, M. A. (2011). The maximum clique enumeration problem: Algorithms, applications and implementations. In *Bioinformatics research and applications: 7th international symposium, isbra 2011, changsha, china, may 27-29, 2011. proceedings 7* (pp. 306–319).
- Eisenbrand, F., & Grandoni, F. (2004). On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1-3), 57–67.
- Feige, U., Seltser, M., et al. (1997). *On the densest k-subgraph problem*. Citeseer.
- Grünert, T., Irnich, S., Zimmermann, H.-J., Schneider, M., & Wulfhorst, B. (2002). Finding all k-cliques in k-partite graphs, an application in textile engineering. *Computers & operations research*, 29(1), 13–31.
- Krokhmal, P. A., Grundel, D. A., & Pardalos, P. M. (2007). Asymptotic behavior of the expected optimal value of the multidimensional assignment problem. *Mathematical programming*, 109(2-3), 525–551.
- Li, R., Gao, S., Qin, L., Wang, G., Yang, W., & Yu, J. X. (2020). Ordering heuristics for k-clique listing. *Proc. VLDB Endow.*

- Liu, Q., Chen, Y.-P. P., & Li, J. (2009). High functional coherence in k-partite protein cliques of protein interaction networks. In *2009 ieee international conference on bioinformatics and biomedicine* (pp. 111–117).
- Makino, K., & Uno, T. (2004). New algorithms for enumerating all maximal cliques. In *Algorithm theory-swat 2004: 9th scandinavian workshop on algorithm theory, humlebæk, denmark, july 8-10, 2004. proceedings 9* (pp. 260–272).
- Mirghorbani, M., & Krokmal, P. (2013). On finding k-cliques in k-partite graphs. *Optimization Letters*, 7, 1155–1165.
- Mushlin, R. A., Kershenbaum, A., Gallagher, S. T., & Rebbeck, T. R. (2007). A graph-theoretical approach for pattern discovery in epidemiological research. *IBM systems journal*, 46(1), 135–149.
- Ruparelia, N. B. (2010). Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, 35(3), 8–13.
- Sanderson, M. J., Driskell, A. C., Ree, R. H., Eulenstein, O., & Langley, S. (2003). Obtaining maximal concatenated phylogenetic data sets from large sequence databases. *Molecular biology and evolution*, 20(7), 1036–1042.
- Shaham, E., Yu, H., & Li, X.-L. (2016). On finding the maximum edge biclique in a bipartite graph: a subspace clustering approach. In *Proceedings of the 2016 siam international conference on data mining* (pp. 315–323).
- Tomita, E., Tanaka, A., & Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical computer science*, 363(1), 28–42.
- Vassilevska, V. (2009). Efficient algorithms for clique problems. *Information Processing Letters*, 109(4), 254–257.
- Wu, Q., & Hao, J.-K. (2015). A review on algorithms for maximum clique problems. *European Journal of Operational Research*, 242(3), 693–709.
- Zaki, M. J., & Hsiao, C.-J. (2002). Charm: An efficient algorithm for closed itemset mining. In *Proceedings of the 2002 siam international conference on data mining* (pp. 457–473).
- Zhang, Y., Chesler, E. J., & Langston, M. A. (2008). On finding bicliques in bipartite graphs: a novel algorithm with application to the integration of diverse biological data types. In *Proceedings of the 41st annual hawaii international conference on system sciences (hicss 2008)* (pp. 473–473).