DEVELOPMENT OF NOVEL ALGORITHMS FOR ANALYSIS AND VISUALIZATION OF LARGE GRAPH

A THESIS SUBMITTED TO

SAVITRIBAI PHULE PUNE UNIVERSITY

FOR AWARD OF DEGREE OF

DOCTOR OF PHILOSOPHY (PH.D)

IN THE FACULTY OF

COMPUTER ENGINEERING

SUBMITTED BY

MS. SWATI KRISHNA BHAVSAR

UNDER THE GUIDANCE OF

Dr. VARSHA HEMANT PATIL

RESEARCH CENTRE

DEPARTMENT OF COMPUTER ENGINEERING

MATOSHRI COLLEGE OF ENGINEERING AND RESEARCH CENTRE

EKLAHARE, NASHIK

SEPTEMBER 2018

MATOSHRI COLLEGE OF ENGINEERING AND RESEARCH CENTRE, EKLAHARE, NASHIK

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the work incorporated in the thesis, **"Development of Novel Algorithms for Analysis and Visualization of Large Graph"** is submitted by **Ms. Swati Krishna Bhavsar** for the **Doctor of Philosophy (Ph.D)** in **Computer Engineering, Savitribai Phule Pune University,** has been carried out by the candidate at **Department of Computer Engineering, Matoshri College of Engineering and Research Centre Eklahare, Nashik** during the period from **July 2014 to September 2018** under the guidance of **Prof.(Dr.) Varsha Hemant Patil.**

Prof.(Dr.) Varsha Hemant Patil
Research Guide,
Vice Principal and Head,
Computer Engineering,
MCOERC, Eklahare, Nashik

Prof.(Dr.) Gajanan K. Kharate
Principal,
MCOERC, Eklahare, Nashik

Certificate of the Guide

Certified that the work incorporated in the thesis **"Development of Novel Algorithms for Analysis and Visualization of Large Graph"** submitted by **Ms. Swati Krishna Bhavsar** was carried out by the candidate for the **Doctor of Philosophy (Ph.D)** degree at **Department of Computer Engineering, Matoshri College of Engineering and Research Centre, Eklahare, Nashik** during the period from **July 2014 to September 2018** under my direct supervision and guidance.

Place:

Date:

Prof.(Dr.)Varsha Hemant Patil
Research Guide,
Vice Principal and Head,
Computer Engineering,
MCOERC, Eklahare, Nashik

Declaration by the Candidate

I Hereby declare that the thesis entitled **"Development of Novel Algorithms for Analysis and Visualization of Large Graph"** submitted by me to the **Savitribai Phule Pune University, Pune** for the degree of **Doctor of Philosophy(PhD)** in **Computer Engineering,** is the record of work carried out by me during the period from **July 2014 to September 2018** under the guidance of **Prof. (Dr.) Varsha Hemant Patil** and has not formed the basis for the award of any degree, diploma, associateship, fellowship, titles in this or any other University or other Institution of Higher learning. I further declare that the material obtained from other sources has been duly acknowledged in the thesis.

Place:

Date:

Ms. Swati Krishna Bhavsar
Research Scholar
Department of Computer Engineering,
MCOERC, Eklahare, Nashik

Abstract

Large Graphs are widely used to represent information in various scientific fields and businesses like Very Large Scale Integrations (VLSI) designs, social networking, medical applications, publication records and similar. Analysis and visualization of a large graph containing thousands of vertices and edges is a very challenging task. Solution to this problem is to partition a large graph into sub-graphs and then analyse and visualize each sub-graph independently. Besides traditional goals of partitioning, an efficient novel algorithm must partition a given graph in minimum amount of time, with minimum edge-cuts and without any loss of data and visualize sub-graphs efficiently independent of application.

In this thesis, we present an efficient graph partitioning algorithm and visualization tool as outcomes of our research work. Firstly, a Novel Cut-set and Partitioning Algorithm (CPA) has been developed. The algorithm follows a principle of finding most connected components in a network and vertex cluster-based approach for graph partitioning. Secondly, for the better understanding of generated sub-graphs, an efficient visualization tool is developed. Thirdly, an algorithm for an author information retrieval is developed that uses CPA algorithm and visualization tool for Digital Bibliographic and Library Project (DBLP) dataset to retrieve specific author publication details and correlation among authors. Further performance of an author is measured by introducing novel parameters like consistency and contribution factor in addition to stability, cooperativeness and solidity.

These algorithms are successfully implemented and tested; it is observed that it effectively works for graph of any size for partitioning and visualization. Using a vertex clustering approach with most connected component, research work signifies that developed algorithms perform better partitioning for any large graph with minimum number of edge-cuts. Further it supports to visualize all sub-graphs obtained after partitioning more

effectively. Algorithms demonstrate that for DBLP dataset, it works efficiently for retrieving specific author publication records. Research work demonstrates that there is significant improvement in performance as compared to existing techniques used for partitioning and visualization of large graph.

Keywords: DBLP, large graph analysis, visualization, author publication record, solidity, consistency.

Acknowledgement

I am grateful to Lord Ganesh for his special blessings that he has always been showering on me.

It is an honor for me to thank those who made this research work possible. My first debt of gratitude goes to my guide, worthy and esteemed mentor, Prof. (Dr.) Varsha Hemant Patil for the valuable guidance, constructive suggestions, critical evaluation and consistent encouragement in completing my thesis. Her advice, guidance, ideas, insights and encouragement is instrumental in making this research work possible.

I render my deep sense of gratitude to respected Prof. (Dr.) Gajanan K. Kharate, Principal, MCOERC without whom my dream would never come to reality. His constant encouragement and constructive suggestions helped me a lot to make my dream come true. Special thanks to the committee members Prof. (Dr.) Shirish S. Sane and Dr. R. S. Tiwari for their valuable suggestions, which were instrumental in improving the quality of the research.

I am also thankful to Prof. N. L. Bhale,, Dr. R.G. Tathed, Dr. J. J. Chopade and other staff members of MCOERC, Nashik for their suggestions and support during this research journey.

I am grateful to my parents, Late Krishna B. Bhavsar and Smt. Rajani Bhavsar, parents in law, Mr. Prabhakar Wakde, Sou. Rukmini P. Wakde for their moral support and blessings. My heartful thanks go to my husband, Avinash for his patience and understanding and my son Mast. Gaurang who had missed many affectionate hours during this work.

Ms. Swati Krishna Bhavsar

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Partitioning and visualization of large graph are very useful tasks for analyzing complex data sets like social networking sites, experimental devices and sensor networks. For the research work undertaken, the topics covered in this chapter include introduction to large graph, motivation, objectives and research contributions.

1.1 Large Graph

A graph is a set of vertices and edges. It is a general data structure that models peculiar relationship among objects, where vertices are objects and relationships as edges. A large graph is one which consists of hundreds to thousands of nodes and millions of edges [1]. Large graphs are popularly called colossal graphs. Some common examples of large graph include Very Large Scale Integrations (VLSI) designs, telephone networks, social networking sites, flight systems,medical applications, publication records among many more. For VLSI designs, units on a chip are represented as vertices and wires connecting them as edges. In the case of social networking sites, vertices represent web pages and edges represent links among them. Similarly, for publication records, authors symbolize vertices and co-authors symbolize edges. DBLP (Digital Bibliography and Library Project) is a dataset of authors and co-authors relationships [45, 111]. Figure 1.1 shows an example of general graph and figure 1.2 shows example of DBLP graph[120].

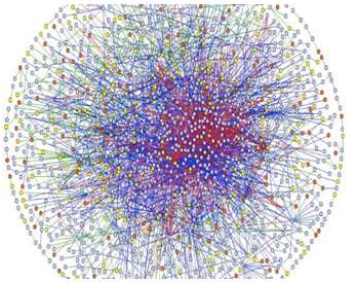

Figure 1.1: General Large Graph Example

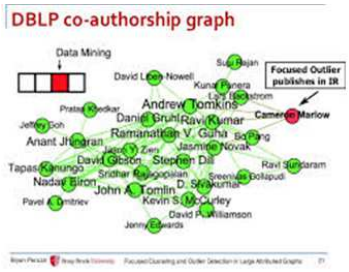

Figure 1.2: DBLP Large Graph Example

Large graphs are sizably voluminous graphs and those are commonly represented by means of popular data structures - adjacency matrix and adjacency lists. Memory Requirement for storage of matrix is too high as compared to adjacency list. Dynamic updates in nodes and edges are difficult operations especially in case of the adjacency matrix.

A large graph is an complex data structure and it requires excessive processing, higher memory for storage and knowledge of patterns of the graph [2]. To find patterns in a large graph, it is desirable to analyze, visualize, summarize and mine it [2]. Some of the graphs that change with time are known as dynamic graphs. So it is very difficult to comment on the exact size and pattern of such large and dynamic graphs. These large graphs cannot be analyzed and visualized as a whole. There is a need to partition the large graph into smaller subgraphs for effective analysis. Hence powerful graph partitioning along with graph visualization are to be used while processing and analyzing large graphs efficiently.

Large graphs are viewed using graph visualization techniques. Cutting the graph into smaller parts is one of the algorithmic operations, which is known as graph partitioning.

1.1.1 Graph Partitioning

If $G = (V, E)$ is a graph, then graph partitioning is a process to divide vertex set V into k parts (subsets) $\{v_1, v_2, ..., v_k\}$ such that the subsets are disjoint and are of equal size, partitioning is balanced and the number of edges with endpoints in different subset is minimized.

For analysis of graph, at a given time, it is very important to partition the sizably voluminous graph into smaller subgraphs so that graph will fit into relatively less memory without any information loss in terms of connectivity [3].

For partitioning a minimum number of edge cuts are desirable. Any edge that is cut in

order to partition a graph into smaller graphs can be called as cut-edge. Therefore, finding the threshold value of an edge cut plays a paramount role in the processing and analysis of large graphs [4].

1.1.2 Graph Visualization

The level of graph hierarchy by which graph can be seen is known as graph visualization. Graph visualization is a representation of interconnected nodes arranged in space and its representation in a particular structure so that user can understand it easily.

As the size of the graph is too large, it becomes one of the major challenges in efficient graph visualization. Large graph can cause various difficult problem named as [8, 10] algorithmic complexity, display clutter, readability and navigation.

i. Algorithmic Complexity: Graph size is vital to algorithms in some cases because a lot of useful graph algorithms are either NP-complete or NP-hard. Therefore, some layout algorithms can be totally unusable when facing graphs of a large size. Algorithms require long processing time and make it hard to interact in real-time.

ii. Display Clutter: Large Graph contains thousands of vertices or more. When the size of the data grows, graph becomes visually cluttered and confusing. It becomes difficult for user to discriminate between nodes and edges.

iii. Readability: Human perception able to visualize only small for graphs including firstly finding the node and secondly finding links between nodes.

iv. Navigation: Navigating large information spaces, such as graphs with thousands of nodes suffer from the problem of viewing a large space on a small display.

The large graph analysis is performed with the help of an efficient partitioning and efficient visualization. The large graph problem can be treated through graph hierarchies, according to which a graph is recursively broken to define a tree of sets of partitions. For understanding a graph hierarchy, some of the important terms used are as below-

- Hierarchical Navigation: It is the relation between arbitrary groups (partitions) of nodes.

- Representation and Processing: Adjacencies of given graph nodes are computed in consideration with an entire graph, instead of a particular partition [1].
- Mining: From a given subset of nodes in the graph, particular induced sub-graph that best summarizes the relationships of these subsets are to be identified efficiently.
- Visualization: The levels of the graph hierarchy that are can be seen [7] [8].
- Interaction: It shows that how the above-mentioned tasks is efficiently and intuitively performed.

1.2 Motivation

As a promising technology, large graph analysis received a lot of attention in recent years. The concept of large graph analysis is simple to understand and has two important parameters viz graph partitioning and graph visualization. It has a tremendous amount of applications in web graphs, social networks, recommendation systems, VLSI designing and many similar ones [19]. Literature reveals that noticeable graph partitioning and visualization algorithms were designed and developed by various researchers, but the minimum numbers of edge-cuts are not ensured while partitioning moreover existing graph visualization systems are application specific. Hence it is a need of the day to develop an efficient system for graph partitioning and graph visualization that will ensure the minimum number of edge-cuts while partitioning and will provide an efficient visualization of sub-graphs.

This motivates to carry out a theoretical analysis with mathematical properties contributing to the development of novel partitioning algorithm.

1.3 Research Statement and Objectives

Statement for proposed research work is, To develop Novel Algorithms for Analysis and Visualization of Large graphs. The objectives of proposed research work are-

1. To undergo thorough review of different partitioning and analysis techniques proposed and implemented by various researchers.

2. To study various visualization and summarization techniques for large graph and identify the potential research gaps.
3. To develop the algorithms for analysis and visualization for large graph.
4. To implement and test the proposed algorithms using available standard data set.
5. To compare the performance of proposed algorithms with techniques proposed by researchers and validate the results.

1.4 Research Contributions

This research work focuses at reducing the number of edge cuts while partitioning the large graph into subgraphs using a novel algorithm for analysis and visualization of graph. The outcomes of research include:

- The novel algorithm **"Cut-set and Partitioning Algorithm (CPA)"** for graph partitioning has been successfully designed and implemented.

  o The numbers of edge cuts obtained by the proposed algorithm are significantly less as compared to existing algorithms

  o Number of subgraphs obtained as a result of partitioning requires relative less memory for storage.

- **"An Effective Visualization tool"** is developed for visualization of partitioned sub-graphs.

  o Visualization tool is capable of efficient dynamic visualization of 128 sub-graphs obtained after partitioning a large graph.

  o Visualization tool efficiently visualizes author and co-authors relationship for DBLP publication records. This visualization tool will be useful for visualizing many complex data sets.

- Design and implemented, An Author Publication Information Retrieval Algorithm for retrieving authors publication details from DBLP dataset.

  o The system works efficiently for DBLP dataset for retrieving authors publication information along with its all co-authors and their association.

o When selected one of the co-authors to visualize association, the co-author node becomes main author node and can be browse through and continue.

o Authors performance is measured by designing several parameters like consistency and contribution factor and retrieval and visualization of n most influential authors from a specified span of years along with parameters proposed by Forcoa. Net like stability, cooperativeness and solidity.

1.5 Organization of Thesis

Graph Analysis and Visualization system is designed, implemented and tested for standard graph dataset. The rest of the thesis is organized as follows. Chapter 2 is a review of previous related work, detailed process of graph mining including graph classification and clustering. It also contains survey of graph partitioning algorithms and graph visualizations tools. Chapter 3 provides details of our framework for development of partitioning algorithm and visualization tool. It also discusses the design of DBLP processing and visualization system for visualization of a large graph. Chapter 4 includes an evaluation of graph partitioning algorithms for computation of number of edge-cuts and subgraph visualization. Chapter 5 provides a summary of the salient points of the thesis, and directions for further research.

Chapter 2

Literature Survey

A significant development in graph analysis and visualization has been seen in the last two decades, and an over abundance of literature in the form of journals, transaction papers, patents, reviews, and surveys is available. To provide an overall idea of the domain, the main stages in graph partitioning and visualization along with relevant literature are described in subsequent sections.

2.1 Preliminaries of Graph

- Graph: A graph *G(V, E)* where V is the set of vertices, E is the set of edges and the number of vertices $n = |\mathrm{V}|$.
- Cyclic Graph: If a graph contains a cycle means a simple path that begins and ends with the same vertex is known as a cyclic graph. If a graph that contains no cycle then it is known as acyclic. An acyclic graph may be called as a tree and a set of tree forms a forest.
- Sparse Graph / Dense Graph: A Sparse graph is a graph in which the numbers of edges are close to the minimal number of edges while the dense graph is a graph in which a number of edges are close to the maximal number of edges.
- Directed and Undirected Graph: For given graph $G(V, E)$, the graph is undirected if $(v, w)\epsilon E \Leftrightarrow (w, v)\epsilon E$ graph as shown in figure 2.1, otherwise the graph is un directed

if $(v, w)\epsilon E \neq (w, v)\epsilon E$. Directed graph is as shown in figure 2.1 and undirected graph is as shown in figure 2.2.

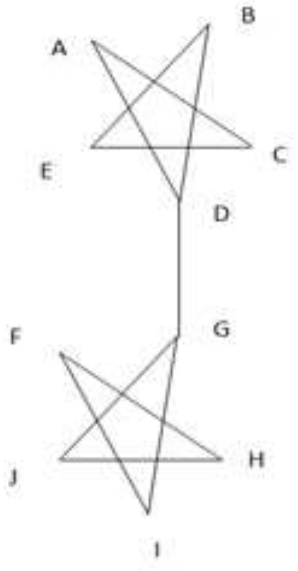

Figure 2.1: Undirected Graph

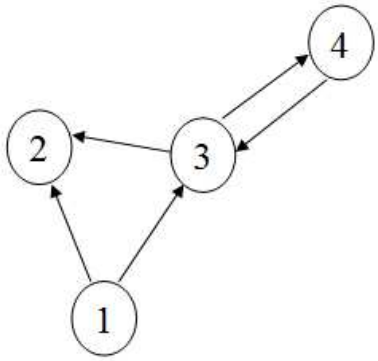

Figure 2.2: Directed Graph

- Static / Dynamic: Static graph consists of a fixed number of nodes and edges. In a dynamic graph, insertion and deletion of edges and vertices are performed at any time.

- Hyper graph: A hyper graph is a generalization of a graph in which an edge can join any number of vertices. Formally, a hyper graph is a pair where a set of elements is called nodes or vertices and is a set of non-empty subsets called hyper-edges or edges.An example of hypergraph is shown in figure 2.3[65].

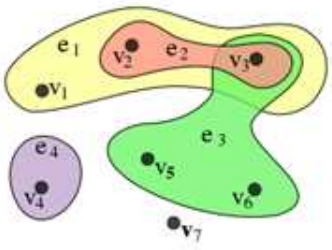

Figure 2.3: Hyper Graph G = (V, E)

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$$
$$E = \{e_1, e_2, e_3, e_4\} = \{\{v_1, v_2, v_3\}, \{v_2, v_3\}, \{v_3, v_5, v_6\}, \{v_4\}\}$$

- Adjacent Vertices and Edges, Incident Edges: Let *G = (V, E)* be an undirected graph. If *e (v, w)* $\in E$, then v and w are said to be adjacent vertices of e and an edge e is said to be an edge incident on v, w. Two edges e_1 and e_2 are said to be incident if they are incident on same vertex.

- Simple and multi graph: A graph *G = (V, E)* is simple if it does not have loops and/or more than one edge incident on the same vertex. Otherwise if more than one edge incident on the same vertex then the graph is known as a multi-graph.

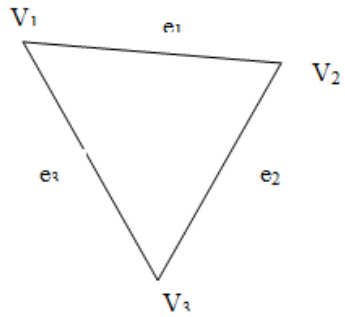

Figure 2.4: Simple Graph

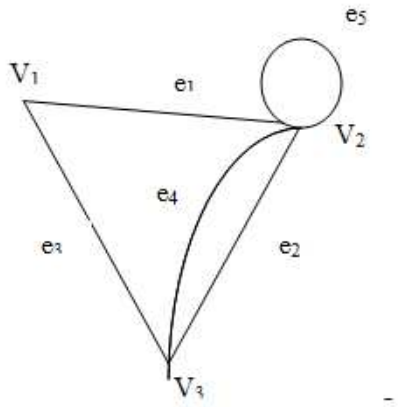

Figure 2.5: MultiGraph

Figure 2.4 shows a simple graph whereas figure 2.5 shows multi-graph with edges e_2, and e_4 are incident on vertices v_2 and v_3.

- Degree of vertex: Let $G = (V, E)$ be undirected graph and $v \in V$, degree of v is the number of edges incident to the vertex, with loops counted twice.

- Loops and Parallel Edges: An edge is a loop if both ends of an edge are incident on the same vertex. If the same pair of vertices is adjacent by two or more edges, then those edges are called as parallel edges.

- Weighted Graph: It is a graph in which each edge is given a positive value as a numerical weight.

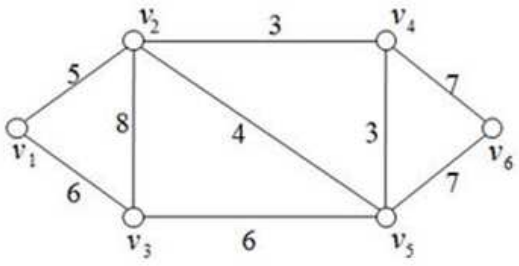

Figure 2.6: Weighted Graph

For all weighted graph, a degree of vertex v is an addition of weights of all edges incident on v. Figure 2.6[68] shows an example of a weighted graph.

- Degree Matrix: Degree matrix is a diagonal matrix which contains information about the degree of each vertex.

$$d_{i,j} = \begin{cases} deg(v_i) & \text{if} i = j \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

- Adjacency Matrix: Let $G = (V, E)$ be simple graph on v vertices then adjacency matrix A is of order vxv $\forall$ (i, j) $\in$ (1,2,....v)^2

$$A_{ij} = \begin{cases} 0 & \text{if} i = j \\ w(i,j) & \text{otherwise} \end{cases} \tag{2.2}$$

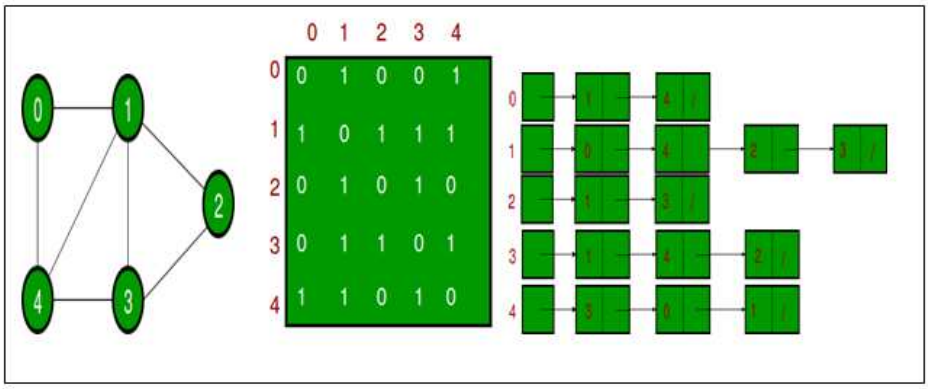

Figure 2.7: Graph, Adjacency Matrix of Graph and Adjacency List of Graph

Figure 2.7 shows an input graph, its adjacency matrix representation and adjacency list respectively[68].

- Laplacian Matrix: Let $G = (V, E)$ be a graph, then the matrix $L = D - A$ is called as Laplacian matrix, where D is degree matrix and A is the adjacency matrix.

- Path and Circuit in Graph: Let $G = (V, E)$ be a graph and v_1, v_2 be nodes in a graph. If there is a sequence of edges from v_1 to v_2 then there is a path from v_1 to v_2. The path is called as circuit if the start and end vertex is same.

- Connected Graph: Let $G = (V, E)$ be a connected graph if there is path exist between every pair of vertices.

- Matching of Graph: Let $G = (V, E)$ be a graph. Set containing pairwise non-adjacent edges are called a matching of graph. Maximal matching includes the highest possible number of edges.

- Cut: A cut performs partitioning of graph into two subsets A and B. The cut-set of a cut is the set of edges that have one endpoint in A and the other endpoint in B. If v_1 and v_2 are specified vertices of the graph G, then v_1-v_2 is a cut in which v_1 belongs to the set A and v_2 belongs to the set B.

- Hypergraph Partitioning: It is the process of partitioning of hypergraph in such a way that the net cut, a cost function is minimized. Even it is expected that solution has to satisfy balanced constrained and then a process is called balanced hypergraph partitioning.

2.2 Graph Mining

Since last decade, various researchers are working on data mining for enhancing the performance of information retrieval. If information is stored in the form of structured data, where a structure of data is represented by means of graph and has the relationship among the data then it is called as graph mining. Graph mining is used to discover the information from graph data [1]. Graph mining is a process of obtaining desired subgraphs from the graph. It has many applications such as Social Network Analysis, Designing Computer Networks, Chemical Reactions, Bio-Informatics, Program Flow Structures, Image Processing, Enterprise data, Chemical Reactions and Sparse Matrix ordering [2].
The graph mining techniques are categorized into Graph clustering, Graph classification, and Subgraph mining.

- Graph Clustering: Graph vertices are grouped together to form clusters based on the edge structure of the graph is known as graph clustering. The care has been taken while forming the structure such that the inter-cluster edges may be more but intra-cluster edges should be less [12]. Graph clusters are formed based on some similarities in graph structures.

- Graph Classification: The graph is classified into separate graphs using supervised/semi-supervised learning technique where classes of the data are unknown in prior known as graph classification [2, 13].

- Sub-graph Mining: The frequent sub-graph mining is used to produce the set of sub-graphs depending on a threshold value[16]. Sub-graph is a graph whose vertices and edges are subsets of another graph.

2.3 Graph Clustering

In graph clustering process, the graph is divided into clusters by considering important property as cluster fitness measure. Cluster fitness measure is used to decide the quality of the cluster with two measuring parameters such as density measures and cut based methods.

2.3.1 Density Measures

In this technique, the density of a graph is computed based on the threshold value and based on the threshold value, maximal sub-graphs are searched [14].

2.3.2 Cut Based Method

In addition to direct density measures, connectivity measures with the rest of the graph are used to identify high-quality clusters. Measures of the independence' of a sub graph of the rest of the graph have been defined based on cut sizes. Possibly the most important such measure in the context of clustering is conductance which leads to the graph partitioning [15]. The problem is to partition the vertices of a graph into equal parts such that the number of cuts should be minimized. The graph partitioning problem is NP-Complete.

2.4 Graph Classification

In general, graph classification is defined as task of assigning a discrete class label set Y to input instances in an input space X. suppose molecular structure is considered, then X = {valid molecular structures} and Y = {toxic, non-toxic}. The graph structure as shown in figure 2.8 considered as toxic, then each full graph is assigned a class label and based on the label, classification is performed as toxic or non-toxic[28].

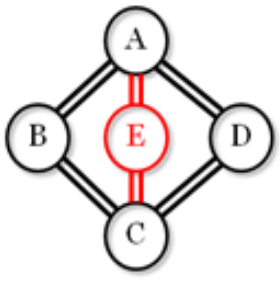

Figure 2.8: Molecular Graph With Label As Toxic

The Classification approaches of the graph are classified into relational approaches, kernel method, random walks and Local Approaches [22].

2.4.1 Relational Approaches

Almost all real-world data are stored by using the relational database; therefore, the relational approach is used to classify the objects in one relation. Multi-relational classification classifies an object using multiple relations. Multi-relational classification is used to discover useful patterns across multiple relations who are stored in tables [104]. As a multi-relational approach is based on the traditional machine learning approach, it collects the random sample of homogeneous data from the single relation. As the real world data is based on multi-relational and heterogeneous data. This solution cannot be realistically applied for it [112].

2.4.2 Kernel Method

Kernel method is useful for graph classification. It is based on machine learning and data mining communities. Support Vector Machine(SVM) supports kernel method and uses kernel function and eigenvalues. Direct product kernel matrix is computed for all pairs of graphs. This matrix is used as input to SVM function to create the classification model.
For creating kernel matrix, several assumptions are used as- K is positive definite if and only if K has only non-negative eigenvalues and K is strictly positive definite if and only if K has only positive eigenvalues, that is no zero eigen-values[20].

2.4.3 Random Walk

Random Walk is based on the Markov Chain model. D^y is a random walk starting in a labelled node and ending when any node having the same labels which is reached for the first time. D-walk betweenness $B(q, y)$ is the expected passage time on an unlabeled node for each class[16]. The D walks have an ability to classify directed and undirected graphs. Its Complexity is linear with respect to -

1. Number of edges in the graph
2. Maximum walk length
3. Number of classes

This technique has been tested on the CORA database. The experimentation shows that it classifies the unlabeled nodes of the graphs and performs better than the techniques available in the literature such as [17] and [18]. Their main attainment is towards handling the graphs of large a number of nodes and edges.

2.4.4 Kernel Based Approach

Another approach for graph classification is a kernel-based approach which can handle extremely large no of nodes and edges [20]. It is based on the concept of the inner product of two graphs where nodes of graphs and labels of edges are considered for inner products. If two graphs similarities are identical, then they are classified into the same group. It has been implemented for the prediction of properties of the chemical compound using the mutag and PTC dataset.

2.5 Sub-graph Mining Approach

Frequent sub-graphs are beneficial for graph classification, clustering, and characterization of graph sets. As the sub-graph size decreases, the graph pattern size increases exponentially. It may lead to issue of identification of sub-graphs may take more time. This issue can be solved by using scalable algorithms recommended which generate frequent sub-graphs in search space [46-54]. The approaches used for sub-graph discovery are candidate generation and a- priori based algorithms.

2.5.1 Candidate Generation

It is one of the important phase in frequent sub-graph mining in which candidate sub-graphs are systematically generated. Some of the strategies which identify candidate sub-graphs are listed below:

1. Level- wise Join: Two sub-graphs of size m are combined together to form a $(m+1)$ candidate sub-graph.

2. Rightmost path extension: In this candidate generation strategy, vertices are added on the rightmost path of m-sub tree to form $(m+1)$ sub tree. And other strategies such

as extension and join, right-and-left tree join and equivalence class based extension are also used in candidate generation phase.

2.5.2 A-Priori Based Algorithms

A-Priori based approach is similar to frequent item-set mining and it works recursively [49]. Some of the a-priori based frequent sub-graph mining algorithms are listed below.

- AGM[48]: This algorithm generates candidate graphs, merges any two candidate graphs. It checks whether a resultant graph is a sub-graph in a given graph. Two graphs of size m are merged together to form a resultant graph of size $m+1$. One of the more resultant graphs of size $m+1$ again fed to a-priori algorithm to obtain as an $m+2$. As two graphs are merged together, graph size is increased by one vertex, hence it is called as vertex based candidate generation algorithm.

- FSG[49]: This method is based on the edge based candidate generation process. The number of edges in a graph represents the size of a graph. Similar to vertex based candidate generation method, two graphs of size m are merged together to form the resultant graph of size $m+1$ which must be frequent. In further iterations, it generates candidate sub-graph whose size is exactly greater than 1 of previous ones.

- Edge disjoint path-join algorithm[50]: It measures the number of disjoint paths of a graph. A path is said to be disjoint if it does not share a common edge. Two candidate graphs of m disjoint paths are merged together to form a resultant graph containing $m+1$ disjoint paths.

2.6 Graph Partitioning Problem

If $G = (V, E)$ is unwieghted undirected graph,then Graph Partitioning Problem(GPP) is to divide vertex set V into k parts (subsets) $\{v_1, v_2, ..., v_k\}$ such that the subsets are disjoint and have equal size, partition is balanced and the number of edges with endpoints in different subsets is minimized. Partitioning graph into distinct sub graphs of approximately same size with the objective of minimizing cut value is NP complete problem.

Basic graph partitioning algorithms are based on bi-partitioning of graph and furthermore generalized to kpartitioning. figure 2.9 shows bipartitioning of a graph and figure 2.10 shows k partitioning[68].

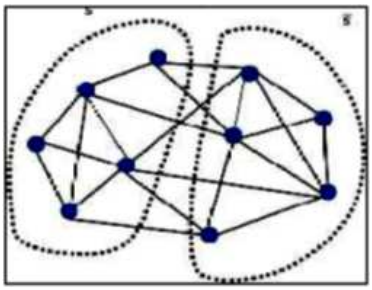

Figure 2.9: Bi Partitioning

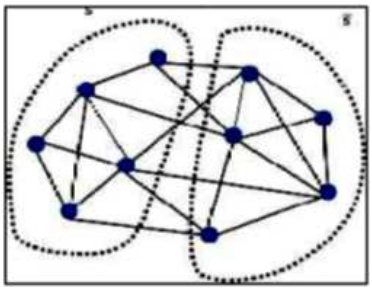

Figure 2.10: K Partitioning

2.7 Thrust of Large Graph Analysis

From the preliminaries of graph discussed earlier, for large graph, graph analysis is performed by applying graph partitioning means by dividing the large graph into subgraphs to fit into limited memory and to apply parallelism for the assignment of multiple processors. There is several thrust areas graph analysis. Some of them are discussed as-

2.7.1 VLSI Circuit Design

Very Large Scale Integration consists of the creation of integrated circuits. It is composed of thousands of transistor on a single chip. These are used to perform one or many operations at a time. The circuit design is transformed into the physical circuit. These interactions are in terms of physical wires. It increases circuit size, cost and power usage. Partitioning is used in the circuit design to handle the issue of complexity of designing VLSI, to divide the system into smaller and manageable components. It plays a major role towards improving system performance. In electronics, propagation delay is the time between input and output state becomes stale and valid. Wires have a delay of 1 ns for 15cm of length and logic gates are in picoseconds. In a circuit, the wire connects more than two components; it needs representation of graph using hyper-graphs where every wire is represented by a hyper edge.

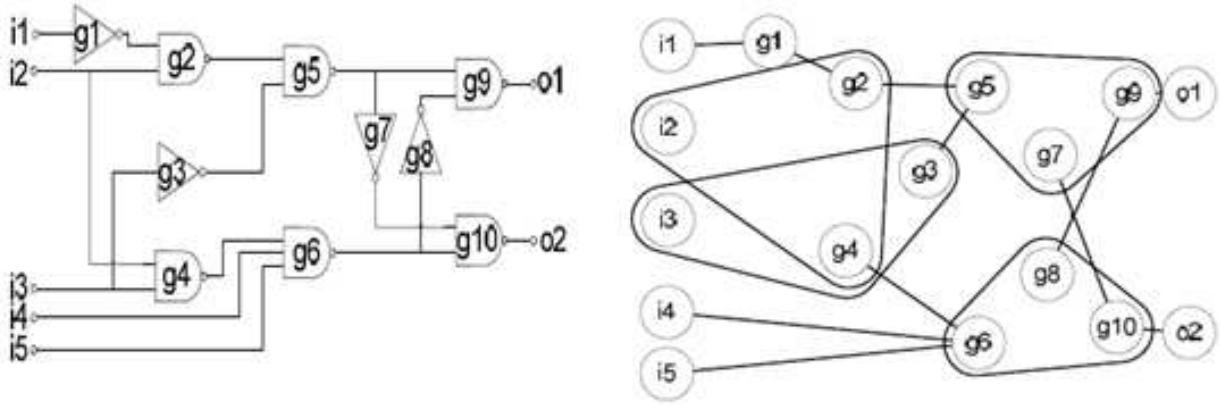

Figure 2.11: An Example of Logic Circuit and Its Corresponding Hyper Graph

Figure 2.11 shows logic circuit and its corresponding hyper-graph. The Logic circuit is composed of gates called as standard cells which are connected through metal wires. When electrical signal propagate through it, such a connection is called a net. To construct a hyper-graph from it, gates are considered as vertices and nets as hyper edges. This graph is represented by a matrix [58]. Hyper-graph is transformed into the pattern of non zero entries of matrix A Vertices corresponds to the column of A and hyper edges corresponds to rows of A. Each hyper edge e will be connected to A vertex v, if $A_{e,v} \neq 0$ as shown in figure 2.11

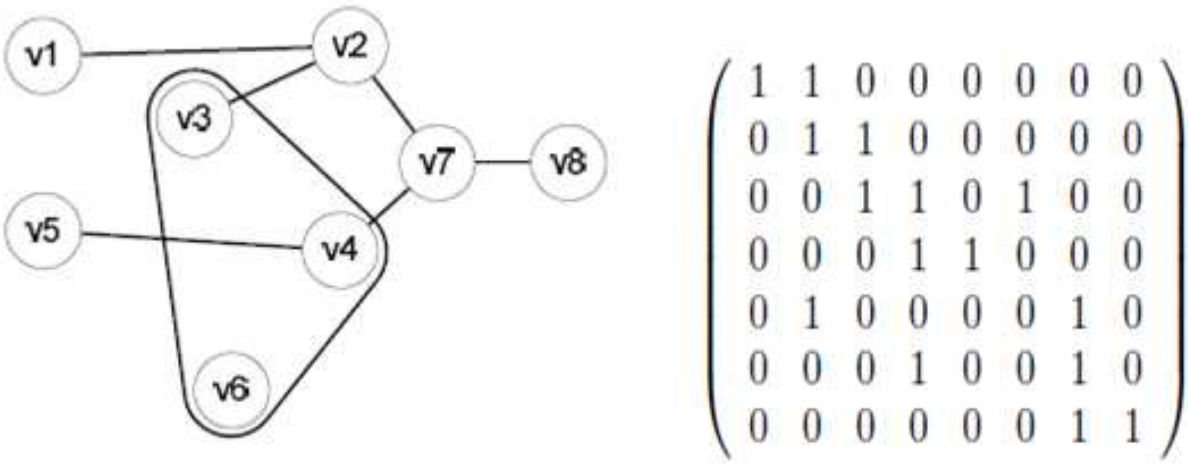

Figure 2.12: An Example of Hyper-Graph and Its Respective Matrix Representation

Figure 2.12 shows an example of hyper-graph and its corresponding matrix representation[58]. There is a need for partitioning of such a hyper-graph with the minimum number

of edge cut value to parallel assign the task of the circuit design with minimum loss of information.

2.7.2 Authorship Network Analysis Significance of Authorship Network in DBLP data

DBLP dataset keeps track of approximately 9, 13, 534 authors and their IEEE, ACM journals and conference publication in computer Engineering. Using the information about authors and their papers cooperation with co-authors can be represented. This maintains the relationship between authors and co-authors in association with various parameters as article key, mdate, co-author, title, pages, year, volume, Journal, and number. All this information needs to visualize by using the visualization tool. As it is a huge data, it is time and memory consuming task to load it at the time of visualization. Therefore need to preprocess and partition the data for retrieving particular authors information and to measure the performance of author [59].

2.8 Existing Algorithms for Graph Partitioning

Graph partitioning algorithms are mainly classified into constructive algorithms and refinement algorithms.

2.8.1 Constructive Algorithm

A constructive algorithm is a heuristic type method [28]. It starts with an empty solution and repeatedly extends the current solution until a complete solution is obtained. Examples of some constructive heuristics developed for famous problems are: flow shop scheduling and vehicle routing problem. Different types of constructive algorithms are an exact algorithm, random partitions, spectral bisection methods, greedy construction, and graph growing partition algorithm.

2.8.1.1 Exact Algorithm

GPP problem can be solved by applying branch and bound techniques. Bounds are determined by using semi definite programming [21], multi commodity flow [23] and linear programming [24]

Hanger et al. applied branch and bound technique to Graph Partitioning Problem in the continuous quadratic form [25]. Furthermore, Delling et. al [26] had derived bounds by computing minimum $s-t$ cut between vertices(v_1, v_2) such that $v1 \cap v2 = \emptyset$. This method is used for partitioning of complex road network having millions of nodes. But the limitation of this method its running time is more. Running time is dependent on the bisection width of graph.

All these methods are used in solving small size graph partitioning problem and having large running time. Modified Lanczos algorithm can be used to compute eigen-vector with respect to second smallest eigen-value but it also has high running time. This method produces good partitions but the drawback is it can partition graph only into 2^k blocks.

2.8.1.2 Random Partition

This method of partitioning is applied to small graphs, such as coarsened graphs, which are used in multilevel partitioning. This algorithm is easy for implementation and its time complexity is O (n). But it produces poor solution towards finding edge cut value. The number of edge cuts produced is lot greater than best known cuts.

2.8.1.3 Spectral bisection method

It is one of the excellent partitioning techniques used to divide graph into sub-graphs [27]. In this method, global information about the associability of the graph is maintained by computing the laplacian matrix with its eigen-values. The steps of the algorithm are as follows.

Algorithm 1 Spectral bisection

For every Graph $G = V, E$

1: Represent Adjacency Matrix A_{ij}
2: Compute Degree Matrix D
3: Compute Laplacian Matrix $L = D$ - A_{ij}
4: Eigen values(λ) of L gives a lower bound of optimal cost C of partition
5: Eigen Vector (V) corresponding to λ, called as Fiedler Vector.
6: Determine Median of V
7: For each node u_1 of G if u_i
8: It bisects the graph into two sub graphs based on the sign of corresponding vector entry.
9: Set up graph $G_1 = V_1, E_1$ and $G_2 = V_2, E_2$

2.8.1.4 Greedy Construction

Greedy construction algorithm is one of the constructive heuristic algorithms. Detailed algorithm is explained in algorithm 2.

Algorithm 2 Greedy Constructive Procedure

Input: Graph G = (V, E) desired size k of partition
Output: k partition of V

1: P $\leftarrow P_0, ..., P_{k-1}$
2: $V^{'} \leftarrow$ V
3: for p $\in$ [0, k-1] do
4: u $\leftarrow$ random vertex from $V^{'}$
5: $P_p \leftarrow \setminus$ {u}
6: $V^{'} \leftarrow V^{'}$ {u}
7: p $\leftarrow 0$
8: While $\mid v^{'} \mid > 0$ do
9: b $\leftarrow$ greedyfunction($V^{'}$, P, p, G)
10: $P_p \leftarrow P_p \cup$ {b}
11: $V^{'} \leftarrow V^{'} \setminus$ {b}

Initially, all partitions are empty, k different random vertices are also known as seed vertices are assigned to those empty partitions. Remaining vertices are greedily assigned to each partition in a circular fashion. Total n-k iterations are performed. Greedy function is used to build k way initial partitions [68].

This algorithm is specifically based on the selection of seeds. If bad seeds are chosen, then leads to a problem of poor quality in terms of finding number of edge cut value.

2.8.1.5 Graph Growing Partition Algorithm

This is the heuristic graph partition construction algorithm based on greedy function. It performs k iterations. Each iteration selects a random vertex and grows around it. It is based on depth-first search procedure and checks for n/k vertices are added to particular partition [68].

This algorithm requires a lesser number of iterations as compared to greedy partitioning algorithm, but it suffers from the same problem of poor quality partitions with chosen of bad seeds.

2.8.2 Refinement Algorithm

A balanced bipartition A, B is achieved by swapping subsets $X \epsilon A$ and $Y \epsilon B$ with $\mid X \mid = \mid Y \mid$ to opposite sides so that cut size is reduced. One of the best refinement algorithms is Kernighan Lin Algorithm [69]. Furthermore, it is replaced with Fiduccia and Mattheyeses algorithm [70]. It uses the bucket data structure leads to reduce time complexity from O (n^2) from $O(m)$.

2.8.2.1 Kernighan Lin Refinement Algorithm

Kernighan and Lin proposed KL- local search and iterative method to perform graph partitioning. The input to the algorithm is an undirected graph $G = (V, E)$ with vertex set V, edge set E, and may have numerical weights on the edges in E. If the graph is a weighted graph, then its weight is considered and if it is unwieghted graph, then unique weight is considered for computation. The goal of the algorithm is to partition V into two disjoint subsets X and Y of equal (or nearly equal) size, in such a way that it should minimize the sum T of the weights of the subset of edges that cross from X to Y . [69]. Algorithms start with any partition of *V (G)* into X and Y . The algorithm maintains and improves a partition in each pass by using a greedy algorithm to pair up vertices of A with vertices of B in such a way that moving the paired vertices from one side of the partition to the other will improve the partition. Swapping of vertices in two different subsets helps to decrease the edge cuts obtained. The algorithmic steps are based on basic computations of cost and gains are discussed below.

- Internal Cost: It is the cost of edges connecting node v_i within its own group. Internal cost of vertex $v_i \in$ X is the addition of edge weights of adjacent vertices of v_i in the same partition that is X only. Likewise, for each node, the internal cost is computed.

$$I(\mathrm{V}_i) = \sum_{V_j \in X} \mathrm{w}\ (v_i, v_j) \tag{2.3}$$

- External Cost- It is cost of edges connecting node v_i within other group. External cost of vertex $v_i \in$ X is the addition of edge weights of adjacent vertices of v_i in other

partition that is Y only. Likewise, for each node, an external cost is computed.

$$E(\mathrm{V}_i) = \sum_{V_j \in B} \mathrm{w}(v_i, v_j) \tag{2.4}$$

- The difference between External cost and internal cost for each vertex where v_i V is known as the variation of cost computed as

$$G(\mathrm{v}_i) = \mathrm{E}(v_i) - \mathrm{I}(v_i) \tag{2.5}$$

- If vertices v1 and v2 are exchanged, then their gain are computed by using formula

$$g(\mathrm{v}_1, v_2) = \mathrm{G}(v_1) + \mathrm{G}(v_2) - 2\mathrm{w}(v_1, v_2)) \tag{2.6}$$

- Then swap the pair of vertices for which gain value is maximum
- if v_1 and v_2 are interchanged, then mark them as locked vertices
- for all such unlocked vertices from both divided sets excluding v_1 and v_2 , update the gain values by formula

$$\mathrm{g}^{'}(v_i) = g(v_i) + 2w(v_i, v_1) - 2w(v_i, v_2), \forall v_i \in x - \{v_1\} \tag{2.7}$$

$$\mathrm{g}^{'}(v_j) = g(v_j) + 2w(v_j, v_2) - 2w(v_j, v_1), \forall v_j \in x - \{v_2\} \tag{2.8}$$

- Repeat above procedure, if no more vertices are to swap.

If an initial partition is good, then KL algorithm is more efficient and faster. The complexity of this algorithm is O ($|\mathrm{E}|^2 \log|\mathrm{E}|$). Algorithm 3 gives major steps of Kernighan Lin algorithm.

Algorithm 3 A Heuristic Kernighan Lin Procedure

Kernighan-Lin (G (V, E))

1: Determine a balanced initial partition of the nodes into sets X and Y
2: do
3: compute G values for all V1 in X and V2 in Y
4: Let g_v, a_v and b_v are empty lists
5: for (n := 1 to | V |/2)
6: find V_1 from X and V_2 from Y, such that g = G[V_1] + G[V_2] - 2*W(v_1, v_2) is maximal
7: remove V_1 and V_1 from further consideration in this pass
8: add g to g_v, a to a_v, and b to b_v
9: update G values for the elements of X = X \ v1 and Y= Y \ v2
10: end for
11: find k which maximizes g_{max}, the sum of g_v[1],...,g_v[k]
12: if ($g_{max} > 0$) then
13: Exchange a_v[1],a_v[2],...,a_v[k] with b_v[1],b_v[2],...,b_v[k]
14: until ($g_{max} \leq 0$)
15: return G(V,E)

The limitations of Kernighan Lin algorithm are in terms of time complexity and cut size. Time complexity is O (n^2) and unable to minimize cut size for imbalance structure of the graph. These parameters improvement is proposed by Fiduccia Mattheyses (FM) heuristic algorithm in which time complexity is reduced to O (m) and also deals with imbalanced partition by moving single vertex instead of swapping two of them.

2.8.2.2 Fiduccia- Mattheyes Heuristic

The Fiduccia-Mattheyses (FM) heuristic is beneficial for hyper-graph bi-partitioning. This is an iterative algorithm type developed with an aim of reducing net-cut costs. It works on the same principle of KL Heuristic; starts with random solution and sequence of moves are organized as passes. Differs from KL is that instead of moving the pair of vertices, each time single vertex is moved across the cut in a single move. It can handle unbalanced partition as well for which a balance factor is introduced. A special data structure known

as the bucket is used to select vertices to be moved across the cut to improve running time. Each possible move can cause a change in cost known as gain. A move with the highest gain is selected and executed. For the selected vertex with the highest gain, it is locked with consideration to the value of gain. After successful execution of the move vertex, it leads to change in gains of adjacent vertices, and their gains are updated. Selection and execution of best gain move, gain update are repeated until every vertex is locked. Then best solution seen during the pass is adopted as starting solution of the next pass. The algorithm terminates when a pass-fail to improve solution quality. All moves with the same gain are stored in a linked list representing a gain bucket [70].

Algorithm 4 Heuristic Fiduccia- Mattheyes Procedure

Input: A hyper graph with a vertex (cell) set and a hyper edge (net) set
Output: 2 partitions

```
1: n(i): ♯ of cells in Net i; e.g., n(1) = 4
2: s(i): size of Cell i
3: p(i): ♯ of pins of Cell i; e.g., p(1) = 4
4: C: total ♯ of cells; e.g., C = 13
5: N: total ♯ of nets; e.g., N= 4
6: P: total ♯ of pins; P= p(1) + .... + p(C) = n(1) + .... + n(N)
7: Area ratio r, 0 < r < 1
8: Cutsetsize is minimized by FMpass (gain container, partitionment)
9: solution cost = partitionment.get cost();
10: while(not all vertices locked)
11: move = choose move();
12: solution cost ± gain container.get gain(move);
13: gain container.lock vertex( move.vertex() );
14: gain update(move);
15: partitionment.apply(move);
16: roll back partitionment to best seen solution;
17: gain container.unlock all();
```

2.8.3 Multilevel Technique

Barnard et al have introduced multilevel graph partitioning technique. This technique helps in accelerating existing graph partitioning tools. The main idea behind this algorithm is to group vertices together in order to deal with groups of vertices instead of processing on independent vertices in case of partitioning of the larger graph in k parts[91]. The Multilevel method is divided into following three steps in consideration with a weighted graph $G_0 = (V_0, E_0)$

Coarsening Phase: The graph G_0 is reduced into a sequence of smaller graphs G_0, G_1 ... G_m such that $\mid v_0 \mid > \mid v_1 \mid > \mid v_2 \mid > .... > \mid v_m \mid$. Random matching method is used for collapsing of vertices and to form a multi node.

Partitioning Phase: A 2-way partition p_m of the graph $G_m = (V_m, E_m)$ is computed that partitions V_m into two parts, each containing half the vertices of G_0.

Uncoarsening Phase: The partition P_m of G_m is projected back to G_0 by going through intermediate partitions P_{m1}, P_{m2}. . . P_1, P_0.

Multilevel bi-partitioning is shown in figure 2.13 and multilevel k-partitioning is shown in figure 2.14[91]. Coarsening deals with providing a global outline of a graph. Partitioning phase deals with partitioning of the original graph. Uncoarsening phase improves partition by preserving the fine structure of partition.

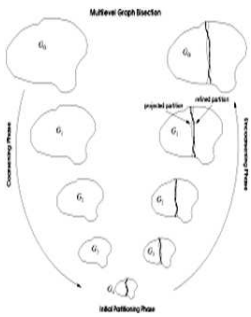

Figure 2.13: Multilevel Bi Partitioning

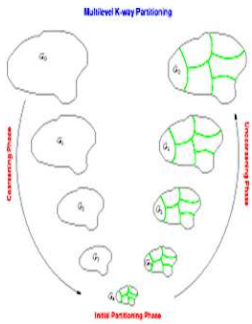

Figure 2.14: Multilevel K Partitioning

2.8.3.1 Graph Coarsening

It is an iterative method. Suppose Graph G_i has a set of vertices v_i are grouped together to form a single vertex v for the next level graph G_{i+1}. Vertex v is known as multi-node. V_i^v is the set of vertices of G_i which are grouped together to form v of G_{i+1}.

$$W(v) =\mid (V_{iv}) \tag{2.9}$$

The cut value of partitions in coarsen graph is the same as that of cut value for the partition in the original graph. Graph Coarsening can be performed in two ways. One is known as random matching and clubbing of matched vertices into a multimode. Another approach is based on the clustering phenomenon in which highly connected vertices generates multimode.

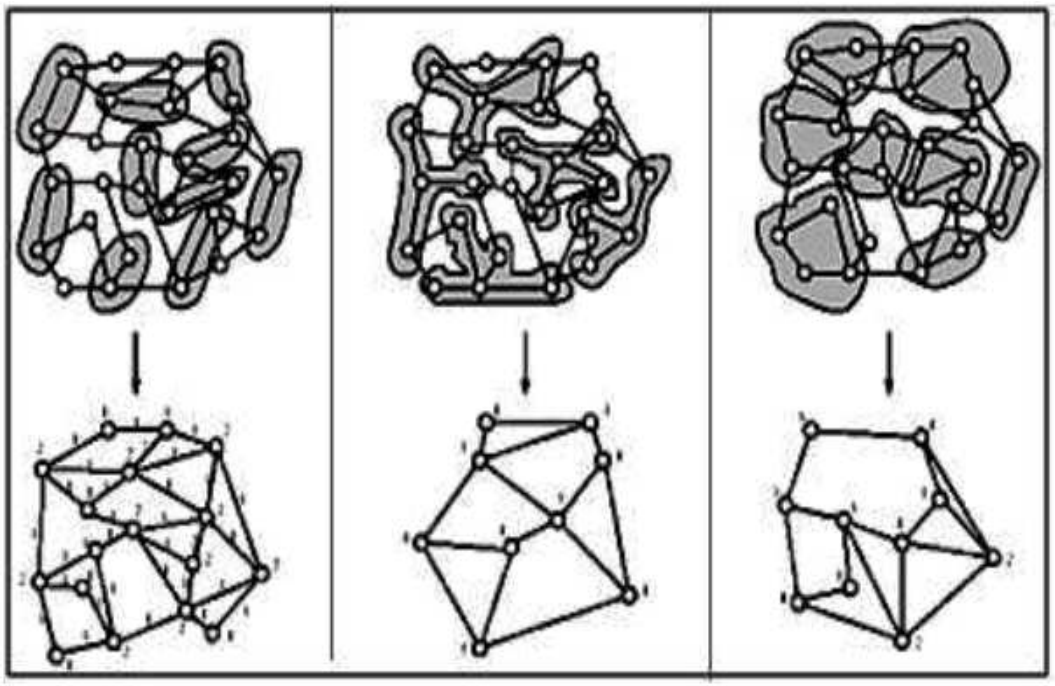

Figure 2.15: Graph Coarsening Examples

Matching is defined as a set of edges, in which no two edges are incident on the same vertex. Graph G_i is transformed into G_{i+1} by coarsening phase in which vertices to be matched are collapsed together to form a multi-node. When the vertices are collapsed, the size of the original graph is reduced. If the matching is of maximum size, then it is called a maximal matching.Example of maximal matching is shown on figure 2.16[91]. The size of maximal matching is different based on the matchings calculated. This maximal matching is useful in coarsening phase. The Complexity of all these schemes is O ($\mid E \mid$).

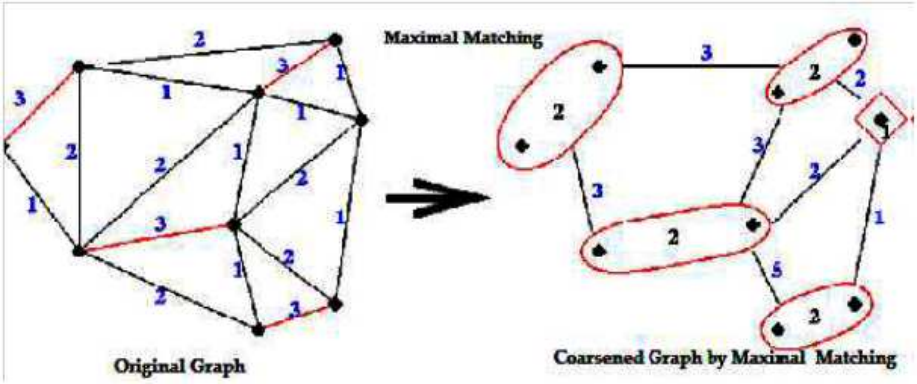

Figure 2.16: Graph Coarsening by Maximal Matching

Random Matching

The Randomized algorithm helps for generating maximal matching and more efficiently. The Major assumption is vertices are visited in random order including the following steps.

- Consider a graph $G(V, E)$ where a vertex u has not been matched yet for any of its unmatched adjacent vertex is selected randomly

- If such a vertex v exists, then edge (u, v) is included in the matching and mark vertex u and v is being matched

- If there is no unmatched adjacent vertex v, then vertex u, remains unmatched in the random matching

Heavy Edge Matching (HEM)

This technique is much similar as of maximal matching but uses weights of the graph. The main objective of HEM is to find maximal matching of the graph in addition to minimization of edge cut. In this technique, the vertices are selected in random order. Instead of randomly matching a vertex u with one of its adjacent unmatched vertices, match u with the vertex v such that the weight of edge (u, v) is maximum.

2.8.3.2 Graph Partitioning

A two-way partition P_m of the graph $G_m = (P_m, E_m)$ is computed which partitions V_m into two parts, each containing half the vertices of G_0. There are various algorithms available for graph partitioning; some of them are spectral bisection [30-32], geometric bisection [33],

and combinational methods [34-36]. Karpys and Kumar developed three algorithms for partitioning of the coarser graph. These are spectral bisection [38], graph-growing heuristic (GGP) which randomly selects a vertex v and grows in a breadth-first manner until half of the vertex weight has been integrated. Another is the second graph-growing heuristic (GGGP) which also randomly select a vertex v and includes only those vertices whose inclusion lead to the smaller increase in edge cut. As these both techniques reside on the selection of vertex v, different vertex v is selected, partitions are computed starting from selected vertices and best is used as initial partition [38].

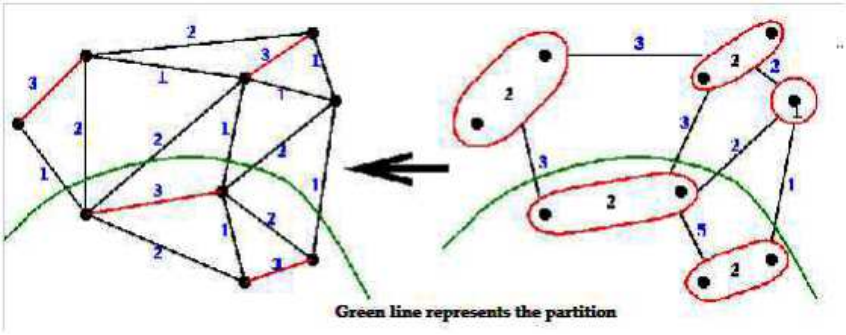

Figure 2.17: Converting Coarse Partition into Fine Partition

2.8.3.3 Graph Uncoarsening and Refinement

In uncoarsening phase, the partition of the coarsen graph (Pm) is projected back to the original graph, G_0 by going through intermediate partitions P_{m-1}, P_{m-2}... P_1, P_0. While projecting back, it maintains the sum of vertex weight in each set. Refinement algorithm selects two subsets of vertices, one from each part such that when swapped resulting partition has smaller edge cut. If X and Y are two parts of bi-sections, a refinement algorithm selects $X \epsilon X$ and $Y \epsilon Y$ such that X $\setminus$ $X \cup Y$ and Y $\setminus$ $Y \cup X$.

Based on the above mentioned portioning algorithms, various researchers worked on these algorithms to summarize few are Chao Wei Ou and Sanjay Ranka have suggested Parallel Incremental Graph Partitioning using Linear Programming [39].It is used to execute several scientific and engineering applications parallel. It requires partitioning of data among processors to balance the computational load on each node with minimum communication overhead. There are many algorithms like geometric, structural, Spectral and refinement algorithms are proposed for achieving parallel graph partitioning.

Researcher Stephen Barnard and et.al has suggested the fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems which are used for the partitioning of the graph. It finds application for dynamic graph too. The Dynamic graph is a graph which changes over time that is a small number of nodes or edges may be added or deleted at any given instant. The drawback of the method is the initial partition is to be calculated using a linear programming based bisection method [40]. The Proposed approach focuses on uniform partitions creation with no loss of information.

Researcher Inderjit S. Dhillon and et al. has described an equivalence between the objective functions and high-quality multilevel algorithm that optimizes various weighted graph clustering objectives such as ratio cut, normalized cut and ratio association criteria [5].

Mahmudur Rahman and et al. discusses and efficient graphlet counting method for large graph analysis, which computes the cost for obtaining the frequency of each graphlet in the network [6]. Also, the local topological structure is considered for the computation of the Graph Frequency Distribution (GFD).

Vladimir Batagelj and et al. described (X, Y) - clustering and Hybrid visualizations technique for visual analysis of large graphs. In (X,Y) clustering the two important properties like intra-cluster and inter-cluster were used for topological graph [7]. By using hybrid visualizations clusters were able to explore the clusters without losing their mental map.

Jose F. Rodrigues and et. al had discussed Large Graph Analysis in the GMine System for the clutter reduction in the graph which is based on graph representation as hierarchies of partition by using concept of super-graph and sub-graph. And graph summarization is performed using Center Piece summarization. Their main contribution is towards the large graph investigation in terms of locally and globally [1].

Liu and et. al. Provided Social Network Analysis, co-authorship networks and their combination[109].He mainly contributed for computing page rank, authors Rank and some coefficients of an author.

Han and et. al. Performed analysis of DBLP dataset to find the supportiveness of author[110]. Value of supportiveness is based on co-authorship ties in a non-symmetric ways.

Zdenek Horak and et al. developed FORCOA.net as an interactive tool for exploring the significance of authorship network in DBLP data [111]. This tool mainly focuses on analysis and visualization of the co-authorship relationship based on their joint publication

and intensity of author. The analysis is performed by using a forgetting function which holds publication information relevant to the selected date.

2.9 Graph Visualization

It is representation of interconnected nodes arranged in space and its representation in particular structure so that user can understand it easily. Graph visualization comprises of graph layouts and graph visualization techniques.

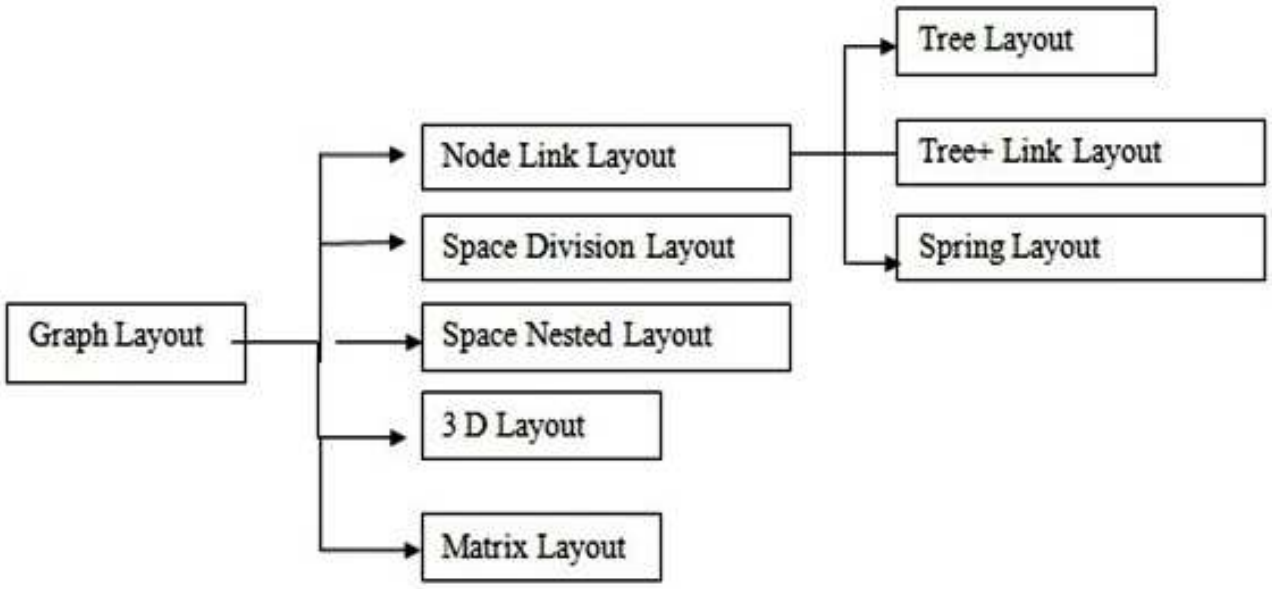

Figure 2.18: Graph Layouts

2.9.1 Graph Layouts

Representation of graph considered as graph layout. It includes trees and more general networks. Figure 2.18 shows graph layouts. It is mainly composed of node-link layout, space division layout, space nested layout, 3d layout and matrix layout.

i. Node Link Layout: It shows a relation among set of nodes and set of edges. A Position of node computed and nodes connected by drawing curve. Generally accepted rules include: evenly distribution of nodes and edges, avoidance of edge crossing, display of

isomorphic substructure and minimization of edge bends. It is further classified into three types such as Tree Layout, Tree+ Link Layout, and Spring Layout.

- Tree Layout: It shows parent- child relationship to indicate a link between nodes as shown in figure 2.19[71,73]. It suffers from a major problem of inefficient use of screen space. It wastes the root side of the tree and severely clutters the opposite side.

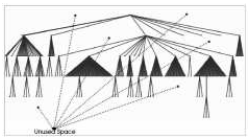

Figure 2.19: Classical Hierarchical View for A Moderate Large Tree

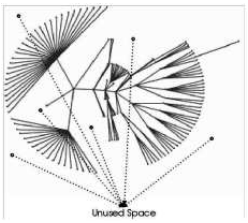

Figure 2.20: Radial View

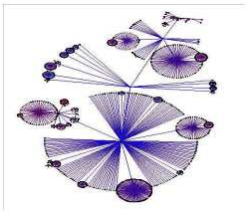

Figure 2.21: Balloon View

This problem can be avoided by another node-link layout called as radial layout as shown in figure 2.20[71]. It recursively positions children of a sub-tree into a circular wedge shape according to their depths in the tree. Generally, radial views including its variations share common characteristics .The focus node is always placed at the center of the layout, and the other nodes radiate outward on separated circles.

Balloon layout is shown in Figure 2.21 is similar to a radial layout[72]. Balloon layouts are formed where siblings of sub-trees are placed in circles around their father node.

- Tree+ Link Layout: Tree Plus enables users to interactively explore a graph by starting at a node and then incrementally expanding and exploring the graph. For example, Web Map [72] gives a visualization of users web browser history. If a webpage is visited for the first time, a new node is added and connected with

its predecessor as part of the underlying tree. However, if the webpage is visited before, a cross link is created to connect the node with its predecessor as shown in figure 2.22[74].

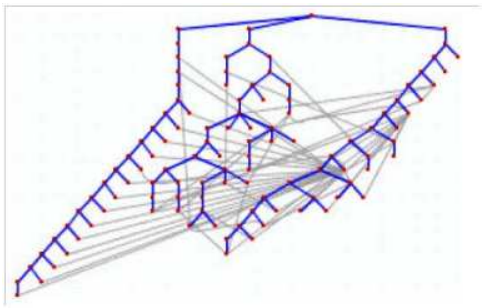

Figure 2.22: Tree + Link Layout

ii. Space Division Layout: In space division layouts, the parent-child relationship is indicated by attaching child node(s) to the parent node. Since the parent-child and sibling relationships are both expressed by adjacency, the layout has clear orientation to differentiate these two relationships as shown in figure 2.23[75].

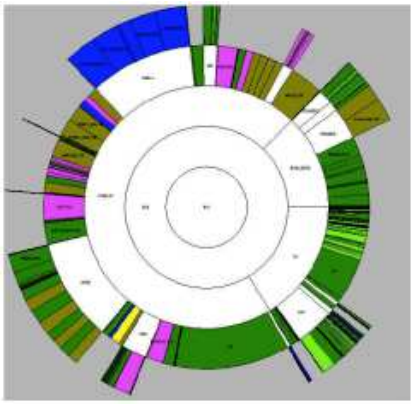

Figure 2.23: Sun Burst Visualization of A File Directory

iii. Space Nested Layout: It uses nested way for drawing the hierarchical structure. A rectangle is subdivided into smaller rectangles horizontally or vertically. An outer large rectangle represents the parent, while the smaller rectangle represents one of its children. Size of rectangle is in proportion with attributes of a node. It is also referred to as treemaps, as it is the most compact display among three layouts [76].

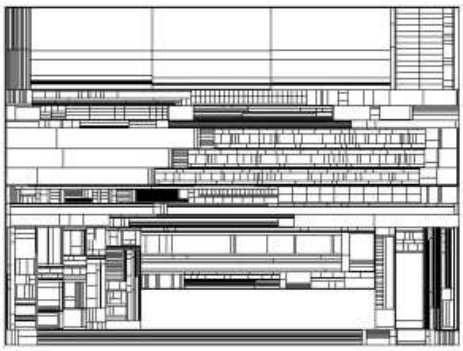

Figure 2.24: Tree Maps Views: Tree Maps View of A File Directory

iv. 3D Layout: Many 2D layouts are extended to 3 D layouts because 3D layouts provide an extra dimension in terms of more space for displaying large structures [75]. Cone tree structures as shown in figure 2.24 is one of the examples of 3D layout[75]. The parent-child relationship is indicated by using cone in which parents are placed at the apex of cone where as children are placed at the base of cone.

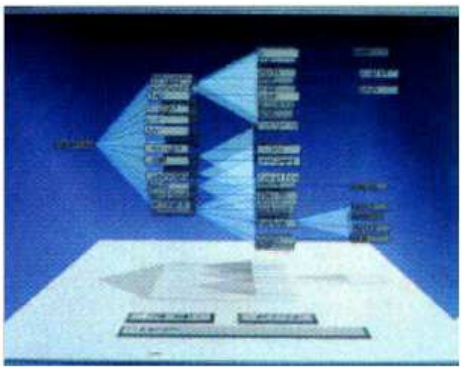

Figure 2.25: Cone Tree Layout

v. Matrix Layout: Another approach for graph visualization is a matrix layout. Graphs can be presented by their connectivity matrixes as shown in figure 2.26[80]. Each row and each column corresponds to a node. If it contains an edge from i to j, then it is represented with an interaction of (i, j).

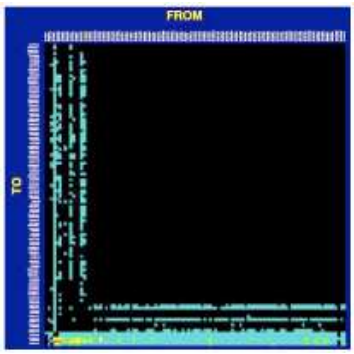

Figure 2.26: Matrix Views

2.9.2 Graph Visualization Techniques

The large graph problems are cannot be solved using static layout techniques hence for the computerized information visualization, the most popular schemes are Interaction and navigation. For solving these problems many visualization techniques have been developed. Graph Visualization Techniques are mainly classified into two types namely visual clutter reduction and interaction and navigation as shown in figure 2.27.

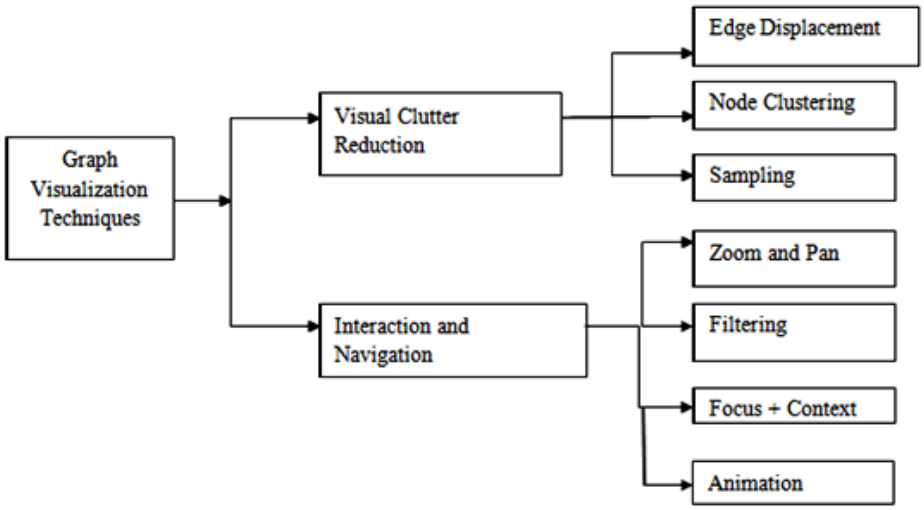

Figure 2.27: Graph Visualization Techniques

2.9.3 Visual Clutter Reduction

Visual clutter is one of the problems of a large graph. A layout should have minimum visual clutter. The different ways used for it are edge displacement, node clustering, and sampling

[63].

2.9.3.1 Edge Displacement

The best method for reducing the edge clutter is to minimize edge crossing. To achieve it, the best solution is to draw edges as splines and polylines. If geographical positions of a graph are fixed, then edge drawing becomes easier. To handle large graph and find its optimal solution is very time- consuming task. Hence, if a layered graph is constructed without edge crossing then it leads to more zigzag lines which is not valid regarding visualization of information.

Some methods have been proposed by researchers to reduce edges. One of the methods is a flow map layouts proposed by Phan et al[81]. However, flow map layout can only be applied to a set of edges which share a common end point and draws them as a freestyle binary tree layout as shown in figure 2.28[65].

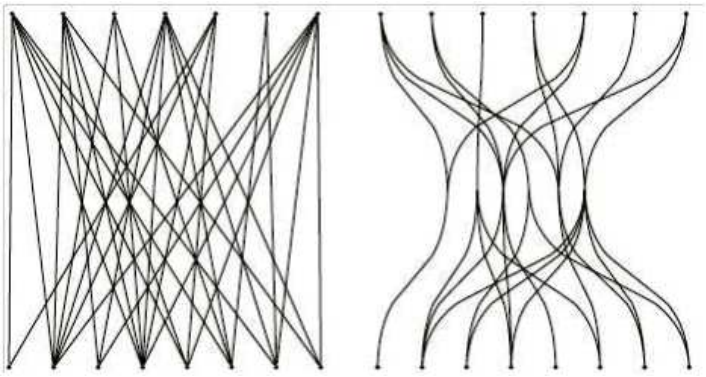

Figure 2.28: An Example of Flow Map: Migration

Confluent drawing: In which, the lines are drawn as curves for smooth intersecting arcs and it leads to reduce edge crossing[82].Two nodes are connected if and only if there is a smooth curve is present without ant sharp turns as shown in figure 2.29[82].

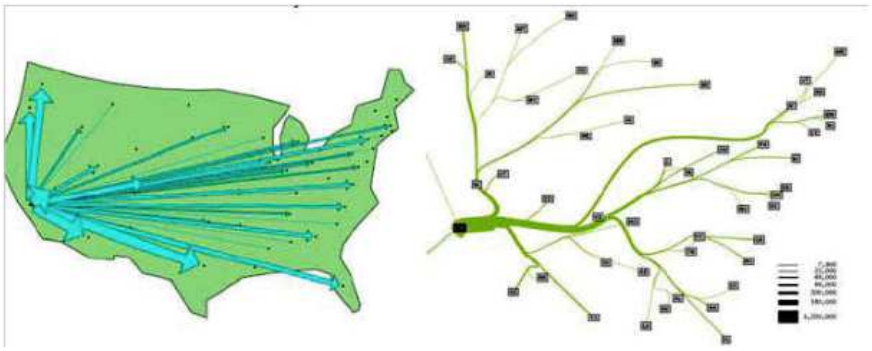

Figure 2.29: Confluent Drawing of Layered Drawings

In the confluent drawing layout, edge clustering approach is used, which focuses on the reduction of edge covering area instead of edge crossing. With the edge merging, freer space is available which causes visual clutter reduction. The most important feature of edge clustering is that it provides a simple and clear picture of the whole graph[82].

Figure 2.30 shows hierarchical structure formation where common endpoints are computed and treated as tree roots, then leaf positions are identified, leaves are preserved[83]. The line widths are proportional to edge weights. Flow maps are provided which helps in reduction of visual clutter. It works well for small graph visualization but fails for overlapped flow maps which generate difficult patters. This drawback can be further improved by another clustering approach known as edge bundles.

Figure 2.30 shows edge bundles where bundle width naturally indicates links connectivity with different parts of a hierarchical structure. In this algorithm, if tree path connects links two endpoints, then that link is curved. All curves are clustered at that segment. Bundles indicate how many links are connecting different parts of the hierarchical structure.

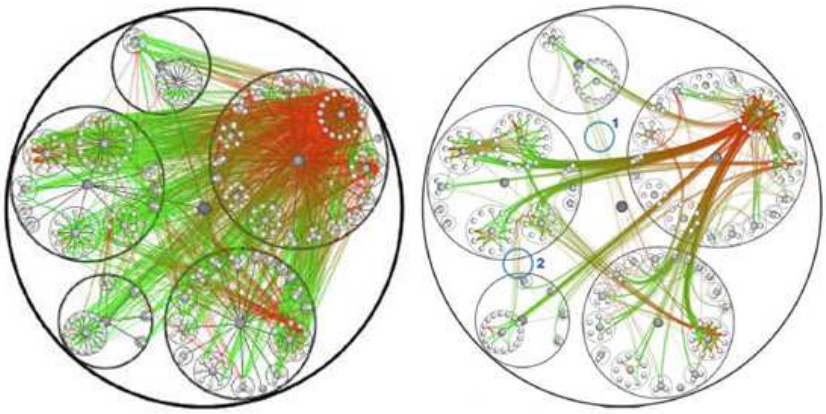

Figure 2.30: Example of hierarchical edge bundles: (a) and (b) show a balloon layout of a Software system and its associated call relations and its bundled result.

Wong et al. presented edge clutter removal strategy named as EdgeLens [84]. It depends upon users request. In this strategy, node positions are fixed, EdgeLens can effectively curve graph edges away from their focus. This structure is shown in figure 2.31[84].

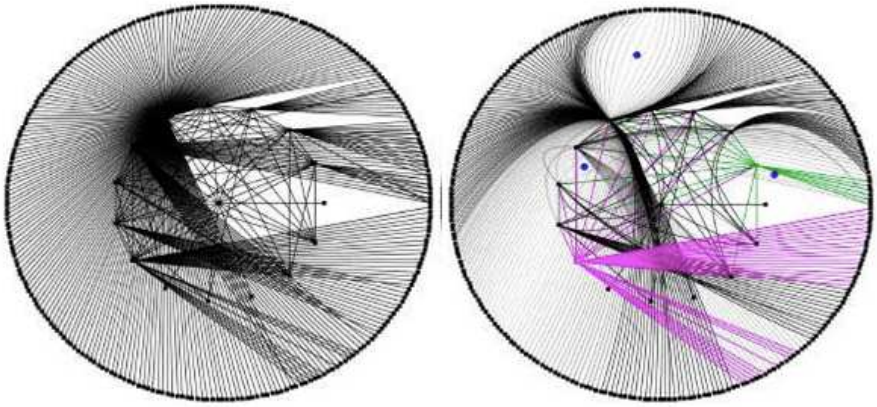

Figure 2.31: Example Of Edge Lens: (A) A Simple Radial Layout With Dense Edges. (B) Edge- Lens Views with Color and Transparency Enhancement

2.9.3.2 Node Clustering

In general, Clustering means a grouping of nodes. It has many applications in various domains such as cluster analysis, grouping, classification, and pattern recognition. For a specific context like visual clutter reduction, clustering refers to divide the whole graph into subgraphs. Representing those subgraphs as a single node, a grouping of similar elements

and free the space. This works well to reduce visual clutter. Mainly there are two types of node clustering as content-based and structure based.

1. Content-Based: This is application relevant approach suitable for specific problems and not a generalized one. It is based on semantic data related to graph elements. It produces meaningful clustering results and explains a method for aggregation of vertices depending on attributes. It computes attribute values like pivot tables in spreadsheet calculators [84, 85].

2. Structure based: This is a generalized approach of clustering and hence more preferred than the content-based approach. In this approach, clusters are generally formed based on graph components that have more intra link connections than the outside elements. In this type of design, various heuristics such as connectivity, cluster size, geometric proximity and statistical variation are proposed [86-88].

Structure based approach is classified into three types as graph theoretical, singe pass and iterative algorithms.

1. Graph Theoretical: in this technique, similarity between individual nodes is computed and similarity matrix is formed. Closely related node forms the cluster, where each cluster indicates connected graph. Different techniques used in graph theoretical are spectral bisection [89], spectral quadra section [90], and octa section [91].

2. Single Pass: In this technique, the individual data seeds are known as cluster seeds. The nodes are added one by one until the big cluster is formed. In this way, clusters are formed by growing individual cluster node [92], [93].

3. Iterative algorithms: It works in iteration and forms hierarchical structure by considering single pass clustering as a starting point. It merges smaller clusters into one to form larger one [94-97]. It involves three more steps such as coarsening, partitioning, projecting and refining of a graph.

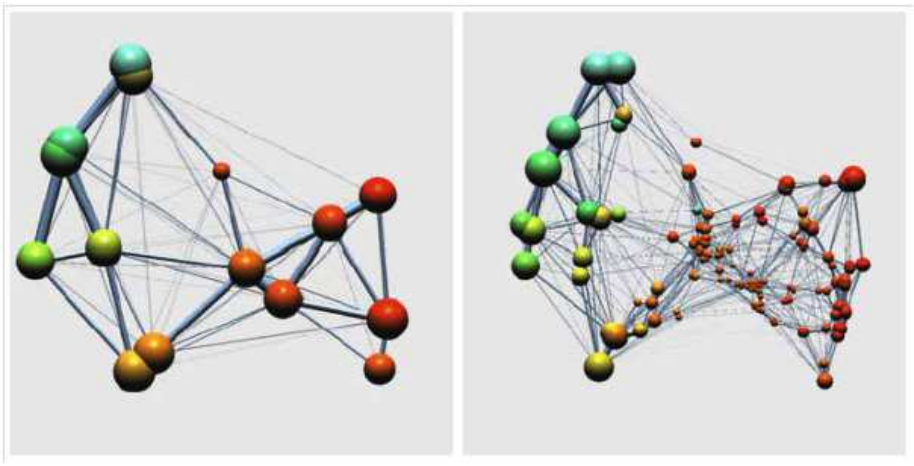

Figure 2.32: Different Clustering Levels of the Same Graph

Although quality of graph is dependent on its application domain. Some general characteristic of good clustering algorithm for human perception as

- Balanced cluster: At each hierarchy level, each cluster size should be same and distribution of nodes must be as even as possible.
- Small cluster depth: In recursive decomposition, the no. of layers should be small.
- Convex cluster drawing: Each cluster should fit in a simple convex region
- Balanced aspect ratio: Cluster regions should not be too thin. It should be big enough.
- Efficiency: cluster computation should not take a long time.
- Symmetry: Display balance should be maximized.

Well arranged clusters can provide clear visualization to the user. Some algorithms provide two or three dimensional representation of the graph are discussed in [99-102] as shown in figure 2.33[104].

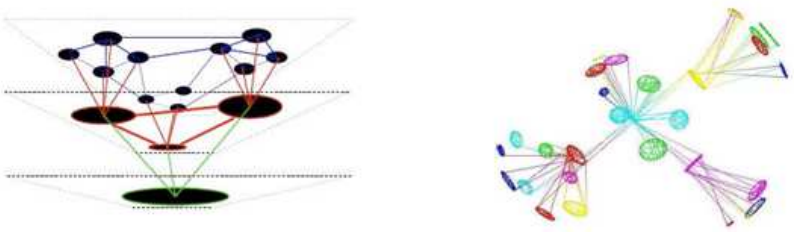

Figure 2.33: Clustered Graph Layouts: (A) Layered Layout (B) 3D Layout

2.9.4 Interaction and Navigation

Navigation and interaction are essential in computerized information visualization. It can help users to reveal detailed structures in large graphs. Yi et. al. proposed a summarization of popular interaction techniques [105]. Based on the purposes, they are categorized into seven groups:

- Select: It helps users to highlight certain focus targets, or request computer to process some specific items.
- Explore: It is used to change the current viewpoint to another part of the data in the same layout representation, such as panning and rotating.
- Reconfigure: It is used to switch between different layouts with the same representation scheme, such as replacing nodes in graphs and reordering data items based on different criteria.
- Encoding: It is used to switch different representation schemes, such as changing the layout from node-link representation to treemaps representation.
- Abstract/Elaborate: It adjusts the level of abstraction of a data representation to give users different insights into the data, such as zooming and clustering.
- Filter: It reduces the amount of data being displayed and makes the remaining items more visible based on user's request.
- Connect: It is used to highlight the connections between items or the items which are relevant to the focus item.

For large graphs, three of them are especially helpful: explore, abstract/elaborate, and filter. Remaining strategies like smooth panning and zooming, focus + context and animation to improve the quality of interaction.

2.9.4.1 Zoom and Pan

Zoom and Pan are important and fundamental tools for large information exploration. Zoom in provides detailed insight of data and Panning means smoothly moving the camera across a scene.

2.9.4.2 Filtering

Filtering refers to hide items from the view. It is a very simple concept based on the property of a filter in which data whose attributes are below some threshold, those are removed. Visual filtering interface should provide various visual browsing tools such as fast and continuous display of results, progressive refinement of parameters, Researchers Ahlberg et al. proposed the dynamic query filters for visual information seeking [106]. Their query parameters are rapidly adjusted with sliders, buttons and so on. A key to these principles is to understand the enormous capacity for human visual information processing.

2.9.4.3 Sampling

Sampling is an approach used for reducing visual clutter. It can be performed by focussing on abstract view of input graph. Clustering algorithms are classified into two types based on clustering criteria as natural clustering and content-based clustering.

1. Natural clustering: It is based on the structural information of graph nodes; hence also known as structural based clustering. It finds information patterns of nodes having some common criteria such as the distances between graph nodes and node degrees [62].

2. Content-Based clustering: It considers the semantic meaning of nodes which are having relations between them. This technique is rarely used due to its application dependent property. [62].

2.9.5 Focus +Context

Focus +context technique is mainly focused to overcome the drawback of zooming of losing the context. In zooming technique, a particular area is zoomed without having an idea of its surrounding. Where Focus+ Context provides an ability to see the focused object in a detailed view and overview of surrounding information [107]. Actually, this technique does not replace zoom and pan but complements them. Majority of the visualization system implements both techniques together. Fish-eye is one of the most popular focus context techniques [107].

2.9.6 Animation

It is one of the unique visualization techniques over other paper-based visualization techniques. It provides data exploration by adding time as an important factor. Animation helps users to understand the concept without thinking more [107]. Animated transition provides a better view of data relations.

2.10 Benchmark Datasets Available

Many Datasets available for graph partitioning, among them two applications are most important. One is in Very Large Scale Integration circuit designing and another is analysis and visualization of the co-authorship relationship based on the intensity and topic of joint publications.

2.10.1 for VLSI Circuits and sparse matrix ordering

The standard dataset by Walshaws benchmark for graph partitioning are as follow [116]-

- BCSSTK31(BC31): It is a 3D stiffness matrix which contains 35588 vertices and 572914 edges.
- BCSSTK32 (BC32): It is a 3D stiffness matrix which contains 44609 vertices and 985046 edges.
- BRACK2 (BRCK): It is a 3D finite element mesh matrix which contains 62631 vertices and 366559 edges.
- 4ELT: It is a 2D finite element mesh matrix which contains 15606 vertices and 45878 edges.
- COPTER2: It is a 3D finite element mesh matrix which contains 55476 vertices and 352238 edges.
- CANT: It is a 3D stiffness matrix which contains 54195 vertices and 1960797 edges.
- ROTOR: It is a 3D finite element mesh matrix which contains 99617 vertices and 662431 edges.

2.10.2 for Social Network Analysis

DBLP is a computer science bibliography website. DBLP originally stood for Database systems and Logic Programming. As a backronym, it has been taken to stand for Digital Bibliography Library Project. DBLP has collection of more than 3.66 million journal articles, conference papers, and other publications on a computer. All important journals and proceedings papers on computer science are tracked ($http://dblp.uni-trier.de/xml/$)[121]. The file dblp.xml contains all bibliographic records which make DBLP. It is accompanied by the data type definition file dblp.dtd. You need this auxiliary file to read the XML file with a standard parser [4]. dblp.xml has a simple layout: record 1 ... record n
These tags correspond to the entry types used in BibTEX[5]. DBLP records may be understood as BibTEX records in XML syntax +:
<article key="journals/jmm2/PatilKBK10" mdate="2017-05-26">
<author> Varsha H. Patil</author>
<author> Gajanan K. Kharate</author>
<author> Dattatraya S. Bormane</author>
<author> Snehal M. Kamlapur</author>
<title> Super Resolution for Fast Transfer of Graphics over Internet.</title>
<pages> 71-78</pages>
<year> 2010</year>
<volume> 5</volume>
<journal> Journal of Multimedia</journal>
<number> 1</number>
<ee>https://doi.org/10.4304/jmm.5.1.71-78</ee>
<url>db/journals/jmm2/jmm5.htmlPatilKBK10</url>
</article>

Various attributes of Record are-

- key : key is unique key of record. It shows UNIX file system with slash separation. The sub trees in the key namespace are for papers published in journals, transactions and magazines. Second part of DBLP depicts conference series or periodicals. Last

part of key contains sequence of alphanumeric characters with ids formed from authors name and year of publication.

- Mdate: Mdate is the of last modification of record. The format of date is YYYY-MM-DD. It provides facility of loading recent additions into application. It contains old versions of records.

- Title: This is one of the important element has to be exist in every DBLP publication record. It has sub elements for subscripts, sup elements for superscripts, i elements for italics, and tt for typewriter text style.

- Pages:It shows length of paper. Preferred style for page numbers is from to. For a single page paper, page number without hyphen is written. For articles in magazines, comma separated list of page numbers of page ranges are used.

- Years: The year element is a four digit number interpreted according to Gregorian calendar. For journal articles, it is assumed that date of publication of the issue is definite. For conference proceedings, the specification of year becomes tricky because sometimes proceedings are not published in the same year in which conference held. Hence year in which conference is held considered as a year in a record. For journal articles, the volume and number field are used to specify the issue in which paper appeared.

- URL and ee: DBLP record contains two URLs under this field. As URLs are of two types, local and global. Global URL is the standard internet URL starts with protocol specification of the form letter + : (http:, ftp:, .) . Locsl URLs does not start with protocol name.
  ee indicates the position of the electronic edition. ee contains the required link information of ACM and IEEE papers. Usually, the ee fields are global URLs.

2.11 Summary

A significant development in graph routing task, analysis has been seen in the last few decades, and literature in the form of journals, transaction papers, patents, reviews, and

surveys is available. Many of the research papers have been referred for knowing the status of research in the domain and identifying the research gap. It has been observed that the process of graph partitioning is the most important and challenging one. The major goals of partitioning are to speed up the design process, independent designing, no loss of original system functionality and simplification of routing tasks with objectives as to minimize interactions between blocks. Its accuracy directly impacts the quality of partitioning and loss of information in the form of connectivities through edges in relevance with the work undertaken the retrieval of author's information from DBLP dataset leading to association discovery.

2.11.1 Research Gaps

Though enough attention has been paid researchers, Literature survey reveals that the existing system for graph analysis has some limitations as -

1. **Existing systems are matrix based**

   As matrix-based approach is used for storing the input file to be partitioned, it suffers from difficulties in maintaining a connection among nodes and is expensive in terms of time. Time required is high so it takes more time for processing towards computation of eigenvector corresponding to second smallest eigen-value known as fiedler vector. It is claimed by many researchers that the number of edge cuts obtained is also more in case of matrix based approach [31].

2. **More communication overhead in parallel approaches**

   Many of the researchers have implemented multilevel approach by using bisection method. It runs more efficiently only on parallel computer only if a good partition of the graph is available. But its efficiency is poor as most of the time is spent for communication [37].

3. **Geometrical information is not available for graph partitioning**

   Many of the researchers have used geometric information of a graph for graph partitioning and it is less efficient technique because most of the time geometrical information is not provided and partitions obtained are optimized as compared to spectral

bisection method [33]. It is observed that less number of edge cuts not ensured by a geometric technique.

4. **Many of the techniques can handle only exact bisections**
   Popularly used Kernighan Lin heuristic algorithm performs partitioning which works in phases and locks vertices after each move. It is only applicable to handle exact bisections. It means a system can be decomposed into 2, 4, 8, 16 .....256 partitions with very high time complexity [31]. There is need to develop partitioning technique for decomposition not only in terms of bisection but also any number of odd partitions too.

5. **Currently available visualization systems is application specific**
   Visualization systems available are application specific, viz Gmine, Gephi, and forcoa.net which are developed for specific applications like VLSI designing, social network analysis etc.[111]. There is a need to develop application independent visualization system which will work accurately for all such applications.

6. **Existing author and co- author information retrieval system are based on dedicated server**
   An author and co- author information retrieval system proposed by Forcoa.net is the dedicated server designed for DBLP data set to retrieve authors performance measure parameters which need the support of silver light and browser- dependent system [111]. There is need to develop browser independent system with some precise performance measure parameters.

Chapter 3

Novel Algorithms for Analysis and Visualization of Large Graph

Graph partitioning and visualization systems are the core techniques for large graph analysis. Graph size and number of partitions to be formed are major concerns for any graph partitioning algorithm. A graph partitioning algorithm is called as reliable and efficient if it successfully partitions a large graph containing huge number of vertices and huge number of edges in given number of partitions without any inconsistency in actual information stored in a graph. The need for an efficient and reliable graph partitioning algorithm is the motivation behind this research work. The main goal of research work is to build novel algorithms for partitioning and visualization of large graphs in order to reduce time complexities and to achieve greater accuracy and better insights of details stored in graph during graph analysis. In this chapter the design procedure of algorithm and working of each developed algorithm for graph partitioning and visualizations are discussed in detail.

3.1 Overview

A graph partitioning problem is defined as - for a graph $G(V, E)$, where V is set of vertices and E is set of edges, partition V in k roughly equal subsets (partitions) such that the number of edges to be removed should be minimum. Let P denotes set of partitions obtained after partitioning a graph G into k number of partitions, such that P= {p_1, p_2,.....p_k | p_i, p_j

$\subseteq V$, $p_i \cap p_j = \varnothing$ and i $\neq$ j} [4]. Ideally, the partitioning algorithm should process a graph represented in simplest form as (v_i, v_j, e_{ij}) where v_i is source vertex, v_j is target vertex and e_{ij} is edge between v_i and v_j. Here e_{ij} may carry weight or label if graph $G(V, E)$ is weighted graph. While building the graph partitioning algorithm, the following objectives were of major concern -

1. Graph $G(V, E)$ needs to be evenly partitioned into disjoint partitions.
2. Number of edge-cuts needs to be minimum.
3. Algorithmic complexity in terms of time and memory needs to be minimum.
4. Data loss during partitioning, that occurs due to elimination of edge- cuts is to be avoided.
5. All sub-graphs needs to be able to visualize dynamically.

During partitioning, it is expected that there should not be an inconsistency occurring among generated partitions as interactions in graph carry vital information. To maintain consistency between original graph and generated sub-graphs, edge-cuts removal between two vertices is another important task. To achieve this goal, the main emphasis of a novel algorithm is on deciding number of partitions to be generated, selection of seed vertices while partitioning and visualization of resultant sub-graphs.

3.1.1 Number of Partitions to be Generated

Number of partitions is total subsets (k) of set of vertices V. An important aspect of graph partitioning is that these subsets are disjoint sets. Any two sets are disjoint when there intersection in null set. In earlier work of research, researchers used bisection partitioning technique for graph partitioning. In this technique, graph is recursively split into two balanced partitions [42], but this always do not holds as complexity increases in each level of partitioning and number of partitions generated are multiple of two, so there is a need to keep algorithm flexible which should split graph into k number of partitions such that 2 to $v - 1$.

3.1.2 Selection of Vertices While Partitioning

In partitioning process of graph $G(V, E)$, each vertex $v \in V$ where $V = \{v_1, v_2, ...v_n\}$ needs to be placed in a proper partition. In an early stage of partitioning, the difficulty is arises while choosing the first vertex in each partition around which partition has to be built. Researchers have designed some algorithms like greedy graph growing partitioning algorithm and multilevel partition algorithm to select a random vertex or set of vertices as initial vertices or a sub-graph for each partition and further, these partition graphs grows [15]. As these approaches are heuristic based, they require the number of additions and removals of vertices till it finds the best suitable partition for it. Some of the partitioning algorithms are refinement algorithms, like KL algorithm, which is based on the selection of two vertices randomly to bisect a graph into two sub-graphs [29, 37].

The performance of these algorithms depends on the quality of the bisection that it starts with. As a random selection of initial vertices may not always give correct results and it may lead to higher complexity in terms of time and space. If initial bisection is not correct then performance of partitioning algorithm degrades [29].

To overcome these shortcomings, it is most important to select appropriate initial (seed) vertices. For selection of seed vertices, an algorithm 5 is designed in which initially degree of each vertex v, (δ_v) in V is computed. Further vertices in V are sorted in decreasing order of their degrees. To partition graph G into k partitions such that $P = \{p_1, p_2, ...p_k\}$, seed vertex is obtained for each partition p_i using algorithm 5. First k vertices from the sorted V are considered as seed vertices such that v_i is seed vertex for partition p_i.

Algorithm 5 Degree based Seed Vertex Computation

INPUT: Graph $G(V, E)$, number of partitions to be generated (k)
OUTPUT: Seed vertices for partitions S
Procedure SeedVertexComputation ( )

1: **for** each vertex v in V of G **do**
2: $\delta_v \leftarrow$ degree of vertex v
3: **end for**
4: Sort V in descending order based on degree of vertices
5: **for** $i = 0$ to $k - 1$
6: Seed vertex for partition p_i,S[i],$\leftarrow$V[i]
7: **end for**
end procedure

This approach of selection of seed vertices exhibited correct results for some smaller graphs but failed to generate expected partitions for graphs having two or more vertices of same degree. As this approach considers first k vertices as seed vertices in an ordered manner, if degree of any $(k+1)^{th}$ vertex $\delta_{V[k+1]}$, equal to degree of the k^{th} vertex $_{V[k]}$, then the k^{th} vertex is considered as seed vertex, though $(k+1)^{th}$ vertex is more appropriate seed vertex than k^{th} vertex. It leads to an improper seed vertex selection and results in inconsistent graph partitioning, also the number of edge-cuts obtained is more than expected. Table 3.1.shows results obtained for some graphs using algorithm 5.

Table 3.1: Number of Desired Edge-cut and Obtained Edge-Cut

Sr. No.	Number of Vertices	Number of Edges	Number of Partitions to be Generated	Expected Number of Edge-cut	Number of Edge-cuts Obtained
1.	8	11	2	2	**2**
2.	10	11	2	2	**2**
3.	8	10	2	2	**2**
4.	8	13	2	03	**05**
5.	317	452	4	32	**100**
6.	313	768	5	171	**368**
7.	313	1194	5	651	**664**

To overcome these flaws of algorithm 5, it is required to modify algorithm 5. Algorithm

6 is the result of improvements done in algorithm 5. Algorithm 6 is based on set difference approach. In this approach, in order to determine appropriate seed vertex, first V is sorted in descending order based on degrees of vertices same as in algorithm 5. From sorted V maximum degree value, δ_{max} is obtained. δ_{max} is utilized significantly in algorithm 6 for selection of seed vertices. Vertices in V having degree value equal to δ_{max} are mainly considered as most suitable seed vertices M , where $M = \{v \mid v \in V \text{ and } \delta_v == \delta_{max}\}$.

If number of partitions k, in which graph g is to be partitioned is greater than $\mid M \mid$ then algorithm 5 is used for selection of seed vertices else set difference based approach is adopted to decide seed vertex for each partition p in P. In this approach, to decide whether vertex $v_i \in M$ is appropriate seed vertex or not, the differences between degrees of $M[i]$ and each $M[j]$ where $j = i + 1$ are calculated using equation 3.1 and termed as set difference of $D_{M[i]}$.

$$\triangle_{M[i],M[j]} = \delta_{M[i]-M[j]} \tag{3.1}$$

From the set difference obtained for vertex $M[i]$, the vertex $V[j]$ having minimum set difference is considered as seed vertex for partition p. The same process is repeated to determine seed vertices of remaining $k - 1$ partitions.

Algorithm 6 Set Difference based Seed Vertex Computation

INPUT: $GraphG(V, E)$, number of partitions (k)
OUTPUT: Seed vertices for each partition
Procedure SetDifferencebasedSeedVertexComputation ( )

1: **for** each vertex v in V in G **do**
2: $\delta_v \leftarrow$ degree of vertex v
3: **end for**
4: Sort V in descending order based on degree of vertices
5: $v \leftarrow V[0]$
6: Maximum degree for G, $\delta_{max} \leftarrow \delta_v$
7: $M \leftarrow \varnothing$
8: **for** each vertex v in V **do**
if $\delta_v == \delta max$ then
9: add v to M
10: **end if**
11: **end for**
12: **if** length$(M) < k$ **then**
13: **for** each vertex v in M **do**
14: Seed vertex for partition $p_i, \chi_{pi} \leftarrow v$
15: **end for**
16: **else**
17: **for** $i \leftarrow 0$ to $k-1$ **do**
18: set difference for $M[i], D_{M[i]} \leftarrow \phi$
19: **for** $j \leftarrow i+1 tolength(M)-1$ **do**
20: $\Delta_{M[i],M[j]} \leftarrow \delta_{M[i]-M[j]}$
endprocedure

The selection of seed vertices using algorithm 5 and 6 is illustrated for graph $G_1(V_1, E_1)$.

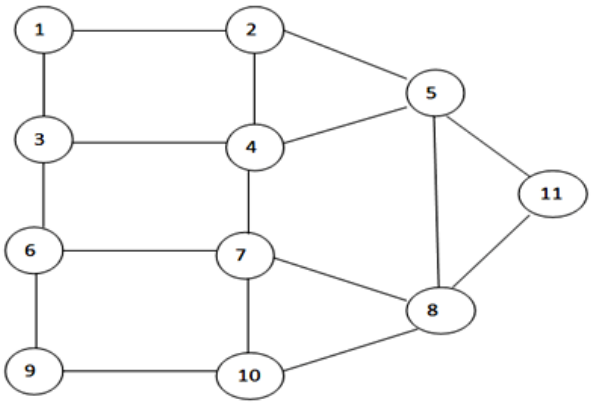

Figure 3.1: Graph $G_1(V_1, E_1)$

Assume Graph G_1 (V_1, E_1) as shown in figure 3.1 is to be partitioned into two partitions ($k = 2$). Algorithm 5 is applied to determine seed vertex for partitions (p_1 and p_2) and the intermediate results obtained during each step of algorithm 5 are discussed below-

Let $V_1 = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$, vertices in G_1

Degree of each vertex in V_1 is computed and stored in D_1

D_1= { 2, 3, 3, 4, 4, 3, 4, 4, 2, 3, 2}

Then V_1 is sorted in descending order based on degree of vertices. The modified values of V_1 and D_1 are below-

V_1= {4, 5, 7, 8, 2, 3, 6, 10, 1, 9, 11}

D_1= {4, 4, 4, 4, 3, 3, 3, 3, 2, 2, 2}

As graph $G_1(V_1, E_1)$ is to be partition in to two partitions, first two vertices from V_1 (M_1 = {4 , 5}) are considered as seed vertices. Vertices 4 and 5 are determined as seed vertex for partition p_1 and p_2 respectively. Partition p_1 and p_2 become p_1= {4} and p_2 = {5}.

Further, by using algorithm 7 remaining vertices from V_1 are evenly distributed into partitions p_1 and p_2. The partitioning process resulted in p_1= {1, 2, 3, 4, 6},p_2 = {5, 7, 8, 9, 10, 11}. The numbers of edge-cuts obtained are 5 as F = {2-5, 4-5, 4-7, 6-7, 6-9}.But this count is greater than the value of expected edge-cut count which is 4. This denotes that the partition was not occurred efficiently and it still has scope for improvement.

For proper and efficient partitioning of graph G_1 an algorithm 6 is applied for selection of seed vertices. As δ_{max} value for graph $G_1(V_1, E_1)$ is 4 and vertices whose degree equal to δ_{max} are M_1= {4, 5, 7, 8}. After applying algorithm 3.2 vertices 5 and 8 are obtained as

seed vertices for partition p_1 and p_2 respectively. Now partition p_1 and p_2 became p_1= {5} and p_2= {8}. Further using algorithm 7 remaining vertices from V_1 are evenly distributed into partitions p_1 and p_2. The partitioning process resulted in p_1= {1, 2, 3, 4, 5, 11}, p_2= {6, 7, 8, 9, 10} and edge cut F = {3-6, 4-7, 5-8, 11-8]. The value of count of edgecuts obtained $\mid F \mid$ is 4 which is equal to ideal edge cuts count for partitioning a graph G_1 into two partitions. Resulted two partitions for graph G_1 are as shown in figure 3.2 and 3.3 respectively.

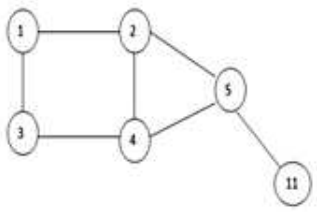

Figure 3.2: Partitioned sub-graph $G_1^{'}(V_1^{'}, E_1^{'})$

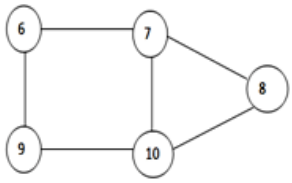

Figure 3.3: Partitioned sub-graph $G_1^{''}(V_1^{''}, E_1^{''})$

In order to evaluate accuracy of algorithm 6, it is applied on graphs of different sizes listed in table 3.1. Table 3.2 shows the ideal value of count edge- cuts and value of obtained edge cuts count for each graph. From table 3.2 it is observed that values obtained for edge cut counts are equal to ideal edge count values.

Table 3.2: Number of Desired Edge-cuts and Obtained Edge-cuts after Refinement

Sr. No.	Number of Vertices	Number of Edges	Number of Partitions	Expected Number of Edge cut	Number of Edge cuts Obtained
1.	8	11	2	2	**2**
2.	10	11	2	2	**2**
3.	8	10	2	2	**2**
4.	8	13	2	03	**03**
5.	317	452	4	32	**32**
6.	313	768	5	171	**171**
7.	313	1194	5	651	**651**

3.1.3 Graph Visualization

Literature survey reveals that the problem of graph visualization for generated dynamic partitions (sub graphs) is unhandled [90], hence visualization of generated sub graphs become important and need to be focussed. Algorithm 12 is designed for visualization of sub-graphs dynamically. Sub graphs generated as an output of partitioning are provided as input to the visualization system which displays these sub graphs as a network of vertices and edges.

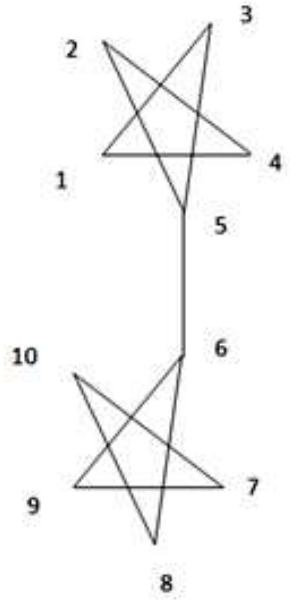

Figure 3.4: Partitioned sub graph $G_3'(V_3', E_3')$

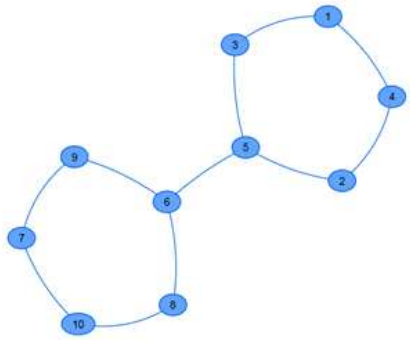

Figure 3.5: Partitioned sub graph $G_3''(V_3'', E_3'')$

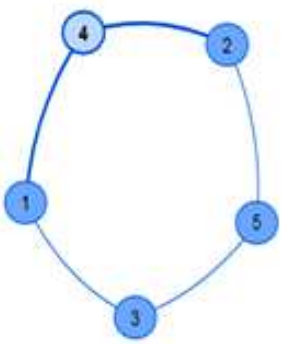

Figure 3.6: Partitioned sub graph $G_3^{'}(V_3^{'}, E_3^{'})$

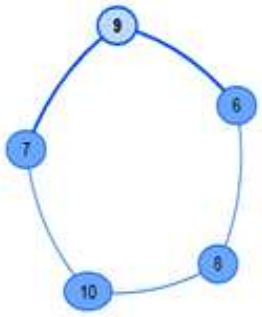

Figure 3.7: Partitioned sub graph $G_3^{''}(V_3^{''}, E_3^{''})$

Figure 3.4 shows an input graph $G_2(V_2, E_2)$, its visualization before partitioning is shown in figure 3.5. $G_2(V_2, E_2)$ is partitioned into two partitions. Figure 3.6 and 3.7 show the dynamic visualization of two sub-graphs $G_2^{'}(V_2^{'}, E_2^{'})$ and $G_2^{''}(V_2^{''}, E_2^{''})$ respectively with a number of edge-cut value as 1(5-6).

3.2 Efficient Partition Building and Cut set Computing Algorithm

The architecture of an efficient partitioning and edge- cut set computing system is as shown in figure 3.8. It comprises of functional blocks as - data set parsing, degree computation, vertices ordering, partition generation based on most connected vertex and partition size and edge-cut computation and sub-graphs visualization. The working of each block is discussed in brief.

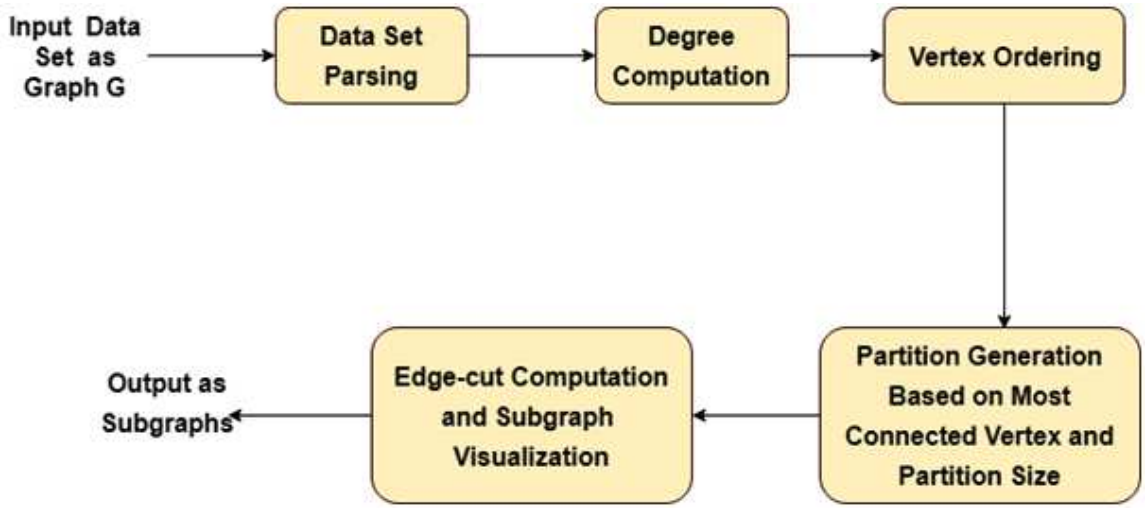

Figure 3.8: System Architecture for an Efficient Partitioning and Edge-cut Set Computing

Partitioning system accepts input as Graph $G(V, E)$ where V is a set of vertices and E is a set of edges. This system is based on finding stronger connected component with respect to seed vertices by eliminating minimum number of edges (edge- cut) to disconnect the whole graph into subgraphs (partitions).

Let $V = \{v_1, v_2, v_3, v_4 \ldots v_n\}$

$E = \{e_1, e_2, e_3, e_4 \ldots e_n\}$

n = Total number of vertices in graph $G(V, E)$

S = Set of seed vertices

3.2.1 Data Set Parsing

Graph $G(V, E)$ is given as an input for data set parsing. For each vertex v of graph $G(V, E)$, all adjacent vertices are found by verifying direct edge connectivity between v_i and remaining vertices of $G(V, E)$. It is represented in the form of adjacency list Al as $v_i \in \{v_1, v_2, v_3, v_4 \ldots v_n\}$, where $v_i \in V$ and $\{v_1, v_2, v_3, v_4 \ldots v_n\}$ are adjacent vertices(direct neighbour) of v_i.

3.2.2 Degree Computation and Vertex Ordering

From A_l obtained in the previous step, the degree of each vertex v_i is computed as δ_v and stored in D where $v_i \in V$. Further V is sorted in descending order based degree of vertices.

3.2.3 Partition Generation

Graph $G(V, E)$ is partitioned into k partitions based on stronger connected components. P is set of partitions represented as P= $\{p_1, p_2.....p_k\}$. Capacity for each partition is the maximum number of vertices from V can be placed in partition p_i and it is calculated using equation 3.2 and 3.3.

Let γ = Capacity of partition p_i

$$\gamma = (\mid v \mid /k) + 1 \text{ for } \mid v \mid \text{ is odd} \tag{3.2}$$

$$\gamma = (\mid v \mid /k) \text{ for } \mid v \mid \text{ is even} \tag{3.3}$$

$sizeof(p_i)$= Size of partition p_i (number of vertices present in partition p_i)

χ= Seed vertex

S = Set of seed vertices

$X = V - S$ = Remaining vertices of graph $G(V, E)$ excluding seed vertices.

Using algorithm 6, δ_{max} of vertices for graph $G(V, E)$ are determined and seed vertices are selected for each partition.Seed vertex χ in every partition is permanent and to distinguish seed vertex χ from other vertices, v_s assigned an initial weight for partition p_i as w_{χ,p_i} = 1.1.

After obtaining seed vertices for all partitions and assignment of initial weight to each seed vertex, now each partition p_i contains a single vertex. Next step is to place each vertex v from X into appropriate partition. For this vertex-partition weight (ω_v, p) is calculated for each vertex v in V using algorithm 7. The ω_v, p is the weight of vertex v for partition p and $W_v[p]$ stores the weight of v for all partition to be build. The ω_v, p is calculated by obtaining the distance of v from seed vertex of partition p using equation 3.3 and its value ranges in between 0 to 1.With reference to algorithm 10, remaining vertices from X are added to partition p in an ordered manner. The partition having maximum weight ω_v, p for vertex v is chosen as a suitable partition for vertex v.

If v is not member of p, then initial distance of v from p , $\sigma_{v,p}$ is set to 1. The value of the

distance is incremented by 1 if v is not adjacent to seed vertex of partition p. The value of distance is incremented till path from v to seed vertex of p is not obtained. If there is no path exist between v and seed vertex of p, then distance $\sigma_{v,p}$ remains 0. Using value obtained for distance $\sigma_{v,p}$ of vertex v to partition p, w_v, p is calculated using equation 3.3.

$$\omega_{v,p} = 1.0/\sigma_{v,p} \tag{3.4}$$

Algorithm 7 Weight Computation

INPUT: Graph $G(V, E)$, Partitions P, Vertex v
OUTPUT: Vertex Partition Weight W_v
Procedure WeightComputation()

1: Weights of vertex v for each partition p in P, $W_v = \{\phi\}$
2: **for** each partition p in P **do**
3: distance of v to partition p, $\sigma_{v,p} \leftarrow 1$
4: **for** each vertex, v_i in partition p **do**
5: **if** v and v_i are adjacent **then**
6: Weight of v for partition $p, \omega_{v,p} \leftarrow 1.0/\sigma_{v,p}$
7: $W_v[p] \leftarrow \omega_{v,p}$
8: break;
9: **else**
10: $\sigma_{v,p} \leftarrow \sigma_{v,p+1}$
11: **end if**
12: **end for**
13: **end for**
14: return W_v
endprocedure

$W_v[p]$ weights of v for all partitions in P is sorted in descending order. Partition p with maximum weight is considered as the most appropriate partition for v. If p is full to its capacity then it is required to find another suitable partition for vertex v. So before placing vertex v in partition p following cases are checked

Case 1: if $sizeof(p) < \gamma$ then vertex v is placed in partition p.

Case 2: If $sizeof(p) < \gamma$, it means partition p is full to its capacity and this signifies that

v cannot be placed in p. But as the weight of v for p is maximum in comparison to other partitions, so p is most suitable partition for v. In order to place v in partition p, it is required to move a vertex in p to other partition. Lets consider vertex v_x is to be moved from partition p to p_x in order to make place for v in p. To select most suitable p_x from remaining $k-1$ partitions ,weight of v_x for each partition p_i is calculated. Using these weights values first all possible partitions (T_v) to which v_x can be moved are obtained. A partitioned p_i in P such that $p_i \neq p$ is marked as possible partition if $\omega_{vx,pi} > w_{vx,p}$. After obtaining all possible partitions T_v, it is sorted in descending order based on weights. Vertex v_x from partition p is moved to first vacant partition p_x in T_v and vertex v is placed in partition p in the place of vertex v. If there is no any vacant partition in T_v then v_x remains in partition p and vertex v is marked as remaining vertex and it has not get assigned to any partition.

Case 3: If no partition is assigned for vertex v in case 2, then vertex v is considered as remaining_ vertex and it is added to X. Finally, all *remaining_ vertices* from X are placed in suitable partitions using the same process discussed for vertex v.

3.2.4 Edge-cut Computation and Graph Visualization

In the previous section 3.2.2, k numbers of partitions are generated for graph $G(V, E)$ by partitioning V into k number of subsets. The edge-cuts between partitions are found using algorithm 11. To find disconnected edges between two partitions, the adjacent vertices between these two partitions in Graph $G(V, E)$ are obtained and edges connecting these adjacent vertices are considered as disconnecting edges for these two partitions. Consider p_1 and p_2 are two partitions such that p_1= $\{v_1, v_2...v_i\}$ and p_2= $\{v_1, v_2...v_j\}$. To find disconnected edges between p_1 and p_2, it is confirmed that each $v_i \epsilon p_1$ and $v_j \epsilon p_2$ are adjacent vertices in Graph $G(V, E)$. If resulting edge e$(v_i \rightarrow v_j)$ is considered as a disconnected edge for partitions p_1 and p_2, e_{ij} is added to edge- cuts. Similarly, all possible edge-cuts between partitions p_i and p_j are determined where $p_i \in P$ and $p_j \in P - p_i$.

For a graph $G_3(V_3, E_3)$from DBLP dataset having 313 vertices and 450 edges is provided as an input to algorithm 10 and 11. Number of partitions to be formed are $k = 3$, partitioning algorithm resulted in 19 edge-cuts as listed in table 3.3 and dynamic subgraphs visualization of three subgraphs as $G_3'(V_3', E_3')$, $G_3''(V_3'', E_3'')$ and $G_3'''(V_3''', E_3''')$ is shown in

figure 3.9, 3.10 and 3.11 respectively.

Table 3.3: List of Edge-cuts Obtained

2686-3500	3023-2044	4459-4448	1731-3849
4750-2458	2236-3023	1877-3007	4383-2456
1788-3694	4983-2044	210-3066	266-4857
4707-2868	1039-1943	3081-2456	3868-4650
7617-52	1055-3728	1731-2715	

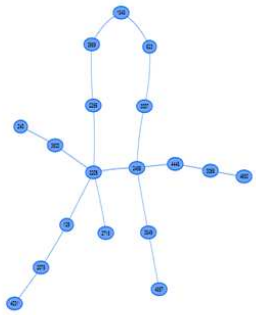

Figure 3.9: Partitioned sub-graph $G_3'(V_3', E_3')$

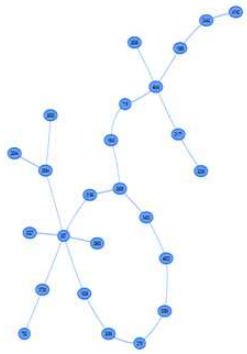

Figure 3.10: Partitioned sub-graph $G_3''(V_3'', E_3'')$

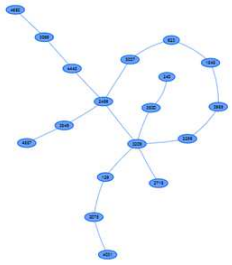

Figure 3.11: Partitioned sub-graph $G_3'''(V_3''', E_3''')$

3.3 Tool for Bibliographic Record Analysis

DBLP dataset is a collection of bibliographic record of computer science publications of various authors and co-authors. It contains approximately 1.5 million bibliographic records. This publication records are analyzed by ranking and profiling of authors, building author publication graph, author conference graph, and author co-author graphs. It is quite difficult to process these records as a whole. Hence this large dataset needs to partition into

smaller sub-graphs and process further for author co-author relationship visualization and to measure authors performance. System architecture for analysis of bibliographic record is discussed as shown in figure 3.12. It consists of major two blocks such as-DBLP pre-processing and DBLP processing and visualization.

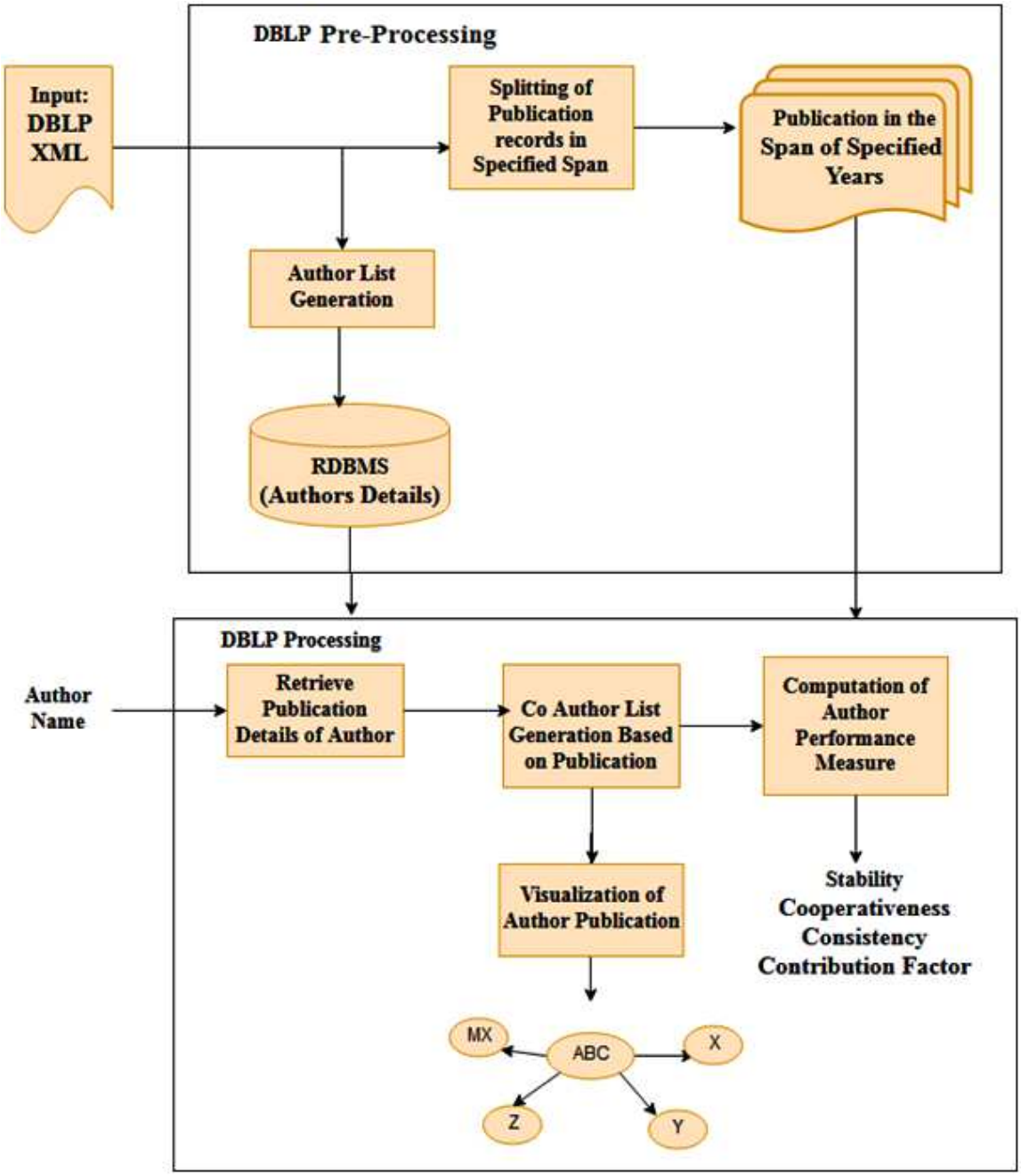

Figure 3.12: System Architecture of Author Information Retrieval

3.3.1 DBLP Pre-processing

DBLP dataset is provided as an input where bibliographic records are stored as XML nodes. As XML is very concrete and highly canonical, it is less suitable for representation of multiple interactions between two or more nodes as compared to a network of vertices and edges (graph). To increase the efficiency of analysis, DBLP dataset is transformed into the number of graphs, N.

DBLP dataset pre- processing is performed in two steps- author list generation and partitioning of publication records in a quantum of specified years. These two operations are simultaneously performed in cooperation to generate a unique list of authors and number of sub-graphs where each sub-graph contains publications published within a quantum of specified years.

Input DBLP XML file contains details of publications, each node from XML file represent a single article or publication and attributes of this node provides various detail of an article. One sample node from DBLP XML file is shown as below-

<article key="journals/jmm2/PatilKBK10" mdate="2017-05-26">
<author>Varsha H. Patil</author>
<author>Gajanan K. Kharate</author>
<author>Dattatraya S. Bormane</author>
<title>Super Resolution for Fast Transfer of Graphics over Internet.</title>
<pages>71-78</pages>
<year>2010</year>
<volume>5</volume>
<journal>Journal ofMultimedia</journal>
<number>1</number>
<ee>https://doi.org/10.4304/jmm.5.1.71-78</ee>
<url>db/journals/jmm2/jmm5.htmlPatilKBK10</url>
</article>

3.3.1.1 Author List Generation

In DBLP dataset, each node symbolizes a publication and attributes signify details of publication. A node may have more than one author attributes, in such a case, first author attribute represents the main author and further author attributes represent co-authors. From the DBLP dataset, all unique authors are retrieved. A unique id (author_ id) is assigned to each author. *author_ id, author_ name* and *partition_ label* represent unique vertex in a graph as shown in figure 3.13.

Let P= Set of Partitions in span of 5 years

$P_a = \{p | p \epsilon P$ and an author is active in $p\}$.

The detail of each author including *author _id, author_name* and *partition_label* $L(P_a) \epsilon l(p)$ in which an author is active are stored in RDBMS.

A fixed span of a year (quantum) is considered successively from year of first publication to current year for creating partitions (P). Partition $p \epsilon P$ is sub-graph (a network of author and co- authors). The process of partitioning is discussed in detail in section 3.3.3.2.

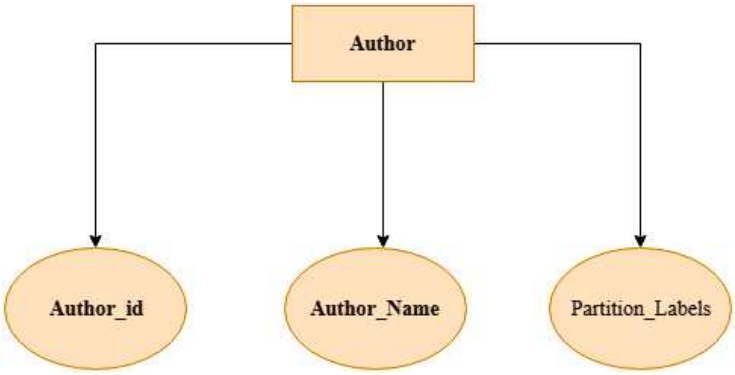

Figure 3.13: System Architecture for Author Information Retrieval

3.3.1.2 Partitioning of Publication Records in Specified Span

As DBLP Dataset is flooded with millions of publications, it is not feasible to store and process all publications in a single graph due to memory limitations and processing issues. This limitation is overcome by producing N number of sub-graphs of DBLP dataset based on the specified quantum of the years. A quantum of 5 years is considered for partitioning of DBLP dataset into partitions (P). Each publication in DBLP dataset is placed in a suitable partition based on its publication year. Each partition stores publication in form

of a network of vertices and edges ($G(V, E)$) where vertices represent author and edges represent an interaction between authors. Edge label, L (e) where ϵ E, signifies publication details between two authors. Figure 3.14 shows specified authors representation in each partition.

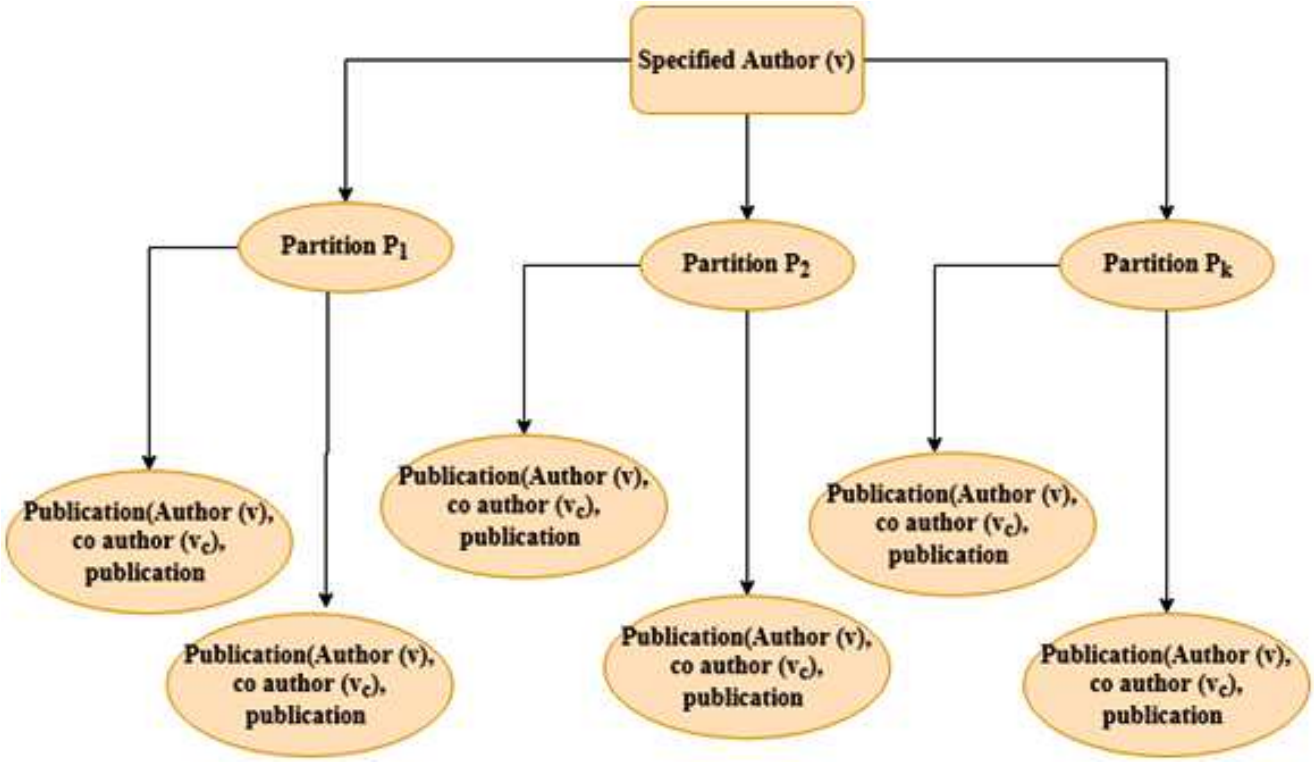

Figure 3.14: Specified Author Representation in Each Partition

3.3.2 DBLP Processing and Visualization

DBLP processing comprises of major blocks as - retrieve publication details of author, co-author list generation based on publication, computation of author performance measure and visualization of authors publication as shown in figure 3.12.

Publication details of specified authors are retrieved and performance of author is measured in terms of parameters such as stability, cooperativeness, consistency and contribution factor. The significance of each parameter is explained in the section 3.3.2.4.

3.3.2.1 Retrieve Publication details of Author

Name of author (*selected _author*) of whose details are to be searched and performance to be measured) is provided as an input. The specified author name is searched in RDBMS

to obtain authors unique id and partitions in which he/she has published articles. If the specified author exists then, articles published by an author are retrieved from all partitions in which, he / she is active.

3.3.2.2 Author List Generation based on Publication

All publications of *selected _author* (the selected author is source node)are obtained from all partitions P_a, then all co- authors of *selected _author* are determined for all publications as shown in figure 3.15 where p_1, p_2 and p_3 indicates publications .

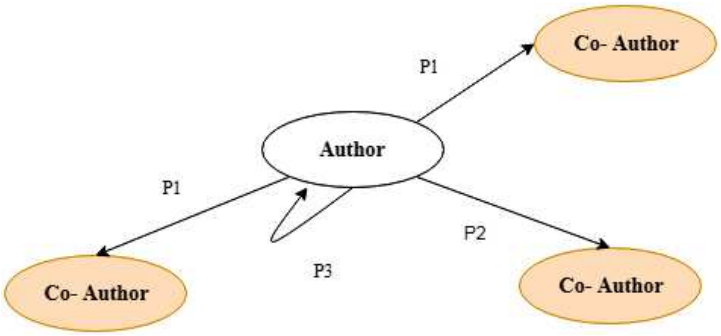

Figure 3.15: Author Publication Information Retrieval

3.3.2.3 Visualization of Authors Publication

For visualization of partitioned sub-graphs and publication details of *selected _author*, a vis Java script based network graph visualization library is used [119]. Authors publication details are transformed in a form required for vis library to visualize a network graph.

3.3.2.4 Computation of Author Performance measures

Depending upon authors information retrieved, the performance of an author is measured in accordance with various parameters as stability, cooperativeness, solidity, consistency, contribution factor and n number of most influential authors in a quantum. Forcoa.net system computed stability, cooperativeness, and solidity, rest of the parameters such as consistency, contribution factor and n number of most influential authors in a quantum are the contributions of research work. Computation details of each parameter are discussed below.

1. Stability

The details of publication published together by two authors are represented by an interaction between two vertices in a network. As two authors may have more than one publication together and this leads to multiple interactions between these two vertices. If the number of interactions between two vertices is more, then these vertices are considered as more stable and bond between them is stronger.

For each vertex and tie, two time changing characteristics are defined as [111]-

- Edge stability: Edge stability ES is time span for which tie between two vertices remain active since first interaction such that $ES > 0$.

- Vertex stability: Vertex stability VS is a time period for which vertex remains active since first publication such that $VS > 0$.

- Self- stability: Self- stability is self- loop (self- edge) which stores information about publication where no co- author is involved.

2.Cooperativeness

It mainly describes the relationship of vertex v with other vertices having interactions with it. As vertex stability is independent of the number of edges. Hence, in this case, important interactions are considered in which adjacent vertex has higher stability.

Cooperativeness for vertex v is computed as [111].

$$\text{cooperativeness(v)} = \Sigma_i \sqrt{ES(e_i.VS(v_i)} \tag{3.5}$$

Where v_i and e_i are vertices adjacent to vertex v

Algorithm 8 Computation of Cooperativeness

Procedure cooperativeness( )

1: **for** each adjacent vertex v_i of v **do**
2: Calculate vertex stability of v_i,Vs_{vi}
3: Calculate edge stability of e_i, ES_{ei} where $e_i = (v, v_i)$
4: Cooperativeness of (v) + = sqrt $(VS_{(vi)}.ES_{(ei)})$
5: end for ***endprocedure***

3.Solidity

The basic motivation is to select strong ties having at least one interaction in aspecific period. Solidity considers only ties having at least some minimal stability (stab). Here we have considered stab=1 month that is tie should have minimum one interaction in the period of a year.

$$\text{Solidity(v,stab)} = \Sigma_i(ES(e_i) - stab) \tag{3.6}$$

4.Consistency

Authors consistency measures variation in the number of an interaction of author v in the surrounding in each successive span of y years. It helps to determine whether an author is consistent or not in the selected period of time t.

Compute arithmetic mean by using the equation 3.11

$$Arithmetic\_mean = \overline{X} = \frac{\Sigma x}{n} \tag{3.7}$$

x= Total number of publications by author v in the span of y years

n=Total number of spans in specified time period t.

$$Standard\_deviation = \sigma = \sqrt{\frac{\Sigma x^2}{n} - \left(\frac{\Sigma x}{n}\right)^2} \tag{3.8}$$

$$Consistency\_of\_author = \frac{\sigma}{\overline{X}} \times 100 \tag{3.9}$$

5. Contribution Factor

The contribution factor is the measure of authors contribution for publication. If more than on author is involved in a publication, then it is essential to measure and distinguish the contribution of main author and his all co- authors. The value of contribution ranges in between 0-1.If the author is the only author with no co- author has assisted, then his contribution is maximum and it is assumed as 1.

If two authors are associated with a paper, then first authors contribution is assumed as 0.6 and the second authors contribution is assumed as 0.4.

If more than two authors are associated in a publication, then first authors contribution remains same as 0.6 and remaining authors contribution is equally divided to a value 0.4.

For particular publication, Contribution factor of an-author and is computed as,

$$Contribution\_Factor(CF) = \begin{cases} \text{CF}_1 = 1 & \text{if} N = 1 \\ \text{CF}_1 = 0.6, CF_{i-1} = 0.4/N - 1 & \text{if} \text{N} \geq 2 \end{cases} \tag{3.10}$$

Where N= Number of authors associated in a publication and i varies from 2 to N

The total contribution of author is computed by using equation 3.10

$$Contribution factor(C.F.) = \Sigma CF_j \tag{3.11}$$

where j is number of publications in which author is active.

6. Find most influential authors in a quantum

In analysis of bibliographic record it is required to find most influential authors in a specified period of time (quantum). Most influential authors are those who have more number of ties with other authors in same quantum. An Efficient Cut Set and Partitioning Algorithm(CPA) is applied on a graph $G(V, E)$ containing publication records for a specified quantum. After partitioning $G(V, E)$ into k number of partitions using CPA algorithm, k number of partitions successfully generated. In each resultant partition, seed vertex represents the most influential author and remaining vertices represent co-authors along with

their interaction (publication) details.

3.4 Novel Partitioning and Visualization Algorithms

This section discusses the workings of algorithms designed for efficient partitioning of large graphs into subgraphs, visualization of subgraphs and analysis of bibliographic records in DBLP dataset.

3.4.1 Algorithm for Partition Building and Edge Cut Computation

A novel algorithm An Efficient Partition Building and Edge Cut Computation is developed for efficient partitioning of any large size graph into k number of partitions. This algorithm accepts graph $G(V, E)$ which is to be partitioned and the number of partitions to be generated as output. This partitioning algorithm comprises of following major steps-

- Seed vertex computation
- Vertex-partition Weight Computation
- Partition building
- Number of edge- cuts computation

Initially, seed vertex for each partition is obtained using algorithm 6 discussed in section 3.1 and each partition grows around its seed vertex. After this vertex-partition weight is calculated in order to decide suitable partition in which vertex is to be placed using equation 3.3 and algorithm 7.
Based on vertex partition, weight is obtained and size of the partition, each vertex is placed in the most suitable partition. Algorithm 10 enlists operations performed during placement of each non-seed vertex in the most suitable partition. Based on partitions obtained, numbers of edge-cuts are computed with reference to algorithm 11.

Algorithm 9 Graph Partitioning

Input: $G(V, E)$: Graph, k: number of partitions to be generated
Output: Edge cuts, P: Sub graphs
Procedure GraphPartitioning ( )

S $\leftarrow$ SetDifferencebasedSeedVertexComputation()
Capacity of each partition p in P, $\gamma \leftarrow 0$
for i=0 to k-1 **do**
    $p[i] \leftarrow \{\phi\}$
end for
if $length(V)\%2 == 0$ **then**
    $\gamma \leftarrow length(V)/k = (\mid v \mid /n)$
else
    $\gamma \leftarrow (length(V)/k) + 1$
end if
for i=0 to k-1 **do**
    $\omega_{,pi} \leftarrow 1.1$
end for
$remaining_vertices$, X = Build_Partition(V)
for each x in X **do**
    $X = Build_Partition(X)$
end for

endprocedure

Algorithm 10 Partition Building

Input: $G(V, E)$: Graph, k: number of partitions to be generated
Output: Edge cuts, P: Sub graphs

```
1: procedure BuildPartition(G)
2:     remaining vertices, X = {φ}
3:     for each vertex, v ∈ V do
4:         if v ∈ partitionp such that p ∈ P then
5:             break
6:         else
7:             W_v = {φ}
8:             possible partitions for v, T_v ← {φ}
9:             allocated partition for v, k ← −1
10:            for each partition p ∈ P do
11:                distance of v to partition p, σ_{v,p} ← 1
12:                for each vertex, v_p ∈ p do
13:                    if v and v_p are adjacent vertices then
14:                        weight of v for partition p, ω_{v,p} ← 1.0/σ_{v,p}
15:                        W_v[p] ← ω_{v,p}
16:                    else
17:                        σ_{v,p} ← σ_{v,p} + 1
18:                    end if
19:                end for
20:            end for
21:            if length(W_v[p]) > 0 then
22:                sort W_v[p] in descending order
23:                for each partition p ∈ W_v[p] do
24:                    if sizeof(p) < γ then
25:                        k ← p
26:                        break
27:                    else
28:                        for each vertex v_x in P do
29:                            for each partition p_i in P do
30:                                if p== p_i then
31:                                    continue
32:                                else if p_i ∈ W_{vx[pi]} && ω_{vx,pi} > ω_{vx,p} then
33:                                    Add p_i to T_v
34:                                end if
35:                            end for
36:                        end for
37:                        p_x ← −1
```

```
38:                         if sizeof(T_v) > 0 then
39:                             sort T_v in descending order
40:                             for each p_i ∈ T_v do
41:                                 if sizeof(p_i) < γ then
42:                                     p_x ← p_i
43:                                     break
44:                                 end if
45:                             end for
46:                             if p_x ≠ −1 then
47:                                 move v_x to p_x
48:                                 k ← p
49:                                 break;
50:                             end if
51:                         end if
52:                     end if
53:                 end for
54:             end if
55:             if k ≠ −1 then
56:                 add v to partition k
57:                 W_v[k] ← ω_{v,k}
58:                 break
59:             else
60:                 add v to X
61:             end if
62:         end if
63:     end for
64: end procedure
```

Algorithm 11 Edge Cut Computation

INPUT: P: Partitions, $G(V, E)$: graph
OUTPUT: Edge-cuts: F, Set of edges, n
Procedure Edge-cutCoputation ( )

```
1: F ← {φ}
2: for each Partition p_i in P do
3:     for each Partition p_j in P do
4:         for each vertex v_i in p_i do
5:             for each vertex v_j in p_j
6:                 if v_i and v_j are adjacent then
7:   add edge e(v_i ,v_j) to F
8:                 end if
9:             end for
10:        end for
11:    end for
12: end for
endprocedure
```

3.4.2 Algorithm for Graph Visualization

An Efficient Graph visualization algorithm is developed to visualize partitioned sub-graphs generated as an output of partitioning. Graph $G(V, E)$ is visualized as a network where nodes represent vertices in a graph and connectivity between two vertices represent an edge. The detailed steps are discussed in algorithm 12.

Algorithm 12 Graph Visualization

INPUT: Graph $G(V, E)$

OUTPUT: Graph $G(V, E)$ visualized as a network of nodes and edges.

(L_V) :Nodes list and edge_array (L_E):Edge list represents edges between two vertices

$ProcedureGraphVisualization()$

1: Nodes list in G, $L_V \leftarrow \{\phi\}$
2: Edge list in G, $L_E \leftarrow \{\phi\}$
3: **for** each vertex v_i in V **do**
4: $\alpha \leftarrow id(v_i) : label(v_i)$
5: add α to L_V
6: **end for**
7: **for** each vertex v_i in V **do**
8: **for** each vertex v_j in graph V **do**
9: **if** v_i and v_j are adjacent vertices **then**
10: $\beta \leftarrow$ label(e(v_i, v_j))
11: **end if**
12: **end for**
13: **end for**
14: vis(α, β)
endprocedure

3.4.3 Algorithm for Retrieval of Author and Co-Author Publication Details

This algorithm accepts an *author_ name* as input whose publication details and performance to be measured. This author information retrieval algorithm comprises of the following major steps.

- DBLP Pre-processing
- DBLP Partitioning and Visualization

DBLP parsing algorithms perform partitioning on bibliographic records and generate the partition of articles published in the span of 5 years along with it generates author vertex having attributes as unique id and publication label using algorithm 13

Based on partition label generated, specified author is searched for finding its co-author and publication information in all partitions using algorithm 14.

Algorithm 13 Authors and Co-authors Information

INPUT: DBLP dataset in xml format

OUTPUT: Partitions of articles published in the span of 5 years, list of authors.

Q = set of quantums as $\{q_1, q_2, q_3....q_n\}$ where q_i is fixed span of specified successive years

v = main author for publication

v_c= co- author of v

e= details of tie- up between v and v_c

author _ list = $\{\phi\}$

Procedure AuthorCo-authorsInformation( )

1: **for** each publication record (node), r in DBLP dataset **do**
2: Extract all attributes of r
3: **for** each author, a in r**do**
4: Extract $author\_name$ for a
5: $a.author_{id}$ = unique integer number
6: **if** a not belongs to $author_{list}$ **then**
7: add a to $author_{list}$
8: Extract year_ of _ publication (y) for r
9: Determine partition label $l(p) = q_i$ such that $q_i \epsilon$ Q and q_i contains y
10: Add r to graph having label $l(p)$, in form of $(v, v_c = \{v_{c1}, v_{c2}..v_{cn}\}, e)$
11: **end if**
12: **end for**
13: **end for**
endprocedure

Algorithm 14 Extract Coauthors (selected _ author)

INPUT: $selected\_author, Q, author\_list$
OUTPUT: co-authors of $selected\_author$
G_a = set of quantum in which $selected\_author$ is active
$ProcedureGraphVisualization()$

```
1: if sizeof(G_a) > 0 then
2:     for each graph G_i in G_a do
3:       Set selected_{author.publications} = {φ}
4:       for each publication record r in G_i do
5:         for each author a in r do
6:           if a == selected_author then
7:             add r to selected_author.publications
8:             selected_author.publications[r].co − authors = {φ}
9:             if (a! = selected_author) then
10:              add a to selected_author.publications[r].coathors[]
11:            end if
12:          end if
13:        end for
14:      end for
15:    end for
16: end for

    endprocedure
```

3.4.4 Algorithm for Visualization of Publication Details and Interaction of Author and Co-Author

Vertex- edge graph in which central node represents selected author and remaining nodes represents co- authors of a selected author. The edge between author and co-author provides

details of publication published together by author and respective co- author with reference to algorithm 15.

Algorithm 15 Visualization of Author and Co -Author

INPUT: *selected_author* **OUTPUT:** Graph $G(V, E)$ visualized as a network of nodes and edges. Central node represents selected author and remaining nodes represents co-authors of selected _ author.
L_V:Nodes list author name as label
L_E: Edge list represents edge between two vertices(publication details)
$Procedure Visualization of author and coauthor()$

1: Node list in $G, L_V \leftarrow \{\phi\}$
2: Edge list in $G, L_E \leftarrow \{\phi\}$
3: **for** each publication, r in $selected\_authors.publications$ do
4: Apply steps 4 to 14 of algorithm 12
5: **end for**
endprocedure

3.4.5 Algorithm for Computation of Performance Measure Parameters of Author

After visualization of author and co-author, performance measure parameters of specified author by retrieving author-id and with reference to equation 3.4 to equation 3.9 and algorithm 16.

Algorithm 16 Author information retrieval

INPUT: Partitions, *author_ list*, author *(author_name)* whose details to be searched
OUTPUT: Details of articles published by specified author

ProcedureAuthorInformationRetrieval()

1: Accept *author_name* whose details to be searched (*selected_author*)
2: Retrieve the id (*author_id*) of *selected_author* from *authors_list*
3: set $selected\_author.publication_count = 0$
4: Extract_ Coauthors(*selected_author*)
5: **if** $selected\_author.publication_count > 0$
6: Calculate Cooperativeness
7: Calculate Solidity
8: Find yearly publication details
9: Calculate consistency
10: Calculate contribution factor
11: **endif**
12: Visualize publication details of selected author in form of vertex edge graph where each vertex represent author and edge represent publication details. *endprocedure*

In this chapter novel algorithms for graph partitioning and visualization are proposed. Author information retrieval algorithm, novel parameters for author performance measure and visualization algorithms for finding author and co-author interaction are also explained.

Chapter 4

RESULTS

Partition Building and Cut-set Computing Algorithm named as "Cut-set and Partitioning Algorithm (CPA)", sub-graph visualization algorithm and an author information retrieval algorithm have been designed, implemented and tested. Results of the system are presented in this chapter. Experimental setup for testing of the designed algorithm is discussed in next section.

4.1 Experimental Setup

Performance of "Cut-set and Partitioning Algorithm (CPA)", tested on standard graph datasets is listed in table 4.1 and on DBLP Bibliographic record data set.

Benchmark for Graph Partitioning

Chris Walshaws graph partitioning archive is predominantly used to test the performance of any graph partitioning algorithm. In Chris Walshaws graph partitioning archive, a wide range of graphs are provided from various domains including DBLP bibliographic record and VLSI designing [119]. Some of the instances listed in table 4.1 are used for experimental evaluation of our CPA algorithm.

Table 4.1: List of Bench Mark Graphs

Sr. No.	Datasets	Order	Description
1	BCSSTK28 (BC28)	4410	Solid element model
2	BCSSTK29 (BC29)	13992	3D Stiffness matrix
3	BCSSTK30 (BC30)	28294	3D Stiffness matrix
4	BCSSTK31 (BC31)	35588	3D Stiffness matrix
5	BCSSTK32 (BC32)	44609	3D Stiffness matrix
6	BCSSTK33 (BC33)	8738	3D Stiffness matrix
7	BRACK2 (BRCK)	62631	3D Finite element mesh
8	CANT (CANT)	54195	3D Stiffness matrix
9	COPTER2 (COPT)	55476	3D Finite element mesh
10	4ELT (4ELT)	15606	2D Finite element mesh
11	MEMPLUS (MEM)	17758	Memory circuit
12	ROTOR (ROTR)	99617	3D Finite element mesh
13	SHELL93 (SHEL)	181200	3D Stiffness matrix
14	TROLL (TROL)	213453	3D Stiffness matrix
15	WAVE (WAVE)	156317	3D Finite element mesh

4.1.1 Edge Cut Computation

The number of edge-cuts computation is a vital parameter to measure the performance of any partitioning algorithm in terms of accuracy. The developed CPA algorithm is tested for 32-way, 64-way and up to 128-way partitioning. Table 4.2 shows comparative results (number of edge-cuts) obtained after applying CPA.

Table 4.2: Number of Edge Cuts Computation

Data sets	Number of Vertices	Number of Edges	Number of Edge Cuts in 32 way partitioning						Percentage Reduction in Edge-cuts(Min. Edge-cut Value)
			Greedy Refinement	Kernighan Lin Refinement	Boundary Greedy Refinement	Boundary Kernighan Lin Refinement	Boundary Kernighan Lin Greedy Refinement	Novel (CPA)	
4ELT	15606	45878	1834	1833	2028	1894	1894	**1305**	**28.81**
BCSSTK31	35588	579147	45267	46852	46251	45047	45991	**26174**	**41.90**
BRACK2	62631	366559	22451	20720	29786	19785	21152	**13219**	**33.19**
ROTOR	99617	662431	38241	38312	36834	36498	36512	**32784**	**10.18**

Figure 4.1 shows the comparative study of results obtained for datasets listed in table 4.1. It reveals that the number of edge-cuts obtained for CPA is lesser as compared to other graph partitioning algorithms like Greedy refinement, Kernighan-Lin Refinement, Boundary Greedy Refinement, Boundary Kernighan-Lin Refinement and Boundary Kernighan-Lin Greedy Refinement computed by researchers using METIS in [4]. It helps to conclude that CPA gives better results over earlier techniques.

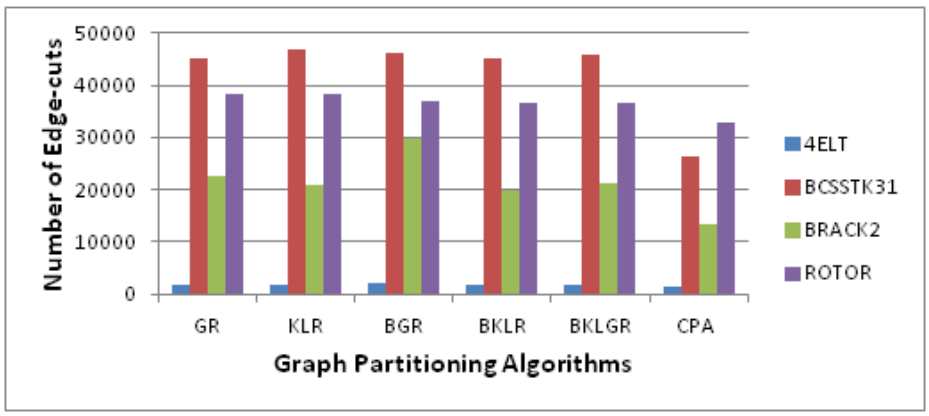

Figure 4.1: Comparative Analysis of Edge Cuts Computation

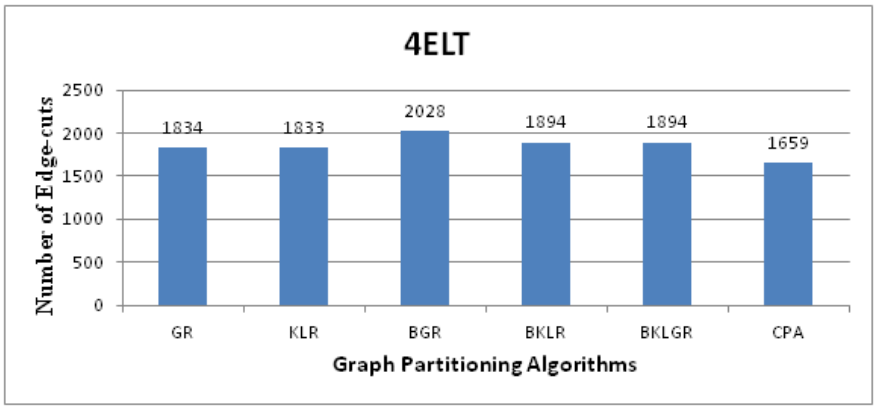

Figure 4.2: Edge cuts Computation for 4ELT dataset

4ELT is the dataset of 2-D finite element mesh. Figure 4.2 shows the number of edge-cuts obtained for the 4ELT dataset in 32-way partitioning. It is noticed that CPA algorithm gives the lesser number of edge- cuts as compared to existing graph partitioning algorithms.

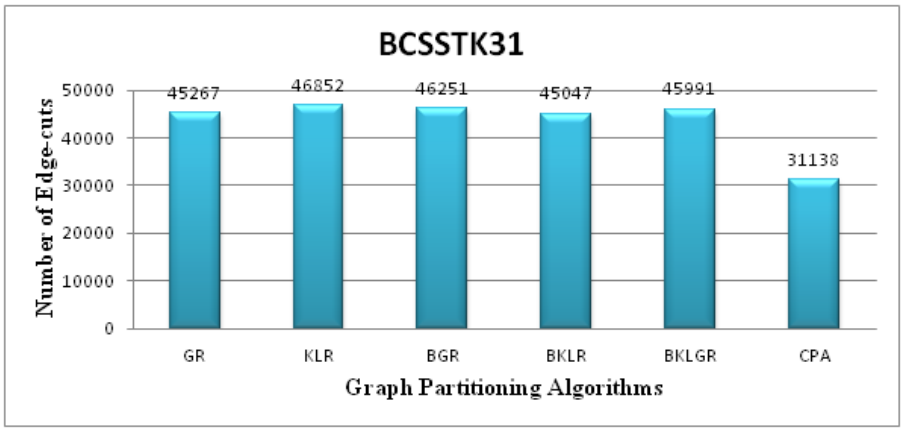

Figure 4.3: Edge cuts Computation for BCSSTK31 dataset

BCSSTK31 is the dataset of 3-D stiffness matrix. Figure 4.3 shows the number of edge-cuts obtained for the BCSSTK31 dataset in 32-way partitioning. It is noticed that CPA algorithm gives the lesser number of edge-cuts as compared to existing graph partitioning algorithms.

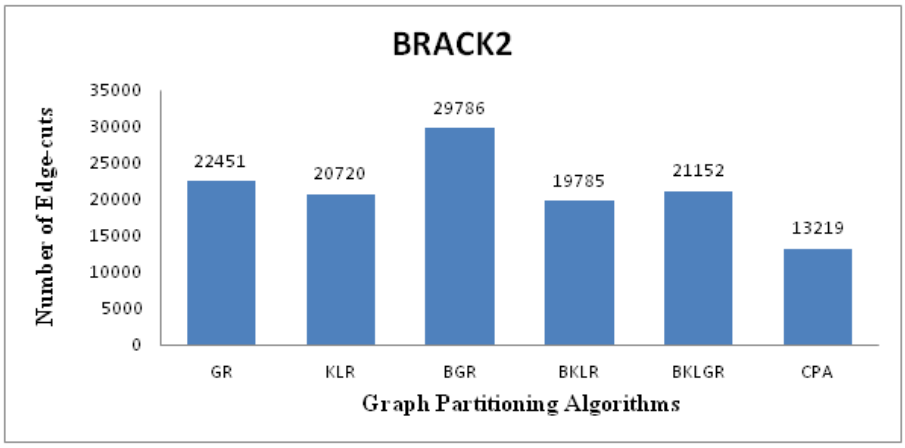

Figure 4.4: Edge cuts Computation for BRACK2 dataset

BRACK2 is the dataset of 3-D finite element mesh. Figure 4.4) shows the number of edge-cuts obtained for the BRACK2 dataset in 32-way partitioning. It is noticed that CPA algorithm gives the lesser number of edge- cuts as compared to existing graph partitioning algorithms.

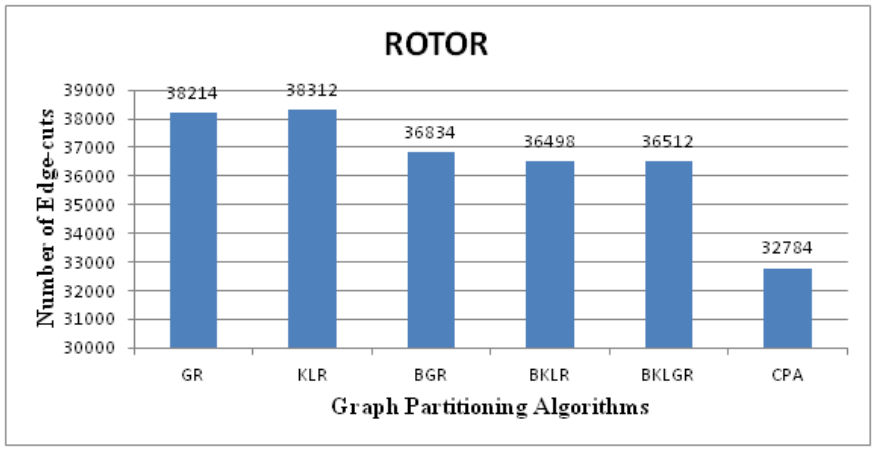

Figure 4.5: Edge cuts Computation for ROTOR dataset

ROTOR is the dataset of 3-D finite element mesh. Figure 4.5 shows the number of edge-cuts obtained for the ROTOT dataset in 32-way partitioning. It is noticed that CPA algorithm gives the lesser number of edge- cuts as compared to existing graph partitioning algorithms.

Table 4.3 shows computation of edge-cuts for 64- way partitioning by using CPA algorithm for datasets listed in table 4.3. Comparison values for edge-cuts in 64-way

Table 4.3: Edge-cuts computation with 64-way Partitioning

Datasets	Number of Vertices	Number of Edges	Number of edge cuts in 64way partitioning by Novel CPA
4ELT	15606	45878	2057
BCSSTK31	35588	579147	48866
BRACK2	62631	366559	27219
ROTOR	99617	662431	79235

partitioning are not available. Therefore comparative analysis is not shown for 64-way partitioning.

Table 4.4: Percentage Reduction in Edgecuts

Datasets/Partitioning Algorithms	GR	KLR	BGR	BKLR	BKLGR
4ELT	28.84	28.81	35.65	31.10	31.10
BCSSTK31	42.18	44.13	43.41	41.90	43.09
BRACK2	41.12	36.20	55.62	33.19	37.50
ROTOR	14.21	14.43	11.00	10.18	10.21

Table 4.4 shows percentage reduction in edge-cuts computation for datasets listed with respect to novel CPA algorithm.

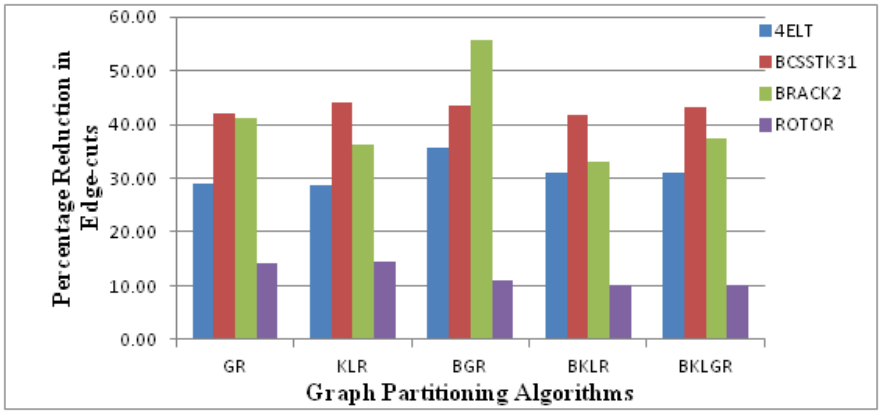

Figure 4.6: Percentage Reduction in edgecuts

Quality of CPA algorithm is compared with greedy refinement, Kernighan Lin Refinement, boundary greedy refinement, boundary Kernighan-Lin refinement and boundary

Kerrnighan-Lin greedy refinement. For each dataset listed in table 4.4, the ratio of edge-cuts obtained for CPA and to all listed algorithms is plotted for 32-way partitions. Figure 4.6 reveals that percentage reduction in edge-cuts varies approximately in the range of 10 to 50.

4.1.2 Visualization of Partitioned graphs

Graph visualization is the visual representation of the nodes and edges of a graph. Visualization tools is an essential layer to identify and analyse the insights from connected data. Graph visualization is useful because of many reasons like

1. Less Time to Assimilate Information: As the human brain processes visual information much faster than writing one, Visual data always ensures better understanding and reduces the time to act.

2. A Better Understanding of Problem: One can achieve better understanding of a problem by visualizing pattern and contests. Graph visualization not only visualizes relationships but also assists to understand the contest of data.

3. An Effective Form of Communication: Visual representation is more effective medium to share the finding and offer more instinctive way to understand the data.

4. Easy To Use: Any user without special technical and programming skills can interact with graphs visualization.

For better representation and illustration of partitions (subgraphs) generated after partitioning graph G, graph visualization tool was required. Existing tools for graph visualization are application specific and they required input data in specific forms only. To overcome this limitation, we developed our own visualization tool using vis java script library [120]. This visualization tool can visualize any graph *G (V, E)*, where V= $\{v_1,v_2,v_3..\}$ and E=e| e= (v_i,v_j,l_{ij}). Where v_i,v_j and l_{ij} are source vertex, destination vertex and label of edge e respectively. It also concurrently displays visualization for each partition (subgraph) generated after partitioning of graph G. Figure 4.4 shows a visualization of 4ELT graph having 15606 vertices and 45878 edges, which is partitioned in 32 partitions and figure 4.5(1) to figure 4.5(32) shows each subgraph visualization.

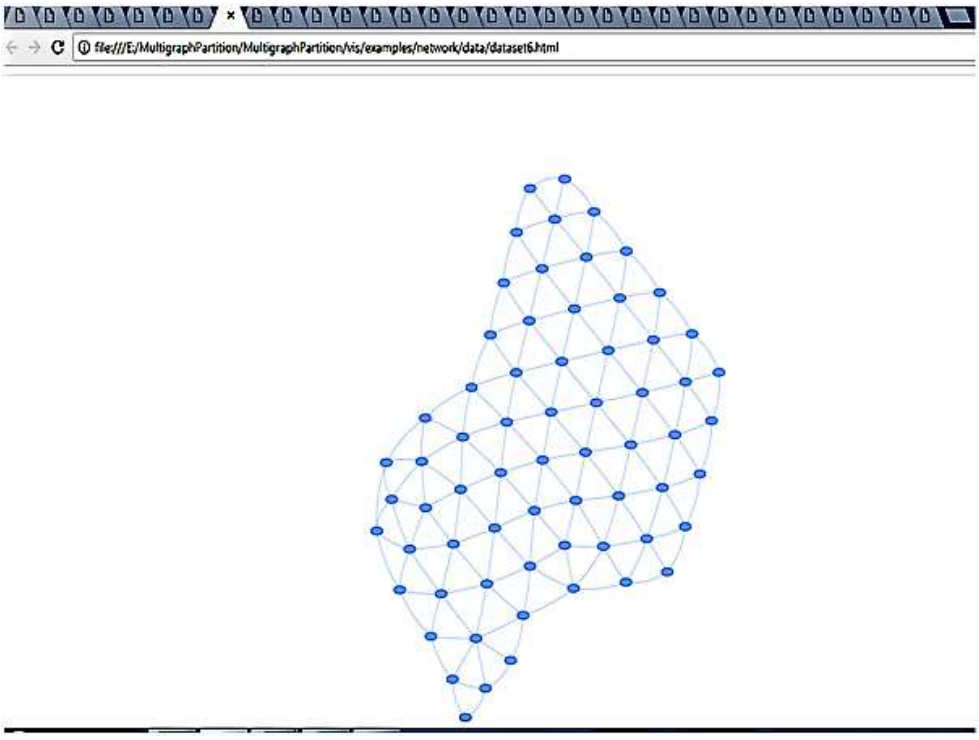

Figure 4.7: Graph Visualization of 4ELT Graph after Partitioning in 32 Partitions

CPA algorithm is applied on 4ELT graph containing15606 vertices and 45878 edges. Graph is partitioned into 32 partitions. Figure 4.7 shows visualization of 32 subgraphs concurrently in a separate tab of the same browser. Figure 4.8(1) to figure 4.8(32) show an independent visualization of 32 subgraphs.

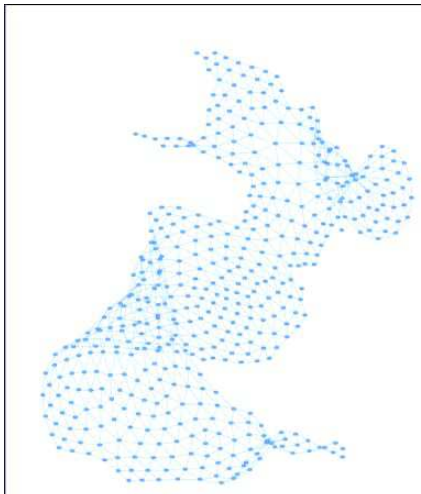

Figure 4.8(1): Subgraph 1

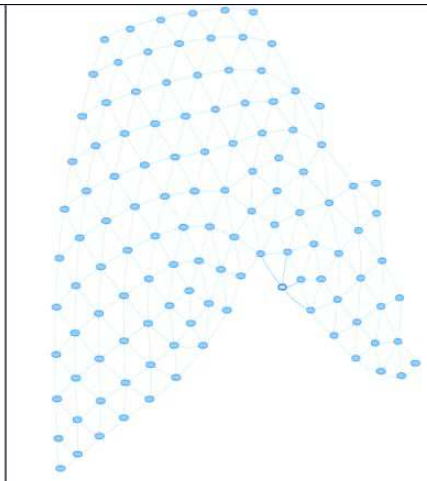

Figure 4.8(2): Subgraph 2

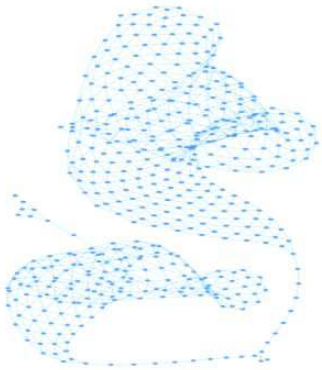

Figure 4.8(3): Subgraph 3

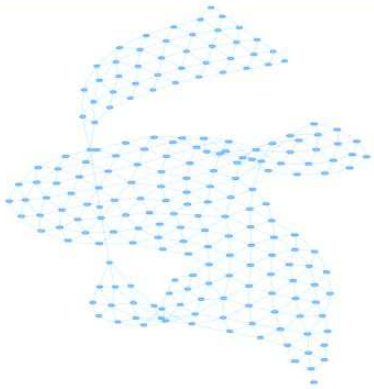

Figure 4.8(4): Subgraph 4

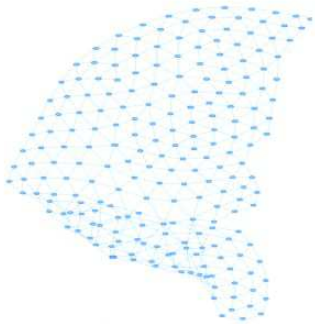

Figure 4.8(5): Subgraph 5

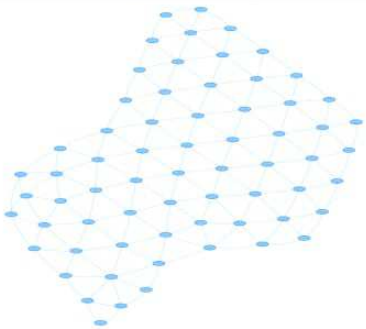

Figure 4.8(6): Subgraph 6

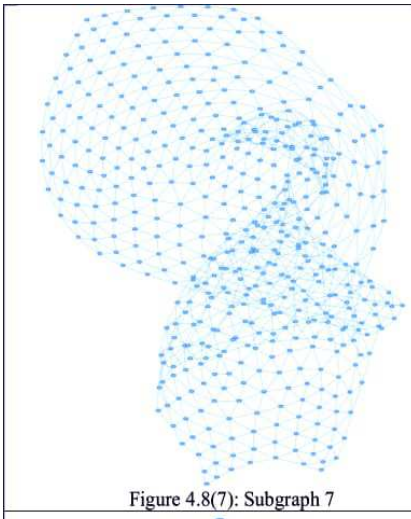

Figure 4.8(7): Subgraph 7

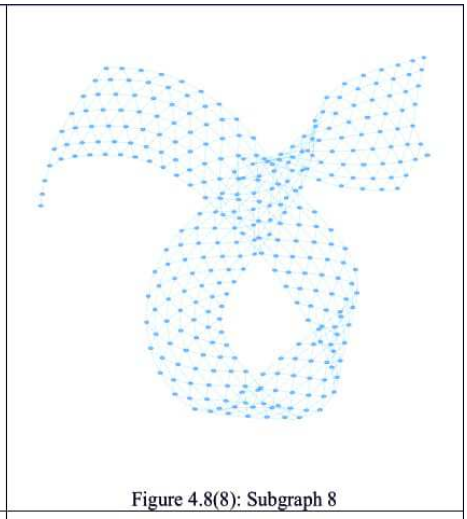

Figure 4.8(8): Subgraph 8

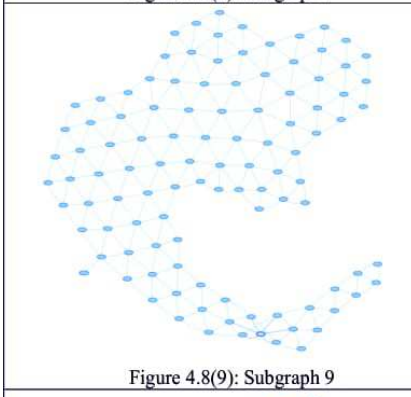

Figure 4.8(9): Subgraph 9

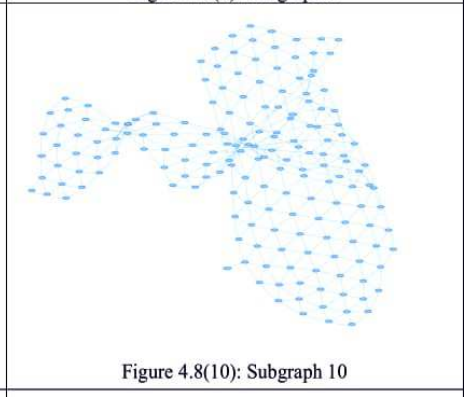

Figure 4.8(10): Subgraph 10

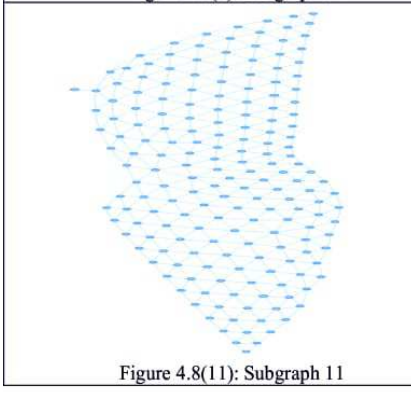

Figure 4.8(11): Subgraph 11

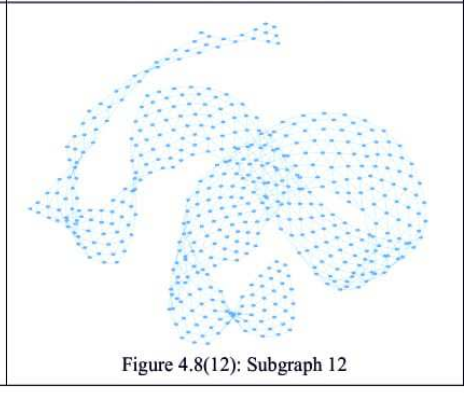

Figure 4.8(12): Subgraph 12

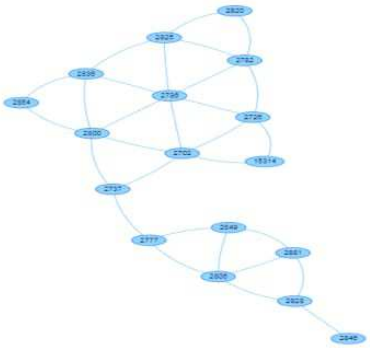

Figure 4.8(13): Subgraph 13

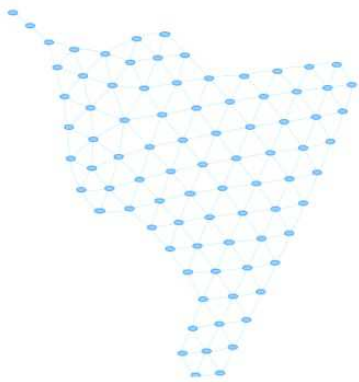

Figure 4.8(14): Subgraph 14

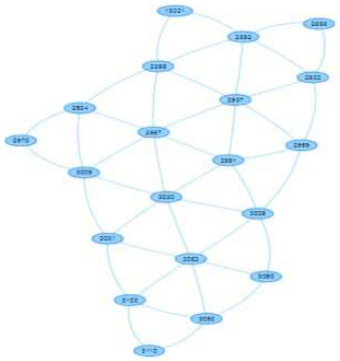

Figure 4.8(15): Subgraph 15

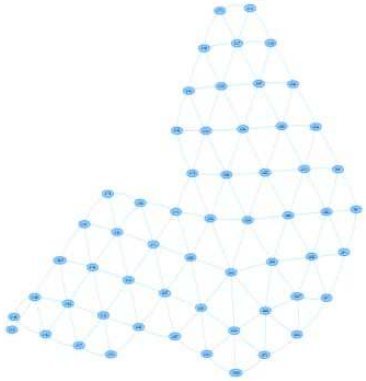

Figure 4.8(16): Subgraph 16

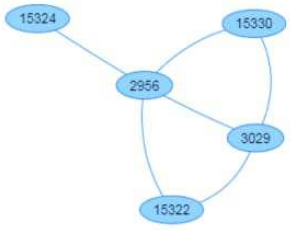

Figure 4.8(17): Subgraph 17

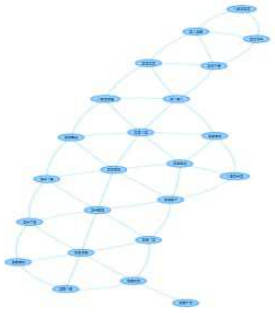

Figure 4.8(18): Subgraph 18

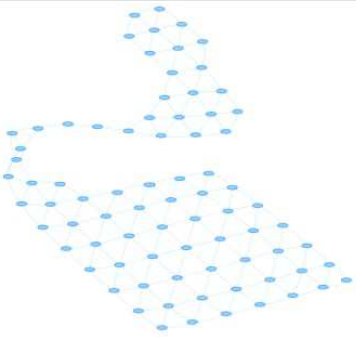

Figure 4.8(19): Subgraph 19

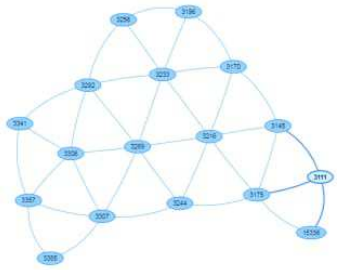

Figure 4.8(20): Subgraph 20

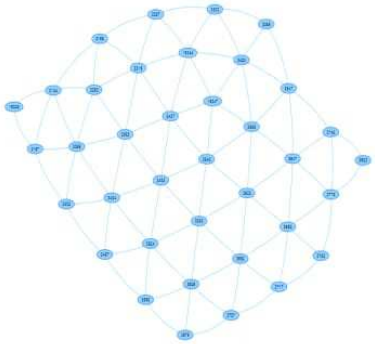

Figure 4.8(21): Subgraph 21

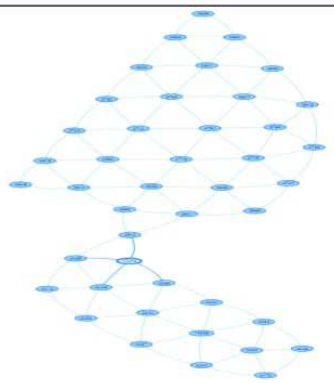

Figure 4.8(22): Subgraph 22

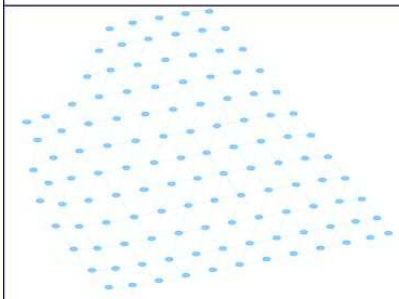

Figure 4.8(23): Subgraph 23

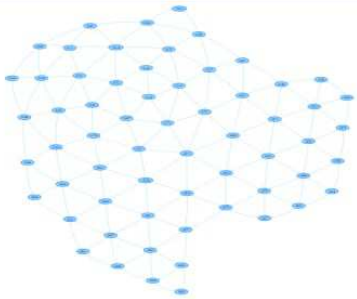

Figure 4.8(24): Subgraph 24

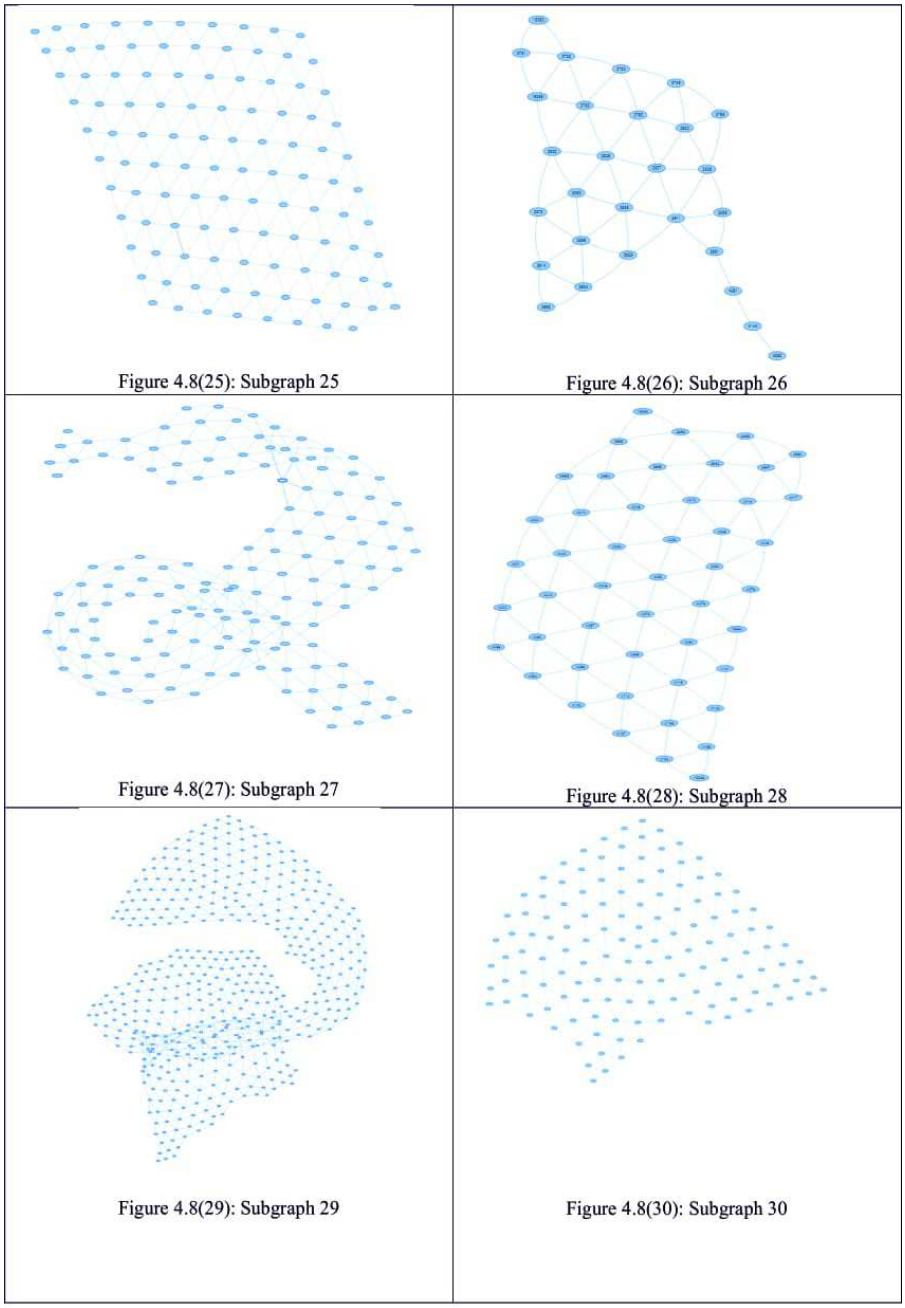

Figure 4.8(25): Subgraph 25

Figure 4.8(26): Subgraph 26

Figure 4.8(27): Subgraph 27

Figure 4.8(28): Subgraph 28

Figure 4.8(29): Subgraph 29

Figure 4.8(30): Subgraph 30

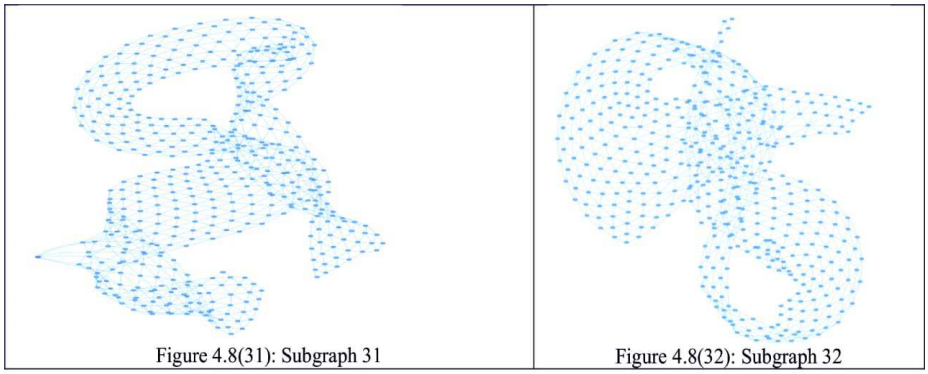

Figure 4.8: Visualization of 4ELT Subgraphs in 32 partitions

4.2 DBLP Bibliographic Record

DBLP dataset is a collection of bibliographic records in which each record represents details of publications as discussed in earlier chapter. While analysing the bibliographic records, it is required to find the details of specific author and ties he/she having with other authors. The results obtained for each operation in a DBLP processing system are below-

4.2.1 Partitioning of Publication Records

The DBLP dataset used contains 9, 85,177 bibliographic records from the year 1936 to 2017. Whole DBLP dataset was partitioned in the span of successive 5 years (quantum) and records in each quantum are stored in form of a graph. Table 4.5 shows quantum and number of publication records in each quantum obtained after splitting of DBLP dataset.

Table 4.5: Quantums And Publication Records in Each Quantum

Sr. No.	Quantums	Publication Records in Each quantum
1.	1936-1940	65
2.	1941-1945	48
3.	1946-1950	100
4.	1951-1955	450
5.	1956-1960	1399
6.	1961-1965	3620
7.	1966-1970	5860
8.	1971-1975	9493
9.	1976-1980	11717
10.	1981-1985	17614
11.	1986-1990	30880
12.	1991-1995	57694
13.	1996-2000	93847
14.	2001-2005	154538
15.	2006-2010	118743
16.	2011-2015	276464
17.	2016-2017	202645

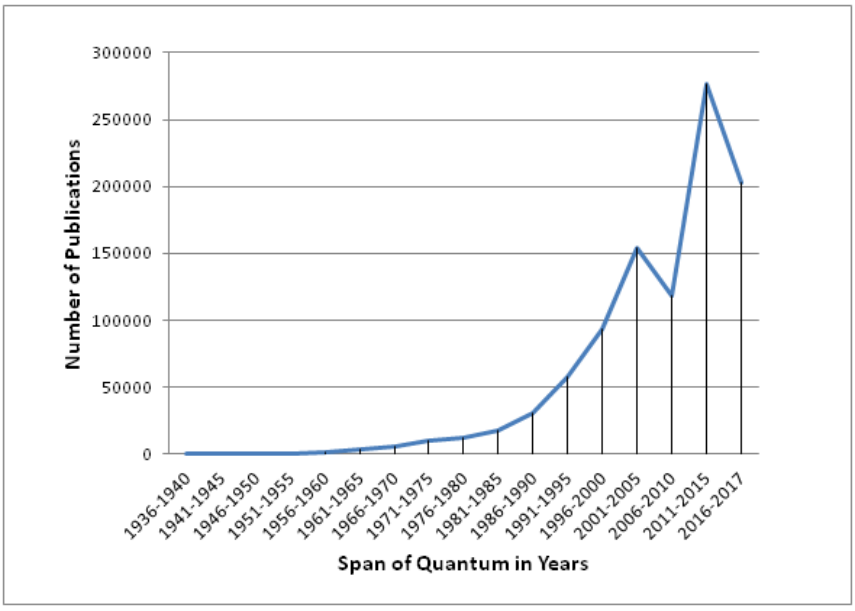

Figure 4.9: Record of Number of Publications From 1936 To 2017

From figure 4.9 it is observed that rapid development has been seen in publications by authors in the span of 1936 to 2017.

Table 4.6: Time Computation for Graph Partitioning for DBLP dataset

Sr. No.	Quantums	Number of vertices	Number of Edges	Time Required for Partitioning in Seconds
1.	**1936-1940**	**39**	**68**	**1.10**
2.	**1941-1945**	**28**	**51**	**1.89**
3.	**1946-1950**	**72**	**112**	**1.22**
4.	**1951-1955**	**419**	**551**	**0.31**
5.	**1956-1960**	**1334**	**1890**	**1.15**
6.	**1961-1965**	**3185**	**4882**	**2.24**
7.	**1966-1970**	**4963**	**7960**	**2.8**
8.	**1971-1975**	**8224**	**13818**	**5.97**
9.	**1976-1980**	**11257**	**18477**	**7.90**
10.	**1981-1985**	**17872**	**29827**	**12.97**
11.	1986-1990	33452	56392	67.60
12.	1991-1995	64687	118680	96.61
13.	1996-2000	108391	208131	1349.37

Table 4.6 shows quantums of DBLP dataset along with number of vertices and edges respectively. CPA algorithm is applied on each quantum to generate 5 partitions and the time required for partitioning is measured. It is observed that for a graph having approximately 30,000 edges, partitioning time required to generate subgraphs lies in seconds which is a noticeable smaller value.

Table 4.7: Time computation for Graph Partitioning for Benchmark dataset

Sr. No.	Dataset	Number of vertices	Number of Edges	Time Required for Partitioning in Seconds
1.	4ELT	15606	45878	9.47
2.	BCSSTK31	35588	579147	150.44
3.	BRACK2	62631	366559	131.11
4.	ROTOR	99617	662431	277.52

Table 4.7 shows benchmark dataset and time required for 32-way partitioning in minutes. From table 4.6 and 4.7, it is observed that for a graph having edges in the range of 30,000 to 45,000, partitioning time required to generate subgraphs lies in minutes and edges more than approximately 45,000 to 6,50,000 , time required for partitioning lies up to 5 hours

4.2.2 Author List generation

All unique authors are identified and a unique id is assigned to each author which is used to represent the author as a vertex while storing a record in a graph. Total 10, 37,449 numbers of authors were obtained in chosen DBLP dataset. Based on Author id, author name and span of quantums in which author has published a paper, his/ her publication details are hunt out. The span in which authors published a paper is called as active span. Table 4.8 shows authors and their active spans.

Table 4.8: List of Authors and Their Active Spans

Sr. No.	Author name	Author id	Active Span
1.	Raghu Ramakrishnan	20081	1991-1995, 1996-2000, 2006-10
2.	Jeffer L. Hiest	115342	1991-1995, 1996-2000, 2011-15
3.	Umeshwar Dayal	5955	1981-1985, 1986-1990, 1991-1995, 1996-2000, 2001-2005, 2006-10, 2011-2015
4.	H. V. Jagdish	19925	1986-1990, 1991-1995, 1996-2000, 2001-2005, 2006-10, 2011-2015, 2016-2017
5.	Briam Curelss	81509	1996-2000, 2001-2005, 2006-10, 2011-2015,2016-2017
6.	Varsha H. Patil	487724	2006-10
7.	Sudipta Mukhopadhyay	413180	1996-2000, 2011-2015,2016-2017
8.	A. Ben Hamza	29277	2001-2005, 2006-2010, 2011-2015,2016-2017
9.	A.Benjamin Premkumar	95929	1996-2000, 2001-2005,2006-2010
10.	Brian A. Davey	53233	1991-95, 1996-2000, C2001-05, 2006-10, 2011-15, 2016-20
11.	Brian A. Wichmann	43621	1971-75, 1976-80, 1981-85, 1986-90, 1991-95, 1996-2000, 2006-10
12.	Brian Alspach	178604	1976-80 , 1981-85, 1986-90, 1991-95, 1996-2000, 2001-05, 2006-10,2011-15, 2016-20
13.	Eva K. Lee	244565	2001-05, 2006-10, 2011-15, 2016-20
14.	Junshan Zhang	57214	2006-10, 2001-05, 1996-2000

15.	Marcus Brazil	89931	1991-95, 1996-2000, 2006-10, 2011-15, 2016-20
16.	Sarath Gopi 15	5595	2006-10
17.	Sarbari Gupta	995119	1991-95,1996-2000
18.	Sargur N. Srihari	81756	1976-80,1981-85,1986-90,1991-1996-2000,2006-10
19.	Sartaj Sahni	90031	1971-75, 1976-80, 1981-85, 1986-90, 1991- 95, 1996-2000, 2001-05
20.	Saru Kumari	12843	2011-15,2016-20
21.	Sarunas Paulikas	994400	2001-05, 2006-10
22.	Sarvesh H. Kulkarni	158477	C2006-10:C2001-05
23.	Lihua Liu	231581	2001-05, 2006-10,C2011-15 2016-17
24.	Paolo Rocchi	74371	1996-2000, 2001-05, 2006-10, 2011-15, 2016-20
25.	Jingguo Ge	134150	2016-20,2011-15

4.2.3 Retrieve Publication Details of Author

For a particular author, publication record is retrieved. A tie between author and co-author for particular publications is represented in the form of vertices and edges.

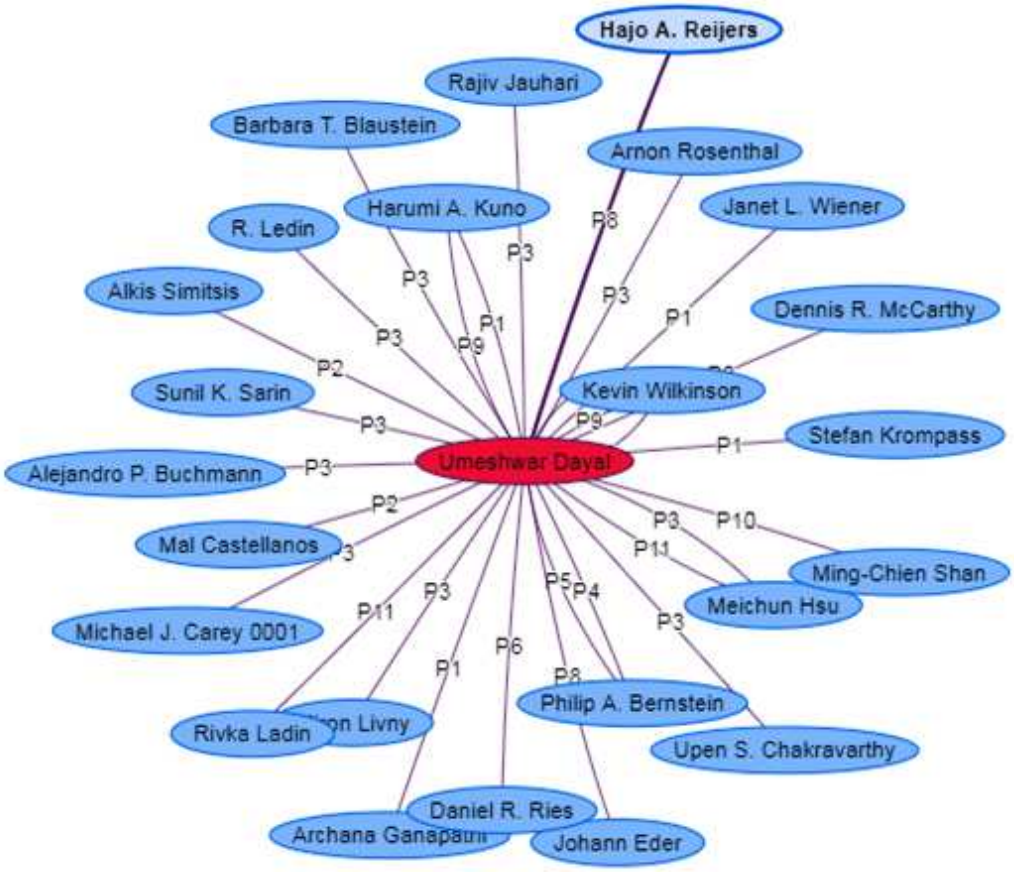

Figure 4.10: Publication Record of Selected Author

Figure 4.10 shows publications record of author Umeshwar Dayal represented by the vertex in red color and his co-authors are in blue color. Edges between red color vertex and blue vertices represent interaction (publication details) between author and co- authors as p_i, where p_i is ith publication of author. Interactions between co-authors Stefan Krompass, Archana Ganapathi, Janet L. Wiener and Harumi A. Kuno of Umeshwar Dayal having label p_1 i.e. article published in an association with them. Self loop represent there is no co-author for author for publication p_i.

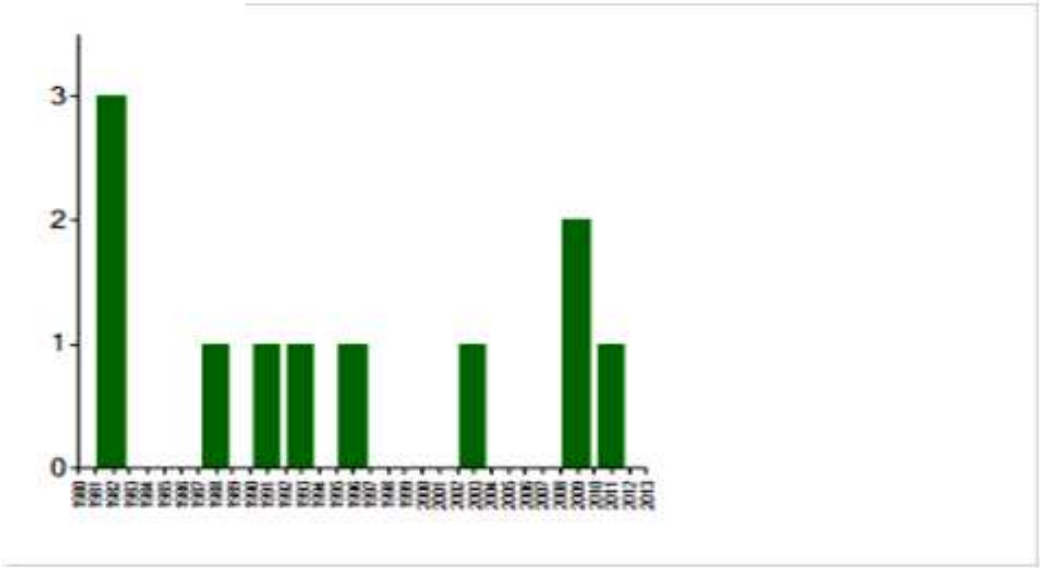

Figure 4.11: Publication Record of Selected Author

Figure 4.11 shows year wise publication graph of author Umeshwar Dayal. It reveals that his first publication was in the year 1981 with total 3 publications in the same year and last publication is in year 2012. Publication records retrieved are listed below.

PUBLICATIONS :

1. 2010-02-01:journals/sigops/DayalKWWGK09:Managing operational business intelligence workloads.:92-98:2009:43:Operating Systems Review:1:http://doi.acm.org/10.1145/1496909.1496927:db/journals/sigops/sigops43.html♯DayalKWWGK0
   COAUTHORS
   1. Harumi A. Kuno
   2. Janet L. Wiener
   3. Kevin Wilkinson
   4. Archana Ganapathi
   5. Stefan Krompass
2. 2009-12-08:journals/debu/DayalWSC09:Business Processes Meet Operational Business Intelligence.:35-41:2009:32:IEEE Data Eng.

Bull.:3:http://sites.computer.org/debull/A09sept/umesh.pdf:db/journals/debu/debu32.html♯DayalWSC0
COAUTHORS

1. Kevin Wilkinson
2. Alkis Simitsis
3. Mal Castellanos

3. 2017-11-06:journals/sigmod/DayalBBCHLMRSCLJ88:http://doi.acm.org/10.1145/44203.44208:The HiPAC Project: Combining Active Databases and Timing Constraints.:51-70:sigmodR/17-1/P051.pdf:1988:17:SIGMOD Record:1:db/journals/sigmod/sigmod17.html♯DayalBBCHLMRSCLJ8
COAUTHORS

1. Barbara T. Blaustein
2. Alejandro P. Buchmann
3. Upen S. Chakravarthy
4. Meichun Hsu
5. R. Ledin
6. Dennis R. McCarthy
7. Arnon Rosenthal
8. Sunil K. Sarin
9. Michael J. Carey 0001
10. Miron Livny
11. Rajiv Jauhari

4. 017-03-30:journals/tods/DayalB82:On the Correct Translation of Update Operations on Relational Views.:381-416:TODS7/P381.PDF:1982:7:ACM Trans. Database Syst.:3:db/journals/tods/tods7.html♯DayalB82:http://doi.acm.org/10.1145/319732.319740:journals/is/TsichritzisK78:conf/ifip/Armstrong74:journals/tods/BeeriB79:conf/vldb/BeeriBG78:journals/tods/BancilhonS81:...:...:...:conf/jcdkb/

Clemons78:persons/Codd72:persons/Codd71a:conf/ifip/Codd74:...:journals/tods/Codd79:books/aw/Date81:...:conf/vldb/DayalB78:journals/is/DayalB82:...:conf/vldb/EswaranC75:...:journals/tods/Fagin81:journals/is/FurtadoSS79:journals/cacm/Guttag77:conf/vldb/HammerM75:...:...:...:...:conf/sigmod/PaoliniP77:journals/is/PelagattiPB78:conf/sigmod/RoweS79:...:conf/sigmod/Stonebraker75:..
COAUTHORS

1. Philip A. Bernstein

5. 2017-05-20:journals/is/DayalB82:On the updatability of network views-extending relational
view theory to the network model.:29-46:1982:7:Inf.
Syst.:1:db/journals/is/is7.html♯DayalB82:https://doi.org/10.1016/0306-4379(82)90004-
COAUTHORS

1. Philip A. Bernstein

6. 2007-12-17:journals/debu/DayalR82:Research on uery Optimization at Computer Corporation of America.:33-37:1982:http://sites.computer.org/debull/82SEP-CD.pdf:5:IEEE Database Eng.
Bull.:3:db/journals/debu/debu5.html♯DayalR8
COAUTHORS

1. Daniel R. Ries

7. 2011-12-05:journals/csur/Dayal96:Database Technology At A Crossroads.:78:1996:28:ACM Comput.
Surv.:4es:db/journals/csur/csur28.html♯Dayal96:CSURs1/csur28/A78.pdf:http://doi.acm.org/10.1145/242224.24232

8. 2017-05-20:journals/dke/DayalER11:Guest editorial: Business process management.:407-408:2011:70:Data Knowl.
Eng.:5:https://doi.org/10.1016/j.datak.2011.02.001:db/journals/dke/dke70.html♯DayalER1
COAUTHORS

1. Johann Eder
2. Hajo A. Reijers

9. 2011-09-16:journals/debu/DayalKW03:Letter from the Special Issue Editors.:4:2003:26
   :IEEE Data Eng.
   Bull.:4:http://sites.computer.org/debull/A03dec/letter.ps:db/journals/debu/debu26.html♯DayalKW0
   COAUTHORS

   1. Harumi A. Kuno
   2. Kevin Wilkinson

10. 2012-09-17:journals/debu/DayalS93:Issues in Operation Flow Management for Long-Running
    Acivities.:41-44:1993:16:IEEE Data Eng. Bull.:2:DEBU/93JUN-CD.PDF:db/journals/debu/debu16.html♯DayalS93:http://sites.computer.org/debull/93JUN-

    CD.pdf:conf/ride/AhmedADKLS93:conf/sigmod/DayalGHKS93:conf/vldb/DayalHL91:books/mk/Elmagarmid92:books/mk/GrayR9
    COAUTHORS

    1. Ming-Chien Shan

11. 2007-12-19:journals/debu/DayalHL91:A generalized Transaction Model for Long-Running
    Activities and Active Databases.:4-8:1991:http://sites.computer.org/debull/91MAR-CD.pdf:14:
    IEEE Data Eng.
    Bull.:1:db/journals/debu/debu14.html♯DayalHL9
    COAUTHORS

    1. Meichun Hsu
    2. Rivka Ladin

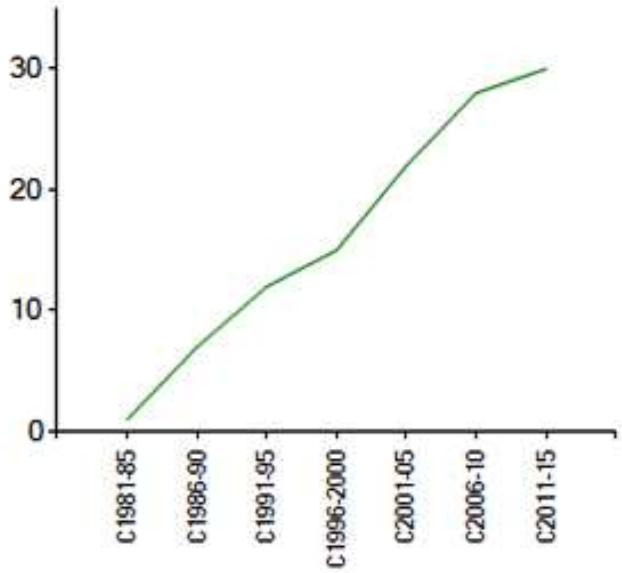

Figure 4.12: Stability Graph of Umeshwar Dayal

Figure 4.12 shows that selected authors stability is highest in quantum of 2011-15. Performance of an author is measured with parameters like consistency, stability, cooperativeness, solidity and contribution factor. Table 4.8 shows these parameter values obtained for 25 sample authors.

Table 4.9: Performance Measures of Authors

Sr. No.	Author Name	Number of Co Authors	Total Number of Publications	Performance Measure Parameters			
				Consistency in %	Cooperativeness	Solidity	Contribution factor
1.	Raghu Ramakrishnan	10	12	35	26.19	13	9.2
2.	Victor Khomenko	8	9	47	15	4	5.2
3.	H. V. Jagdish	32	29	69	71.81	8	22.4
4.	**Umeshwar Dayal**	**24**	**11**	**46**	**43.19**	**3**	**5.8**
5.	Jiawei Han	30	24	58	58.56	5	17
6.	Jeffery D. Connor	10	01	0	10.15	0	0
7.	Mac Schwager	8	04	35	10.35	04	2.4
8.	Sarvesh S. Kulkarni	3	2	0	1.73	0	1.2
9.	Sartaj Sahni	7	18	77	14.15	1	14.8
10.	Mi Ray Oham	2	2	0	1.41	2	1.2
11.	Sagar Venkatesh Gubbi	3	3	0	3	0	1.8

12.	Surajit Chaudhuri	25	33	94	59.31	32	24.2
13.	Immanuel Manohar	2	1	0	2.73	0	0.6
14.	Divesh Srivastava	3	6	33	7.89	0	5.6
15.	Hongjun Lu	11	9	73	38.18	5	5.4
16.	Hector Garcia-Molina	25	19	62	52.99	5	12
17.	Mark A. Abramson	9	7	53	15.6	4	4.5
18.	Brian A. Wichmann	4	13	45	5.7	0	11.4
19.	Junshan Zhang	12	10	79	35.21	3	6
20.	Eva K. Lee	38	18	46	15.15	14	8.2
21.	Saru Kumari	16	12	33	4	0	2.2
22.	Sarbari Gupta	2	3	33	4	0	2.2
23.	Jefferey W. Wilson	1	1	0	4.24	0	0.6
24.	Paolo Amato	7	2	0	13.5	0	0.6
25.	Sarath Gopi	5	2	0	6.24	3	1.2

From table 4.9, it is observed that an author Surajit Chaudhuri is more consistent with the high contribution factor. Cooperativeness of H. V. Jagdish is highest, it indicates that his co- authors are very much active for publications.

In the next level, Meichun Hsu is selected for publication information retrieval. His all co-authors publication details are retrieved and performance measure values are computed. Table 4.10 shows the performance measure of all his c-authors.

Further co-authors of Umeshwar Dayal are selected one by one, their publication details

are retrieved and performance measure values are computed.

Table 4.10: Performance Measure of Co-Authors of Umeshwar Dayal

Sr. No.	Author Name	Number of Co Authors	Total Number of Publications	Performance Measure Parameters			
				Consistency in %	Cooperativeness	Solidity	Contribution factor
1.	Michael J. Carey	16	21	74	26.29	1	15.8
2.	Stefan Krompass	6	1	0	9.56	0	0
3.	Philip A. Bernstein	63	43	72	82.65	16	28
4.	Meichun Hsu	8	8	47	9.05	1	5.2
5.	Sunil K. Sarin	2	2	0	1	0	1.6
6.	Rivka Ladin	2	2	0	3.46	2	1.2
7.	R. Ledin	0	0	0	0	0	0
8.	Johann Eder	11	8	31	13.21	0	5.6
9.	Hajo A. Rejjers	14	12	47	23.58	2	8
10.	Upen S. Chakravarthy	2	2	0	10.69	1	1.2
11.	Mal Castellanos	13	5	6	27.86	3	3
12.	Dennis R. McCarthy	1	1	0	1.41	0	0.6

13.	Rajiv Jahuri	0	0	0	0	0	0
14.	Arnon Rosenthal	16	12	41	22.16	4	7.2
15.	Miron Livny	3	2	0	6.97	0	1.2
16.	Alejandro P.Buchmann	5	3	33	2	0	1.8
17.	Barbara T. Blaustein	4	1	0	5.73	0	0.6
18.	Daniel R. Ries	1 4	0	7.75	1	3.2	
19.	Archana Ganap-athi	0	0	0	0	0	0
20.	Janet L. Wiener	0	0	0	0	0	0
21.	Alkis Simitsis	6	5	6	0.6	18.33	3
22.	Ming-Chien Shan	0	0	0	0	0	0
23.	Kevin Wilkin-son	5	3	33	2.83	0	2.2
24.	Harumi a. Kuno	1	2	0	4	1	1.2

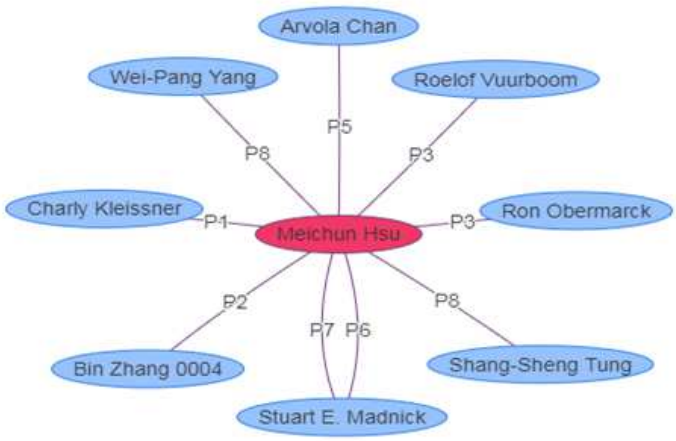

Figure 4.13: Publication Record of Selected Author: Meichun Hsu

Figure 4.13 shows publications record of an author Meichun Hsu represented by the vertex which is centrally placed and his co-authors are represented as adjacent vertices. Edges between central vertex and adjacent vertices represent interaction (publication details) between author and co- authors as p_i, where p_i is i^{th} publication of an author. Figure 4.13 reveals that that interactions between co-authors Roelof Vuurboom and Ron Obermarck having label p_3 which indicates p_3 article is published in an association with these two authors.

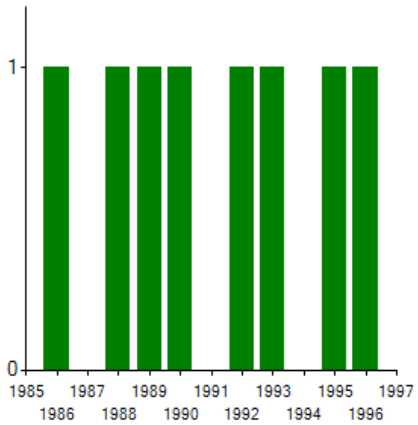

Figure 4.14: Yearwise Publication of Selected Author: Meichun Hsu

Figure 4.14 shows year wise publication graph of an author Meichun Hsu. It is observed that his first publication was in the year 1985 and last publication was in the year 1997. Publication records retrieved are listed below-

Publications

1. 2017-05-27:journals/dpd/HsuK96:https://doi.org/10.1007/BF00204906:ObjectFlow: Towards a Process Management Infrastructure.:169-194:DPD/4/P169.pdf:1996:4:Distributed and Parallel Databases:2:db/journals/dpd/dpd4.htmlHsuK9 COAUTHORS
   1. Charly Kleissner

2. 2017-03-30:journals/tods/HsuZ92:Performance Evaluation of Cautious Waiting.: 477-512:1992:17:ACM Trans. Database Syst.:3:db/journals/tods/tods17.htmlHsuZ92:http://

doi.acm.org/10.1145/132271.132275:TODS17/P477.PDF:journals/tods/AgrawalCL87:
journals/tse/AgrawalCM87:conf/podc/BalterBD82:journals/tods/BernsteinSR80:conf/
vldb/CareyS84:journals/cacm/ChesnaisGM83:journals/tods/FranaszekR85:journals/cacm/
EswarranGLT76:conf/ac/Gray78:conf/sigmod/IraniL79:...:journals/tods/KungR81:journals/
jacm/Papadimitriou79b:journals/cacm/PotierL80:journals/tods/RosenkrantzSL78:...:journals
jacm/TaySG85:journals/tods/TayGS85:.. COAUTHORS

1. Bin Zhang 0004

3. 2012-09-17:journals/debu/HsuOV93:Workflow Model and Execution.:45-48:1993:16:IEEE
Data Eng. Bull.:2:DEBU/93JUN
-CD.PDF:db/journals/debu/debu16.htmlHsuOV93:http://sites.
computer.org/debull/93JUN-CD.pd
COAUTHORS

1. Ron Obermarck
2. Roelof Vuurboom

4. 2012-09-17:journals/debu/Hsu95:Letter from the Special Issue Editor
. :2-3:1995:18:IEEE Data Eng. Bull.:1:DEBU/95MAR-CD.PDF:db/journals/debu/debu18
.htmlHsu95:http://sites.
computer.org/debull/95MAR-CD.pd

5. 2017-03-30:journals/tods/HsuC86:Partitioned Two-Phase Locking.:
431-446:TODS11/P431.PDF:1986:11:ACM Trans. DatabaseSyst.:4:db/journals
/tods/tods11.htmlHsuC86:http://doi.acm.org/10.1145/7239.7477:journals
/tods/BayerHR80:journals/csur/BernsteinG81:journals/tods/BernsteinG83:
journals/tods/BernsteinSR80:journalstocs/CareyM86:journals/tse/ChanG85:conf/
hpts/ChanDH85:conf/sigmod/ChanFLNR82:journals/cacm/EswarranGLT76:journals/
tods/GarciaMolinaW82:conf/ac/Gray78:...:...:conf/pods/HsuM83:conf/vldb/
KedemS80:journals/jacm/KedemS83:conf/sigmod/KungP79:journals/cacm
/Lamport78:journals/tods/PapadimitriouK84:journals/jacm/SilberschatzK80: journals/tse/SilberschatzK82:conf/sigmod/StearnsR8

COAUTHORS

1. Arvola Chan

6. 2017-05-26:journals/ipl/HsuM88:Shifting Timestamps for Concurrency Control in an Information
Hierarchy.:291-297:1988:27:Inf. Process. Lett.:6:db/journals/ipl/ipl27.
htmlHsuM88:https://doi.org/
10.1016/0020-0190(88)90216- COAUTHORS
1. Stuart E. Madnick

7. 2017-05-20:journals/is/HsuM89:Hierarchical timestamping algorithm.:117-129:1989:14:
Inf.
Syst.:2:db/journals/is/is14.htmlHsuM89:https://doi.org/10.1016/0306-4379(89)90040-
COAUTHORS
1. Stuart E. Madnick

8. 2017-05-27:journals/isci/HsuTY90:Concurrent operations in linear hashing.:193-211:1990:51:Inf.
Sci.:2:https://doi.org/10.1016/0020-0255(90)
90026-7:db/journals/isci/isci51.htmlHsuTY9
COAUTHORS
1. Shang-Sheng Tung
2. Wei-Pang Yang

Table 4.11 shows performance measures of co-authors of Meichun Hsu

Table 4.11: Performance Measures of Co-Authors of Meichun Hsu

Sr. No.	Author Name	Number of Co Authors	Total Number of Publications	Performance Measure Parameters			
				Consistency in %	Cooperativeness	Solidity	Contribution factor
1.	Bin Zhang	1	1	47	9.05	1	5.2
2.	Stuart E. Madnick	6	16	74	11.76	3	11.6
3.	Charly Kleissner	0	0	0	0	0	0
4.	Shang-Sheng Tung	0	0	0	0	0	0
5.	Ron Ober-marck	0	2	0	0	0	2
6.	Roelof Vuru-boom	0	0	0	0	0	0
7.	Wei-Pang Yang	0	0	0	0	0	0
8.	Arvola Chan	4	4	0.5	4.41	0	2.4

4.2.4 Author and Co-author Hierarchy

Figure 4.15 shows the author and co- author hierarchy. Umeshwar Dayal as the main author at first level and his all 24 co-authors are shown at the second level. Among those co-authors, Meichun Hsu is retrieved and explored with his 8 co-authors.

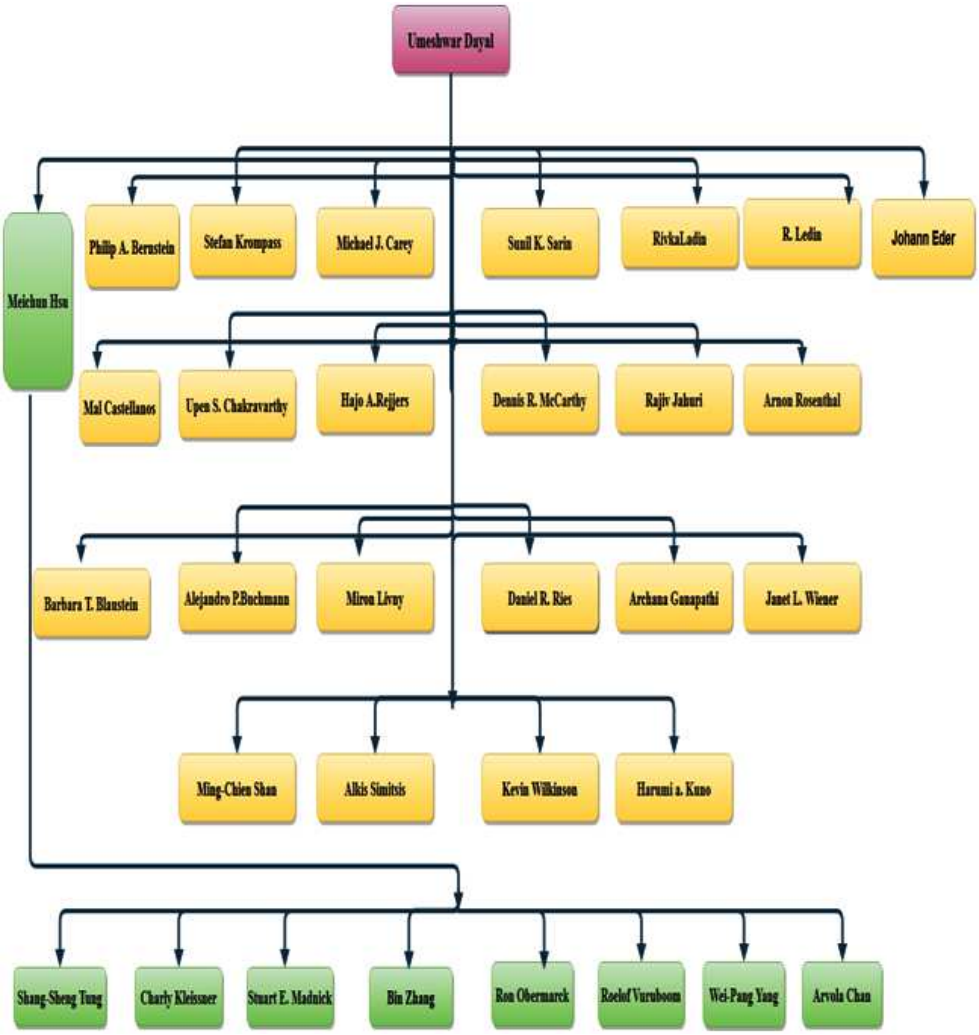

Figure 4.15: Author and Co-Author Hierarchy

4.2.5 Most Influential Author Retrieval

In an analysis of bibliographic record, it is required to find authors who are most active in particular period of time. These active authors are known as most influential authors for a selected period of time. To find the most influential author, CPA algorithm is to a graph containing bibliographic records. After k partitioning, each partition generates most influential author. Experimentation is performed on DBLP dataset for span of years 1991 to 1995 to find the 10 most influential authors in the span. This span contains 57694 authors and 118680 publication records. To find the 10 most influential author, CPA is applied and 10 partitions are generated as shown in figure 4.16(1) to 4.16(10). Figure 4.16 (1) to 4.16(10) reveals that the centrally placed vertex indicates author_id of most influenced

author and its connected vertices show auhor_id of all co-authors network.

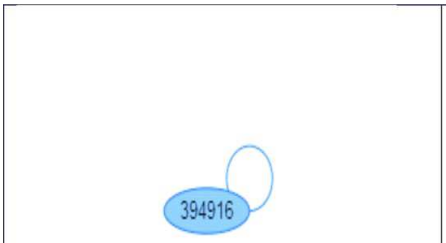

Figure 4.16(1):Partition 1

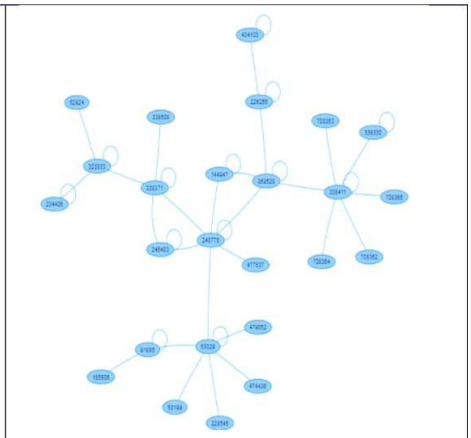

Figure 4.16(2): Partition 2

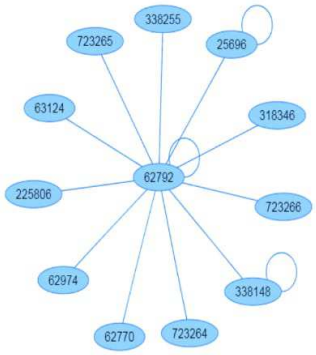

Figure 4.16(3): Partition 3

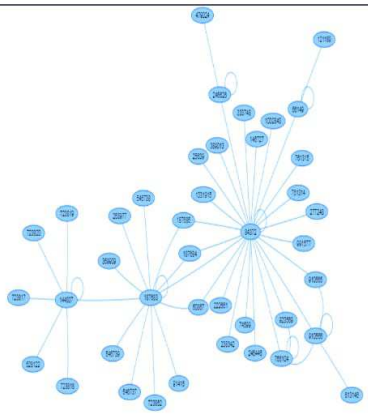

Figure 4.16(4): Partition 4

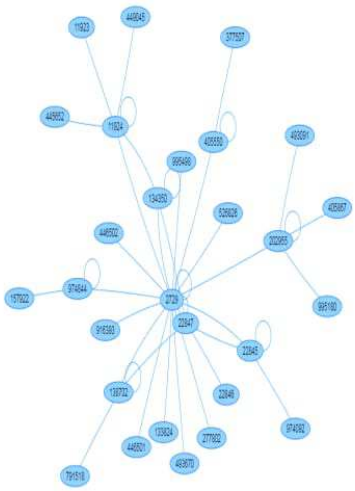

Figure 4.16(5): Partition 5

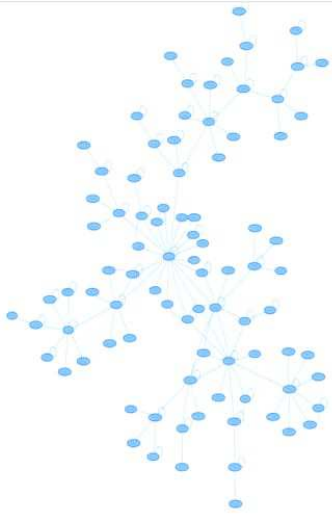

Figure 4.16(6): Partition 6

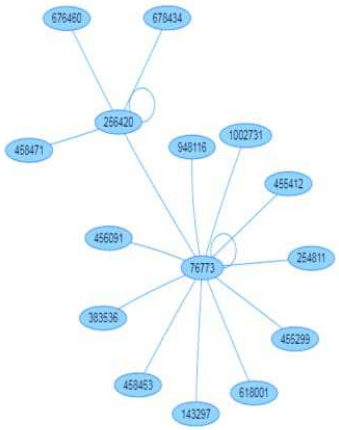

Figure 4.16(7): Partition 7

Figure 4.16(8): Partition 8

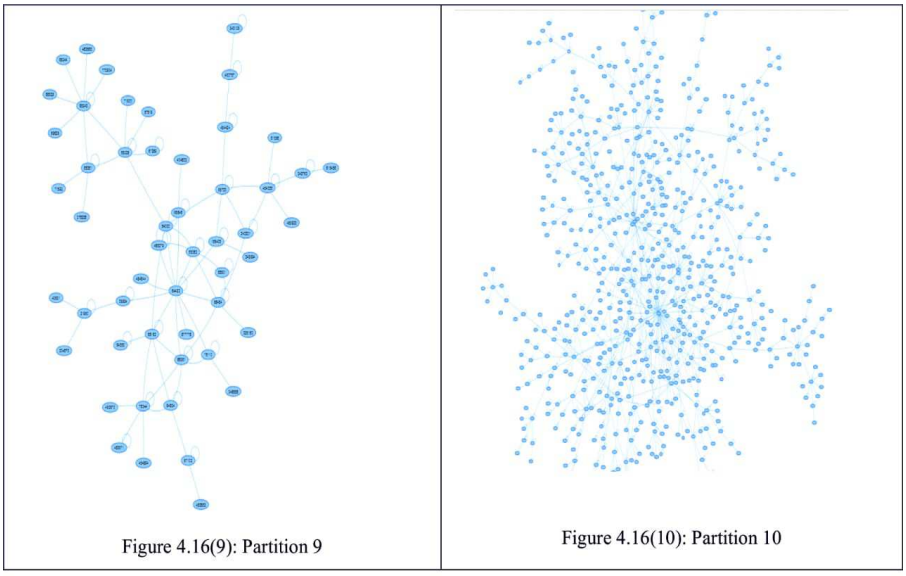

Figure 4.16(9): Partition 9

Figure 4.16(10): Partition 10

Figure 4.16: 10 Most Influential Authors Id of Span1991 - 1995

Along with author_id author names of most influential authors from a specified span are also retrieved. CPA algorithm is applied for a span of 1986 to 1990. This span contains 13566 authors and 30880 publication records. 9 partitions are generated to retrieve the 9 most influential authors.

Figure 4.17(1):Partition 1

Figure 4.17(2): Partition 2

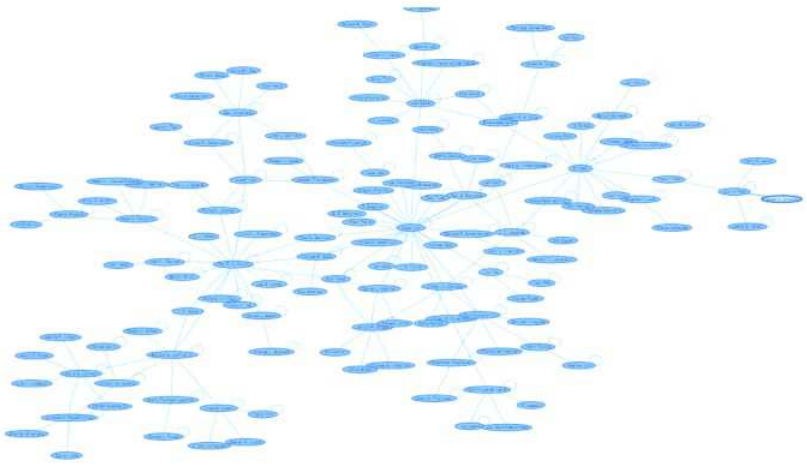

Figure 4.17(3): Partition 3

Figure 4.17(4): Partition 4

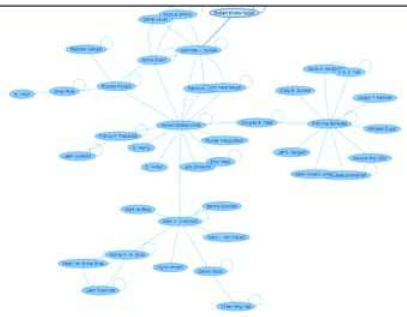

Figure 4.17(5): Partition 5

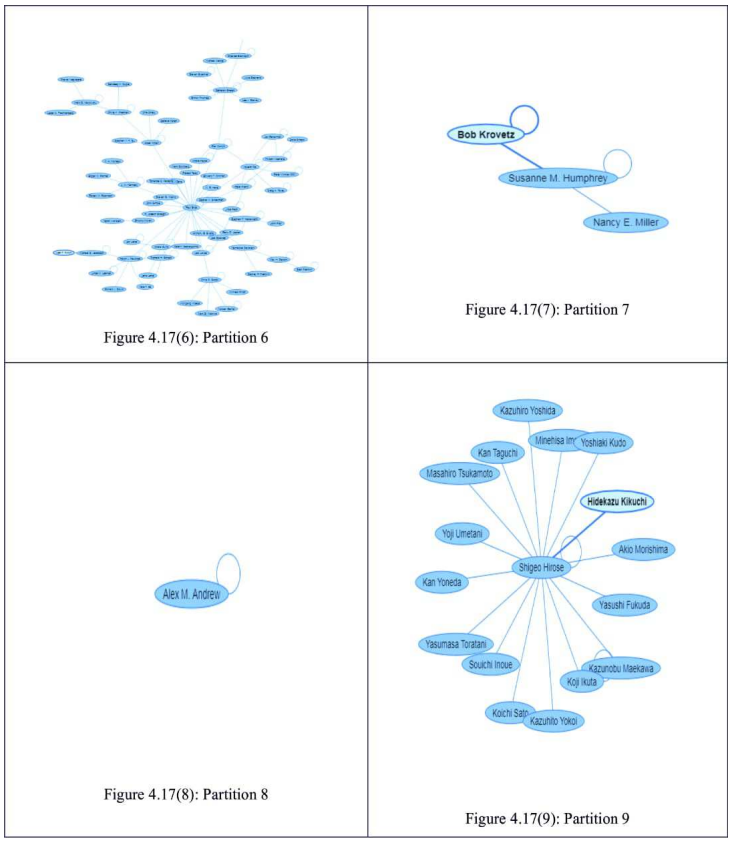

Figure 4.17(6): Partition 6

Figure 4.17(7): Partition 7

Figure 4.17(8): Partition 8

Figure 4.17(9): Partition 9

Figure 4.17: 9 Most Influential Authors of Span 1986 - 1990

Figure 4.17(1) to 4.17(9) reveals 9 most influential authors retrieved from 9 partitions and their interaction network with co-authors. Figure 4.17(3),4.17(5),4.17(6) are attached separately.It is noticed that 9 most influential are - Tony Woven, Vivekre Fk, Naga Alon, Herbert Edelsbrunner, Paul Erds, Sussane M. Humphery, Alex M. Andrew, Shigeo.

In this chapter results of novel CPA algorithm, graph visualization algorithm and author information retrieval algorithm are discussed in detail. Performance analysis of novel algorithms is studied. Performance analysis is carried out for graph benchmark dataset and DBLP dataset. From performance analysis it has been observed that novel algorithms developed provides betterment in results.

Chapter 5

Conclusions and Future Scope

In order to investigate the performance of designed algorithms, we carried out experiments on benchmark graphs. This chapter summarizes the research work done, conclusions based on experimentation performed and provides important directions for future work.

5.1 Conclusions

The central problem addressed in this thesis is partitioning a large graph into smaller sub-graphs with minimum edge-cuts in order to achieve better insights of information stored in a large graph with efficient visualization tools. The research presents the solution for analysis and Visualization of sizably voluminous graphs. For analysis of any large size graph, a partitioning algorithm and visualization tool are developed. The technical details of the novel efficient algorithm, visualization tool, and application of this algorithm and visualization tool are discussed.

1. **Cut-set and Partitioning Algorithm (CPA):** More efficient algorithm is developed. It performs partitioning of a large graph into sub- graphs based on stronger connected components and set differences of degrees of vertices. The performance of the algorithm is evaluated using standard datasets as input which are considered as benchmarks for assessment of graph partitioning algorithms. It is observed that number of edge-cuts required for partitioning is comparatively much lesser. **The percentage reduction in edge-cuts is in the range of 10% to 50%.** Time required for execution is lesser as compared to existing graph partitioning techniques. It is fur-

ther observed that for a graph of approximately **30,000 edges, partitioning time required is just 13 seconds.**

2. **An Effective Visualization tool** is developed for visualization of partitioned sub-graphs. This visualization tool is capable of dynamic visualization of 128 sub-graphs obtained after partitioning a large graph. The tool avoids clutters of edges and efficiently utilizes display area, provide better and quick visualization of a graph with vertex labels and edge labels.

3. The core conclusion drawn from research work is finding most connected components from the set of vertices and computation of set difference of degree of vertices lead to an efficient partitioning of large graph into smaller sub graphs with least edge-cuts.

4. **An Author's Publication Information Retrieval system** is developed for finding publication details of particular author and interactions with co- authors. The visualization tool is applied over DBLP dataset to analyze authors network in bibliographic records and it exhibits correct details of chosen author including his/her co-authors and publication details. Experimental results reveal that DBLP processing system retrieves the authors publication details efficiently and quickly.

- The developed algorithm quickly finds and visualizes publication details of a selected author from millions of bibliographic records.

- Estimates the authors performance by calculating values of performance estimation parameters like stability, cooperativeness, solidity, and consistency and contribution factor.

- The performance values reveal that authors consistency is observed in high percentage. It is observed that an author with good consistency has a good contribution factor too. Contribution factor is efficiently computed which gives contribution as main author and as co- authors too.

- The n most influential authors in a specified period of time are found out efficiently by using **Cut-set and Partitioning Algorithm (CPA)** from a specified span of DBLP dataset.

5.2 Future Scope

We provide few research directions as future scope of the work.

- Our developed Cut-set Partitioning Algorithm (CPA) efficiently performs partitioning of any graph, computes minimum number of edge-cuts in lesser time. Performance of algorithm can be further improved with parallel implementation with concurrent execution of independent partition.

- In case of biological databases link Gnome, it is observed that an efficient tool is required for some complex task to name few- computing range of functional attributes, visualizing biological insights for large proteomic, to extract single gene product that may be annoted to multiple geo terms, to compute number of unique features with ability to tailor annotation sets using multiple filtering options, to construct subset of geo known as geo- slim to map up annotations allowing a general overview of attributes of a set of proteins. This research work can be extended further for developing a tool to support such complex tasks.

- Social network carries huge information in various forms. Our developed tool can be better utilized for retrieving information with several parameters and effectively visualize the information.

Research Publications

Patent

1. Ms. Swati K. Bhavsar, Dr. Varsha H. Patil, "IoT Based System for Computing Human Density for Critical Situation Recognition and Precaution for Avoiding Accident", patent Reference No. 201721046250, Application Number : No.201721046250A https://ipindiaservices.gov.in/PublicSearch/PublicationSearch/ApplicationStatus

Copyrights

1. Ms. Swati K. Bhavsar, Dr. Varsha H. Patil, "Development of Novel algorithm for Analysis and Visualization of Large Graph", Registration No. : L-69172/2017

2. Ms. Swati K. Bhavsar, Dr. Varsha H. Patil, "An Author Publication Information Retrieval Algorithm", Registration No: L-76338/2018

3. Ms. Swati K. Bhavsar, Dr. Varsha H. Patil, "An Efficient Partition Building and Cut set Computing Algorithm(AEPBCCA)", Copyright Diary Number: 10787/2018-CO/L

4. Ms. Swati K. Bhavsar, Dr. Varsha H. Patil, "An Efficient Graph Visualization Algorithm ", Copyright Diary Number: 10808/2018-CO/L

Paper Published

1. Swati K. Bhavsar, Dr. Varsha H. Patil, "Large Graph Analysis and Visualization in Graph Mining: A Survey", I J C T A, 10(8), International Science Press, ISSN: 0974-5572, December 2016, pp. 369-376,Poster presented at 12th Inter-Research-Institute Student Seminar in Computer Science, IRISS Conference 2018.

2. Swati K. Bhavsar, Dr. Varsha H. Patil, "Large Graph Analysis and Visualization", cPGCON-2016, Research Scholar forum, Savitribai Phule Pune University.

3. Swati K. Bhavsar, Dr. Varsha H. Patil, "Image Segmentation using Graph analysis", ICCCC- 2017.ISBN: 978-93-86447-49-4, pp 135-138.

4. Swati K. Bhavsar, Dr. Varsha H. Patil, "Graph Visualization System for Human Density Computation using IoT", ICICI 2018, Coimbatore, Springer series, India, August 2018.

5. Swati K. Bhavsar, Dr. Varsha H. Patil , "Graphviz: An Interactive Visualization Tool for Densed Graphs", Accepted for International Conference on Recent Innovations in Engineering and Technology Crete, Greece, October 2018.

6. Swati K. Bhavsar, Dr. Varsha H. Patil, "Graph Partitioning and Visualization in Graph Mining: A Survey", [submitted to IGI: Data Mining and Databases]

7. Swati K. Bhavsar, Dr. Varsha H. Patil, "An Efficient Cut-set Computation, Partitioning and Visualization tool for Large Graph", [submitted to IEEE Transactions: Knowledge and Data Engineering]

References

[1] Jose F. Rodrigues, Hanghang Tong, Jia Yu Pan, Agma J. M. Traina, Caetano Traina, Christos Faloutsos, "Large Graph Analysis in the GMine System", IEEE Transactions on Knowledge and Data Engineering, Vol.25, No.1, January 2013, pp.106-118.

[2] Chaw Wei, Ou, Sanjay Ranka, "Parallel Incremental Graph Partitioning", IEEE transactions on Parallel and Distributed Systems, Vol 8, No. 8, August 1997.

[3] C. R. Palmer, C. Faloutos, "Electricity Based External Similarity of Categorical Attributes", Proc Seventh Pacific Asia Conf. Advances in Knowledge Discovery and Data Mining, 2003, pp. 486-500.

[4] G. Karypis, V. Kumar, "Multilevel Graph partitioning schemes", Proc. IEEE/ ACM conf. parallel processing, 1995, pp. 113-122.

[5] Inderjit S. Dhilon, Yuqiang Guan, Brian Kulis, "Weighted Graph Cuts without Eigenvectors: A Multilevel Approach", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.29, No.11, November 2007, pp. 1944-1957.

[6] Mahmudar Rahman, Mansurul Alam Bhuiyan, Mohammad Al Hassan "GRAFT: An efficient Graphlet counting method for Large Graph Analysis", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.26, No.10, October 2014, pp. 2466-2478.

[7] Vladimir Batagelj, Franz J. Bran denburg, Water Didimo,"XY Clustering and Hybrid Visualization Technique ", IEEE Transactions on Knowledge and Data Engineering, Vol. 17, November 2011, pp. 1587-1598.

[8] James Abello, Frank Van Ham, Neeraj Krishan, "ASK GraphView: A Large Scale Graph Visualization System", IEEE Transactions on Visualization and Computer Graphics", Vol. 12, No.5, pp. 1587-1598 October 2006.

[9] J. Zhang. The interaction of internal and external representations in a problem solving task. Proc. Thirteenth Annual Conference of Cognitive Science Society, 1991.

[10] Weiwei Cui "A Survey on Graph Visualization", Hong Kong University of Science and Technology Clear Water Bay, Kowloon, Hong Kong, pp. 1-52

[11] S. Palmer and I. Rock., "Rethinking perceptual organization: The role of uniform Connectedness", Psychonomic Bulletin Review, 1994, pp. 29-55.

[12] H. Motoda, "What Can We Do with Graph-Structured Data A Data Mining Perspective", Springer 2006, pp. 1-20

[13] N. S. Ketkar, L.B.Holder and OJ. Cook, "Empirical Comparison of Graph Classification Algorithms", IEEE, 2009.

[14] S. Kim, "Graph theoretic sequence clustering algorithms and their applications to genome comparison", in: J.T.L. Wang, C.H. Wu, P.P. Wang (Eds.), Computational Biology and Genome Informatics, World Scientific Publishing Company, 2003, pp. 81-116.

[15] R. Kannan, S. Vempala, A. Vetta, "On clusterings good, bad and spectral", Journal of the ACM 51 (3) (2004) pp. 497-515.

[16] J. Callut, K. Franisse, M. Saerens and P. Dupont, "Semi-supervised Classification from Discriminative Random Walks", Lecture Notes in Artificial Intelligence No. 5211, Springer, 2008, pp. 162-177.

[17] Geoffrey Ellis and Alen Dix. : Taxonomy of Clutter Reduction for Information Visualization, IEEE Transactions on Visualization and Computer Graphics, 13(6), 2007, pp. 1216- 1223.

[18] M. R. Garey and D. S. Johnson., "Crossing number is NP-complete.", SIAM Journal on Algebraic and Discrete Methods, 4(3), 1983, pp. 312-316.

[19] Meng Qi Yelena Wu, Robert Faris, Kwan- Liu Ma, "Visual Exploration of Academic Career Paths", 2013, IEEE/ ACM International Conference on Advances in Social Networks Analysis and Mining, August 2015, pp. 779-786.

[20] H. Kashima and A. Inokuchi, "Kernels for graph classification", ICDM, Workshop on Active Mining 2002.

[21] Raghavendra, P. , "Optimal algorithms and inapproximability results for every CSP", In Proceedings of the 40th Annual ACM Symposium on theory of Computing, 2008, pp. 245-254.

[22] J. Callut, K. Fran90isse, M. Saerens and P. Dupont, "Semi-supervised Classification from Discriminative Random Walks", Lecture Notes in Artificial Intelligence No. 5211, Springer, 2008, pp. 162-177

[23] Orlin, J. B. (2013). "Max flows in O(nm) time, or better", . Proceedings of the 2013 Symposium on the Theory of Computing, pp. 765-774.

[24] J. E. Beasley, editor, Advances in Linear and Integer Programming", Oxford Science, 1996.

[25] WilliamW. Hager Dzung T. Phan Hongchao Zhang, " An Exact Algorithm For Graph Partitioning " , Springer and Mathematical Optimization Society 2011, pp. 1-10.

[26] Daniel Delling and Renato F. Werneck, "Faster Customization of Road Network", International Symposium on Experimental Algorithms, springer, pp. 30-42.

[27] Alex Pothen, Horst D. Simon, and Kang-Pu Liou, "Partitioning sparse matrices with eigenvectors of graphs. SIAM Journal of Matrix Analysis and Applications", 11(3): 1990, pp. 430-452.

[28] Koulamas, Christos. "A new constructive heuristic for the flowshop scheduling problem." European Journal of Operational Research 105.1 ,1998, pp. 66-72.

[29] Bruce Hendrickson, Robert Leland, "A multilevel algorithm for partitioning graphs", Supercomputing '95 Proceedings of the 1995 ACM/IEEE conference on Supercomputing,ISBN:0-89791-816-9

[30] T. Bui and C. Jones. A heuristic for reducing fill in sparse matrix factorization. In 6th SIAM Conf. Parallel Processing for Scientific Computing, 1993, pp. 445-452.

[31] Bruce Hendrickson and Rober Leland, "An improved spectral graph partitioning algorithm for mapping parallel computations" Technical Report SAND92-1460, Sandia National Laboratories, 1992.

[32] Alex Pothen, H. D. Simon, and Lie Wang, " Spectral nested dissection", Technical Report 92-01, Computer Science Department, Pennsylvania State University, University Park, PA, 1992.

[33] Gary L. Miller, Shang-Hua Teng, and Stephen A., " A unified geometric approach to graph separators", In Proceedings of 31st Annual Symposiumon Foundations of Computer Science, 1991, pp. 538-547.

[34] B. W. Kernighan and S. Lin., "An efficient heuristic procedure for partitioning graphs", The B System Technical Journal, 1970.

[35] A. George, " Nested dissection of a regular finite-element mesh", SIAM Journal on Numerical Analysis, 1973, pp. 345-363.

[36] A. George and J. W.-H. Liu. , "Computer Solution of Large Sparse Positive Definite Systems", Prentice-Hall, Englewood Cliffs, NJ, 1981.

[37] George Karypis and Vipin Kumar, "Multilevel Graph Partitioning Schemes", pp. 1-13.

[38] Alex Pothen, Horst D. Simon, and Kang-Pu Liou., "Partitioning sparse matrices with eigenvectors of graphs", SIAM Journal of Matrix Analysis and Applications, 11(3), 1990, pp. 430-452.

[39] Chao-Wei Ou and Sanjay Ranka "Parallel Incremental Graph Partitioning" IEEE Transactions Onparallel And Distributed Systems, Vol. 8, No. 8, August 1999.

[40] Stephan T. Barnad, Horst D. Simon, "Fast multilevel Implementation of recursive spectral bisection for partitioning unstructured problems, Concurrency: Practice and Experience", Vol6 (2) ,1994, pp. 101-117.

[41] Tefeng Chen , Bo Li, "A Distributed Graph Partitioning Algorithm for Processing Large Graphs", IEEE Symposium on Service-Oriented System Engineering (SOSE), 2016.

[42] Maria Predari, Aurlien Esnard, "A k-Way Greedy Graph Partitioning with Initial Fixed Vertices for Parallel Applications", 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), 2016.

[43] Angen Zheng, Alexandros Labrinidis, Panos K. Chrysanthis, "Argo: Architecture-aware graph partitioning", IEEE International Conference on Big Data, 2016.

[44] Lyu-Wei Wang, Shih-Chang Chen, Wenguang Chen, Hung-Chang Hsiao, Yeh-Ching, " BiFennel: Fast Bipartite Graph Partitioning Algorithm for Big Data", IEEE International Conference on Smart City/SocialCom/SustainCom, 2015.

[45] Michael Ley, "DBLP Some Lessons Learned".

[46] Alpert C. J., Kahng, "Recent Direction in Netlist Partitioning: A Survey", the VLSI Journal Vol. 19, 1-2, 1995, pp. 1-81.

[47] Lawrence B. Holder , Diane J. Cook , Surnjani Djoko, "Substructure Discovery In The Subdue System" , In Proc. of the AAAI Workshop on Knowledge Discovery in Databases, 1994.

[48] Luc Dehaspe , Hannu Toivonen , Ross Donald King , "Finding frequent substructures in Chemical compounds", 1998.

[49] Akihiro Inokuchi, Takashi Washio and Hiroshi Motoda, "An apriori-based algorithm for mining frequent substructures from graph 204 T. Ramraj and R. Prabhakar / Procedia Computer Science 47 (2015) 197 204 data", Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery.

[50] M.Kuramochi and G.Karypis, " Frequent subgraph discovery", In Proc of ICDM, 2001.

[51] Xifeng Yan, Hong Cheng, Jiawei Han, Philip S. Yu, "Mining significant graph patterns by scalable leap search", Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008

[52] M.J.Zaki, "Efficiently mining frequent trees in a forest", In Proc 2002, ACM SIGKDD Int Conf. Knowledge Discovery and Datamining (KDD02), 2002.

[53] J.Huan, W.Wang and J.Prins, "Efficient mining of frequent subgraph in the presence of isomorphism", In Proc. 2003 Int Conf. Data Mining (ICDM03).

[54] J.Huan, W.Wang and J.Prins, "SPIN: Mining maximal frequent subgraphs from graph databases', In Proc. 2004. ACM SIGKDD Int Conf. Knowledge Discovery and Datamining (KDD04), 2004.

[55] M.Kuramochi and G.Karypis, "Finding frequent patterns in a large sparse graph", Journal Data Mining and Knowledge Discovery, Volume 11 Issue 3, November 2005.

[56] S.Nijssen and J.Kok, "A quickstart in frequent structure mining can make a difference", In Proc. 2004. ACM SIGKDD Int Conf. Knowledge Discovery and Data mining (KDD04), 2004.

[57] S. B. Akers, " Binary Decision Diagrams", IEEE Transaction Computers, Vol C 27, no. 6, June 1978, pp. 509-516.

[58] David A. Papa and Igor L. Markov, "Hypergraph Partitioning and Clustering", University of Michigan, EECS Department, Ann Arbor, MI 48109-2121, pp. 1-38.

[59] T.H. Huang, M.L. Huang, "Analysis and Visualization of Co-authorship Networks for Understanding", Academic Collaboration and Knowledge Domain of Individual Researchers, Int. Conference on Computer Graphics, Imaging and Visualization, 2006, pp. 18-23.

[60] Milos Kudelka, Zdenek Horak, Vaclav Snasel and Ajith Abraham, "Social Network Reduction Based on Stability " IEEE, Computer Society, 2010.

[61] Vipin Kumar, Ananth Grama, Anshul Gupta, and George Karypis, "Introduction to Parallel Computing: Design and Analysis of Algorithms", Benjamin/Cummings Redwood City, CA, 1994.

[62] N.Vanetik et.al. , "Computing frequent graph patterns from semi structured data", In Proceedings 2002 IEEE International Conference on Data Mining. ICDM-2002.

[63] Geoffrey Ellis and Alen Dix., "Taxonomy of Clutter Reduction for Information Visualization", IEEE Transactions on Visualization and Computer Graphics, 13(6), 2007, pp. 1216- 1223.

[64] D. Eppstein, M.T. Goodrich, and J.Y. Meng., "Confluent Layered Drawings Algorith-Mica", 47(4): 2007, pp. 439-452

[65] D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd, "Flow Map Layout", IEEE Symposium on Information Visualization, INFOVIS 2005, pp. 219-24.

[66] N. Wong, S. Carpendale, and S. Greenberg, "Edge lens: an interactive method for managing edge congestion in graphs, Information Visualization, INFOVIS 2003 pp.51-58.

[67] D. Holten. Hierarchical Edge Bundles, "Visualization of Adjacency Relations in Hierarchical Data", IEEE Transactions on Visualization and Computer Graphics, 12(5), 2006, pp. 741-748.

[68] Karpys G., Kumar S., "A fast and High Quality Multilevel scheme for Partitioning Irregular Graphs", SIAMJ. Science Computer., USA, 1998, pp. 359-392.

[69] Kernighan, B. W. Lin, S, "An efficient Heuristic Procedure for Partitioning graphs", The Bell System Technical Journal, V 49, no. 2, 1970, pp. 291-307.

[70] Fiduccia C. M., Mattheyeses R. M, "A Linear Time Heuristic for Improving Network Partitions", In Proceeding of 19th Design Automation Conference, NJ, USA, IEEE Press, 1982, pp. 175-191.

[71] Q.V. Nguyen and M.L. Huang. "A space-optimized tree visualization", INFOVIS IEEE Symposium on Information Visualization 2002, pp. 85-92.

[72] K. Andrews and H. Heidegger. Inforation slices, "Visualising and exploring large hierarchies using cascading, semi-circular discs", Proc of IEEE Infovis 98 late breaking Hot Topics 1998, pp. 9-11.

[73] EM Reingold and JS Tilford, Tidier Drawings of Trees", IEEE Transactions on Software Engineering, 7(2), 1981, pp. 223228.

[74] T.J. Jankun-Kelly and K.L. Ma. Moire, "Graphs: Radial Focus+ Context Visualization and Interaction for Graphs with Visual Nodes", Proceedings of the IEEE Information Visualization 2003 Conference, Seatle, USA, 2003.

[75] P. Eades, W. Lai, K. Misue, and K. Sugiyama, " Layout Adjustment and the Mental Map", Journal of Visual Languages and Computing, 6(2), 1995, pp. 183-210.

[76] B. Johnson and B. Shneiderman, "Tree-Maps: a space-filling approach to the visualization of hierarchical"

[77] JJ Van Wijk and H. Van de Wetering. Cushion, "tree maps: visualization of hierarchical information", Information Visualization, 1999.(Info Vis 99) Proceedings. 1999 IEEE Symposium on, 1999, pp. 73-78.

[78] P. Young, "Three Dimensional Information Visualisation", Computer Science Technical Report, 1996, pp. 12-16.

[79] N. Henry and J.D. Fekete. MatrixExplorer, "A Dual-Representation System to Explore Social Networks", IEEE Transactions on Visualization and Computer Graphics, 12(5), 2006, pp. 677-684.

[80] R.A. Becker, S.G. Eick, and A.R. Wilks., "Visualizing network Data", IEEE Transactions on Visualization and Computer Graphics, 1(1) 1995, pp.16-28.

[81] D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd., "Flow Map Layout", IEEE Symposium on Information Visualization, 2005. INFOVIS 2005, pp. 219-224.

[82] D. Eppstein, M.T. Goodrich, and J.Y. Meng, "Confluent Layered Drawings.", Algorithmica, 47(4), 2007, 439-452.

[83] D. Holten. Hierarchical Edge Bundles, "Visualization of Adjacency Relations in Hierarchical Data", IEEE Transactions on Visualization and Computer Graphics, 12(5), 2006, pp. 741-748.

[84] N. Wong, S. Carpendale, and S. Greenberg, "Edgelens: an interactive method for managing edge congestion in graphs", Information Visualization, 2003. INFOVIS 2003, pp. 51-58.

[85] JS Risch, DB Rex, ST Dowson, TB Walters, RA May, and BD Moon, "The STARLIGHT Information Visualization System", Proceedings of the IEEE Conference on Information Visualization, 1997, pp. 42-49.

[86] M. Nollenburg and A. Wolff, " A mixed-integer program for drawing high-quality metro maps", Proc. of the 13th International Symposium on Graph Drawing, 2005.

[87] K. Sugiyama and K. Misue, "Visualization of structural information: automatic drawing of compound digraphs. Systems, Man and Cybernetics", IEEE Transactions on, 21(4), 1991, pp. 876-892.

[88] G.G. Robertson and J.D. Mackinlay, "The document lens", Proceedings of the 6th annual ACM symposium on User interface software and technology,1993, pp. 101-108 .

[89] G.H. Golub and C.F. Van Loan., "Matrix computations.", Johns Hopkins University Press Baltimore, MD, USA, 1996.

[90] G. Karypis and V. Kumar. MeTis, " Unstructured Graph Partitioning and Sparse Matrix Ordering System", Version 2.0. University of Minnesota, June, 1995.

[91] S.T. Barnard, " PMRSB: parallel multilevel recursive spectral bisection", Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM), 1995

[92] B.W. Kernighan and S. Lin , "An efficient heuristic procedure for partitioning graphs", Bell System Technical Journal, 49(2), 1970, pp. 291-307.

[93] G. Chartrand and O.R. Oellermann, "Applied and Algorithmic Graph Theory", McGraw-Hill New York, 1993.

[94] B. Hendrickson and R. Leland., "A multilevel algorithm for partitioning graphs", Proc. Supercomputing, 95, 1995.

[95] G. Karypis and V. Kumar., "Multilevel Algorithms for Multi-Constraint Graph Partitioning ", Supercomputing, 1998. IEEE/ACM Conference, pp. 28-38.

[96] M. Delest, F. Bordeaux, J.M. Fedou, N.S. Antipolis, F. Jean-Marc, G. Melancon, and F. Montpellier, "A Quality Measure for Multi-Level Community Structure", SYNASC 8th International Conference, 2006.

[97] S. Mancoridis, BS Mitchell, Y. Chen, and ER Gansner, "Bunch: a clustering tool for the recovery and maintenance of software system structures. Software Maintenance.

[98] F. van Ham, J.J. van Wijk, and T.U. Eindhoven., "Interactive Visualization of Small World Graphs. Information Visualization", 2004, INFOVIS 2004. IEEE Symposium on, 2004, pp.199-206.

[99] Q.W. Feng, R.F. Cohen, and P. Eades., "How to Draw a Planar Clustered Graph. Proceedings of the First Annual International Conference on Computing and Combinatorics", 1995, pp. 21-30.

[100] P. Eades and Q.W. Feng, "Multilevel Visualization of Clustered Graphs", Graph Drawing, Proc. 4th Int. Symp. GD, 1996, pp. 101-112.

[101] P. Eades, Q.W. Feng, and X. Lin, "Straight-Line Drawing Algorithms for Hierarchical Graphs and Clustered Graphs", Proceedings of the Symposium on Graph Drawing 1996., pp. 113-128

[102] J. Ho and S.H. Hong, "Drawing Clustered Graphs in Three Dimensions", 2006

[103] P. Eades and Q.W. Feng, "Multilevel visualization of clustered graphs", Graph Drawing, Proc. 4th Int. Symp. GD,1996, pp. 101-112.

[104] J. Ho and S.H. Hong, "Drawing Clustered Graphs in Three Dimensions", 2006.

[105] John T. Stasko Ji Soo Yi, Youn ah Kang and Julie A. Jacko, "Toward a Deeper Understanding of the Role of Interaction in Information Visualization", IEEE Transactions on Visualization and Computer Graphics, 13(6), 2007, pp. 1224-1231.

[106] C. Ahlberg and B. Shneiderman," Visual Information Seeking: Tight Coupling of Dynamic Query Filters With Starfield Displays", Conference on Human Factors in Computing Systems, 1994.

[107] P. A. Eades," A heuristic for graph drawing", In Congressus Numerantium, volume 42, 1984, pp. 149-160.

[108] G.G. Robertson, J.D. Mackinlay, and S.K. Card. "Cone Trees: animated 3D visualizations of hierarchical information". Proceedings of the SIGCHI conference on Humanfactors in computing systems: Reaching through technology, 1991, pp. 189-194.

[109] X. Liu, J. Bollen, M.L. Nelson, H. Van de Sompel, "Co-Authorship Networks in the Digital Library Research Community", Information Processing Management, vol. 41 2005.

[110] Y. Han, B. Zhou, J. Pei, Y. Jia, "Understanding Importance of Collaborations in Co-authorship Networks, SIAM Int. Conference on Data Mining", 2009, pp. 1112-1123.

[111] Zdenek Horak, Milos Kudelka, Vaclav Snasel, Ajith Abraham, "Forcoa.NET: An Interactive Tool for Exploring the Significance of Authorship Networks in DBLP Data", 2011 IEEE, pp. 261-267.

[112] Hendrickson, Bruce. "Chaco: Software for Partitioning Graphs".

[113] Karypis, G.; Kumar, V., "A fast and high quality multilevel scheme for partitioning irregular graphs". SIAM Journal on Scientific Computing. 20 (1): 359, 1999.

[114] Schlag, S.; Henne, V.; Heuer, T.; Meyerhenke, H.; Sanders, P.; Schulz, C. "k-way Hypergraph Partitioning via n-level Recursive Bisection",Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX). Proceedings. Society for Industrial and Applied Mathematics. pp. 53-67, 2015.

[115] Chevalier, C.; Pellegrini, F., "PT-Scotch: A Tool for Efficient Parallel Graph Ordering", Parallel Computing. 34 (6), 2007, pp. 318-331.

[116] Walshaw, C.; Cross, M. (2000). "Mesh Partitioning: A Multilevel Balancing and Refinement Algorithm", Journal on Scientific Computing, 2000.

[117] Diekmann, R. Preis, R.; Schlimbach, F.; Walshaw, C. "Shape-optimized Mesh Partitioning and Load Balancing for Parallel Adaptive FEM", Parallel Computing. 26 (12): 2000, pp. 1555-1581.

[118] https://sparse.tamu.edu/DIMACS10

[119] Almende B.V., “vis library”, Copyright (C) ,2010.

[120] https://www.google.com/search/large graph example.

[121] http://dblp.uni-trier.de/xml.