FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

A GENERALIZED MULTIDIMENSIONAL INDEX STRUCTURE FOR MULTIMEDIA DATA TO SUPPORT CONTENT-BASED SIMILARITY SEARCHES IN A COLLABORATIVE SEARCH ENVIRONMENT

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Kasturi Chatterjee

2010

To: Dean Amir Mirmiran
College of Engineering and Computing

This dissertation, written by Kasturi Chatterjee, and entitled A Generalized Multidimensional Index Structure for Multimedia Data to Support Content-Based Similarity Searches in a Collaborative Search Environment, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Jainendra K. Navlakha

Xudong He

Keqi Zhang

Mei-Ling Shyu

Shu-Ching Chen, Major Professor

Date of Defense: March 31, 2010

The dissertation of Kasturi Chatterjee is approved.

Dean Amir Mirmiran
College of Engineering and Computing

Interim Dean Kevin O'Shea
University Graduate School

Florida International University, 2010

DEDICATION

To Baba and Ma

ACKNOWLEDGMENTS

I would like to extend my sincere gratitude and appreciation to my dissertation advisor Professor Shu-Ching Chen for his guidance, support, suggestions and encouragement while this dissertation was being conducted. I am also indebted to Professors Jainendra K Navlakha, Xudong He of the School of Computing and Information Sciences, Professor Keqi Zhang of Department of Environmental Studies and International Hurricane Center, and Professor Mei-Ling Shyu of the Department of Electrical and Computer Engineering, University of Miami, for accepting to be in my dissertation committee, as well as for their suggestions and support.

The financial assistance I received from the School of Computing and Information Sciences and the Dissertation Year Fellowship from University Graduate School are gratefully acknowledged.

I would like to thank all my friends and colleagues whom I have met and known while attending Florida International University and all my group members. In particular, I would like to thank Min Chen, Na Zhao and Fausto Fleites for their generous help. Finally, my utmost gratitude goes to my parents for their unconditional love and for always believing in me; to my brother and sister-in-law for their support and encouragement; to my little nephew for making me feel special without a reason; and to my husband for just being there, always.

ABSTRACT OF THE DISSERTATION

A GENERALIZED MULTIDIMENSIONAL INDEX STRUCTURE FOR MULTIMEDIA DATA TO SUPPORT CONTENT-BASED SIMILARITY SEARCHES IN A COLLABORATIVE SEARCH ENVIRONMENT

by

Kasturi Chatterjee

Florida International University, 2010

Miami, Florida

Professor Shu-Ching Chen, Major Professor

Since multimedia data, such as images and videos, are way more expressive and informative than ordinary text-based data, people find it more attractive to communicate and express with them. Additionally, with the rising popularity of social networking tools such as Facebook and Twitter, multimedia information retrieval can no longer be considered a solitary task. Rather, people constantly collaborate with one another while searching and retrieving information. But the very cause of the popularity of multimedia data, the huge and different types of information a single data object can carry, makes their management a challenging task. Multimedia data is commonly represented as multidimensional feature vectors and carry high-level semantic information. These two characteristics make them very different from traditional alpha-numeric data. Thus, to try to manage them with frameworks and rationales designed for primitive alpha-numeric data, will be inefficient.

An index structure is the backbone of any database management system. It has been seen that index structures present in existing relational database management frameworks cannot handle multimedia data effectively. Thus, in this dissertation, a generalized multidimensional index structure is proposed which accommodates the atypical multidimensional representation and the semantic information carried by different multimedia data seamlessly from within one single framework. Additionally, the dissertation

investigates the evolving relationships among multimedia data in a collaborative environment and how such information can help to customize the design of the proposed index structure, when it is used to manage multimedia data in a shared environment. Extensive experiments were conducted to present the usability and better performance of the proposed framework over current state-of-art approaches.

TABLE OF CONTENTS

CHAPTER PAGE

1 Introduction and Motivation 1
1.1 Challenges . . . 5
1.2 Contributions . . . 11
1.2.1 Generalized Multimedia Index Framework . . . 11
1.2.2 Multimedia Similarity Queries . . . 12
1.2.3 Multimedia Query Refinement . . . 13
1.2.4 Visualizing and Analyzing Multimedia Data Relationships . . . 14
1.3 Scope and Limitations . . . 15
1.4 Outline . . . 15

2 Background and Related Work 18
2.1 Multidimensional Index Structures . . . 19
2.2 Content-Based Image and Video Retrievals . . . 23
2.3 Query Refinement . . . 28
2.4 Query Refinement in Multidimensional Index Structures . . . 30
2.5 Graph Similarity . . . 32

3 Overview of the Framework 35
3.1 Multimedia Data Organization . . . 37
3.1.1 Multimedia Index Structure . . . 37
3.1.2 Multimedia Query Engine . . . 38
3.1.3 Intelligent Multimedia Index and Query Engine Optimizer . . . 39
3.2 Multimedia Retrieval . . . 40
3.2.1 Image Retrieval . . . 40
3.2.2 Video Retrieval . . . 41
3.2.3 Mixed Multimedia Data Type Retrieval . . . 42
3.3 Semantic Relationship . . . 43
3.3.1 Semantic Relationship Matrices . . . 43
3.3.2 Semantic Modeling . . . 44

4 GeM-Tree: A Generalized Multidimensional Index Structure Supporting Image and Video Retrieval 45
4.1 GeM-Tree . . . 46
4.2 Earth Movers Distance . . . 47
4.2.1 Fixed-Length Multimedia Data Signatures . . . 49
4.2.2 Variable-Length Multimedia Data Signatures . . . 50
4.2.3 Node Structures of GeM-Tree . . . 52
4.2.4 Node Insertion . . . 53
4.3 Similarity Search . . . 53
4.3.1 High Level Semantic Relationship . . . 55
4.3.2 Incorporation of Affinity Values . . . 56
4.3.3 Affinity Promotion in GeM-Tree . . . 59

4.3.4 k-NN Search . . . 60
4.4 Empirical Study . . . 63
4.5 Conclusion and Future Work . . . 67

5 Content-Based Image Retrieval Utilizing a Multidimensional Index Structure 69
5.1 Content-Based Image Retrieval in GeM-Tree . . . 69
5.2 Similarity Queries . . . 69
5.2.1 Range Queries . . . 70
5.2.2 k-NN Queries . . . 72
5.3 Experimental Analysis . . . 74
5.4 Conclusion . . . 82

6 Content-Based Video Retrieval Utilizing a Multidimensional Index Structure 83
6.1 Video Modeling . . . 85
6.1.1 Hierarchical Unit-Based Modeling . . . 86
6.1.2 Feature-Based Modeling . . . 87
6.1.3 Video Semantics Modeling . . . 87
6.2 Similarity Search . . . 88
6.3 Experiments . . . 90
6.4 Conclusion . . . 91

7 Hybrid Query Refinement: A Strategy for a Distance Based Index Structure to Refine Multimedia Queries 92
7.1 Hybrid Query Refinement in a Distance-Based Index Structure . . . 94
7.1.1 The Refinement Model for Semantic Relationships . . . 95
7.1.2 Refinement Model for the Feature Space . . . 98
7.1.3 Similarity Search With Hybrid Query Refinement Model . . . 102
7.2 Empirical Study and Evaluation Metric . . . 105
7.3 Conclusion and Future Work . . . 112

8 Generating Social Network Previews Using Graph Similarity 114
8.1 Introduction . . . 114
8.2 Generating Social Network Preview . . . 119
8.2.1 Selecting Nodes . . . 120
8.2.2 Determining Similarity . . . 123
8.2.3 Node Assignment . . . 130
8.2.4 Representative Graph Generation . . . 132
8.3 Evaluation Score . . . 134
8.4 Data Set . . . 137
8.5 Empirical Analysis . . . 139
8.6 Multimedia Data Network . . . 144
8.6.1 Analyzing Multimedia Data Network . . . 148
8.7 Conclusion . . . 154

9 A Distributed Multimedia Data Management over the Grid 156
9.1 Introduction 156
9.2 Related Work 159
9.3 Overall Framework 162
9.3.1 Replicated Multidimensional Index Structure 164
9.3.2 Distributed Query Processing 167
9.3.3 Automatic Load Balancing 172
9.4 Empirical Study 175
9.5 Conclusion and Future Works 179

10 Conclusion and Future Work 181
10.1 Future Work 183

BIBLIOGRAPHY 188

APPENDICES 207

VITA 244

LIST OF TABLES

TABLE PAGE

4.1 Affinity promotion for GeM-Tree 59

4.2 k-NN search algorithm for GeM-Tree 61

4.3 Distance computations during querying the index trees for fixed-length feature distributions 65

4.4 Accuracy for fixed-length feature distribution 65

4.5 Distance computations during index tree formations for variable-length feature distributions 66

5.1 Implementation of range query in GeM-Tree 70

5.2 Implementation of space search for images in GeM-Tree 72

5.3 Implementation of metric search for images in GeM-Tree 73

5.4 Implementation of k-NN search for image retrieval in GeM-Tree 75

7.1 Refined_k-NN search algorithm 100

7.2 Affinity promotion for refined queries 104

7.3 Experimental results 113

8.1 Generating social network preview 120

8.2 Node assignment algorithm maximizing similarity 127

8.3 Representative graph generation 134

8.4 Data set characteristics 135

8.5 Generated pre-determined representative graph characteristics 137

8.6 Generated random representative graph characteristics 141

8.7 Generated clustered graph characteristics 142

8.8 Generated random clustered graph characteristics 144

9.1 Implementation of distributed content-based k-NN similarity search 170

9.2 Load balancing in the distributed multimedia database management framework 173

A.1 Degree centrality computation for the top 100 images of the multimedia data network for COREL datset 208

A.2 Closeness centrality computation for the top 100 images of the multimedia data network for COREL datset 212

A.3 Betweenness centrality computation for the top 100 images of the multimedia data network for COREL datset 217

LIST OF FIGURES

FIGURE		PAGE
1.1	Database management architecture.	3
1.2	Different components of a multimedia database management framework.	4
1.3	(a) Graph plotting the feature-level similarity calculated by euclidean distance function for image id 446 with all other images in the database. (b) Graph plotting the high-level relationships captured by MMM framework for image id 446 with all other images in the database.	8
3.1	Overview of the proposed framework.	36
3.2	General video structure.	37
4.1	Clustering using gaussian mixture models with k=5.	51
4.2	Distances vs. numbers of clusters for variable-length feature distribution.	66
5.1	Distance computation and number of I/O during (a) Building the index trees (b) Range queries (c) k-NN queries	77
5.2	Query results without including the affinity value	78
5.3	Query results for 10-NN query in GeM-Tree	79
5.4	Query results for range search with radius and affinity relationship equal to 0.2 and 0.23, respectively	79
5.5	Query results obtained giving equal importance to similarity measurement and high-level image relationship	80
5.6	Query resultds obtained giving more importance to high-level image relationship	81
6.1	Traditional concept of indexing in video databases from video classification point of view	84
6.2	Distance computation and number of I/O of GeM-Tree compared with sequential search	90
7.1	(a) Accuracy compared over three iterations, (b) Computation time compared over three iterations.	107
7.2	(a) F1 score compared over three iterations, (b) Number of distance computations compared over three iterations.	109
7.3	(a) Computation time score compared over three iterations, (b) Similarity score compared over three iterations, (c) Average model score.	111
8.1	A graph with (a) Star configuration (b) Circle configuration	136
8.2	Characteristics of the original social network graphs	138

8.3 Comparison between original graph and representative graphs for (a) Degree centrality, (b) Betweenness centrality, (c) Closeness centrality 143

8.4 Comparison of E_C between original graph and representative graphs using (a) Degree centrality, (b) Betweenness centrality, (c) Closeness centrality 145

8.5 Comparison of E_C error between original graph and representative graphs using (a) Degree centrality, (b) Betweenness centrality, (c) Closeness centrality 146

8.6 Sample data network for 100 images from COREL 152

8.7 Complete data network for 10000 images from COREL 153

9.1 Overview of the proposed framework. 164

9.2 Distributed query processing. 168

9.3 Relationship of the computation time with the number of distribution nodes during tree generation. 176

9.4 Relationship of the computation time with the number of distribution nodes during k-nn search for data set A. 177

9.5 Relationship of the computation time with the number of distribution nodes during k-nn search for data set B. 177

9.6 Experimental results for load balancing for data set I. 178

9.7 Experimental results for load balancing for data set II. 180

9.8 Experimental results for load balancing for data set III. 180

A.1 Original graph for *enrongraph*_500 222

A.2 Pre-determined representative graph for *enrongraph*_500 223

A.3 Random representative graph for *enrongraph*_500 224

A.4 Clustered graph for *enrongraph*_500 224

A.5 Original graph for *enrongraph*_5000 225

A.6 Pre-determined representative graph for *enrongraph*_5000 226

A.7 Random representative graph for *enrongraph*_5000 227

A.8 Clustered graph for *enrongraph*_5000 227

A.9 Original graph for *enrongraph*_10000 228

A.10 Pre-determined representative graph for *enrongraph*_10000 229

A.11 Random representative graph for *enrongraph*_10000 229

A.12 Clustered graph for *enrongraph_*10000 230
A.13 Original graph for adjnoun 231
A.14 Pre-determined representative graph for adjnoun 232
A.15 Random representative graph for adjnoun 233
A.16 Clustered graph for adjnoun 233
A.17 Original graph for celegansneural 234
A.18 Pre-determined representative graph for celegansneural 235
A.19 Random representative graph for celegansneural 236
A.20 Clustered graph for celegansneural 237
A.21 Original graph for karate 238
A.22 Pre-determined representative graph for karate 238
A.23 Random representative graph for karate 239
A.24 Clustered graph for karate 239
A.25 Original graph for lesmis 240
A.26 Pre-determined representative graph for lesmis 241
A.27 Random representative graph for lesmis 242
A.28 Clustered graph for lesmis 243

CHAPTER 1

INTRODUCTION AND MOTIVATION

Since multimedia data such as images and videos are way more expressive and informative than ordinary text-based data, people find it more attractive to communicate and express with them. With the proliferation of Internet technology and popularity of applications such as facebook, myspace, and google image search, large volume of multimedia data is being accessed and utilized by various communities over the web on a regular basis both for entertainment as well as for work. In addition, multimedia data has other fields of usability, for example in medical imaging for diagnosis purposes, developing interactive distant learning tools, educating the general public about pressing topics, etc. But the very cause for the popularity of multimedia data, the huge and different types of information a single multimedia object carries, makes their efficient management a challenging task.

The multimedia data is atypical in nature due the following two characteristics: the multidimensional representation and the semantic gap. Unlike traditional data which is represented in a single dimension, a multimedia data has a multidimensional representation. Basically, a multimedia object such as an image or a video can be broadly considered to be constituted of two different types of contents viz. the low-level content, that is the visual and the audio, and the high-level semantic content. The visual contents of multimedia data are popularly represented with the help of features such as color histograms [204], texture vectors [124], and shape descriptors [231] extracted from them. Each multimedia object can be represented as a point projected in a multidimensional feature space. The audio contents are typically represented with the help of properties, such as volume, flux, spectral components, etc. as utilized in [144][215]. The semantic content is way more difficult to represent, as a single multimedia object might be perceived differently by different users or even by the same user in different iterations. Researches were conducted in an attempt to map the relation between the

low-level features with the semantic meaning by using an iterative feedback mechanism called Relevance Feedback (RF) [186][187], whereby weights are attached to different components of the feature vector to capture users' information need. But in frequent occasions, it has been seen that the correlation between feature components and semantic interpretations do not follow any regular pattern and the RF method doesn't produce satisfactory results. This is called the Semantic Gap and is one of the major hurdles in generating relevant query results.

The primary function of a database management system is to support efficient data storage and retrieval techniques, typical to the data. As the storage and retrieval requirements of a data change, the database management framework supporting it should also change. It is clear from the discussions in the above paragraph that multimedia data is more complicated and hugely different in composition and representation from traditional text-based data. Thus, it will be inefficient to try to manage them with similar frameworks and rationales as primitive alpha-numeric data. Thus, all/most of the components that the traditional database management framework is made up of should be modified, tuned and customized to accommodate the characteristics of multimedia data and to achieve robust and flexible data storage and management.

A typical database management framework is depicted in Figure 1.1. The major components are (i) Recovery Manager, (ii) Storage Manager, (iii) Access Structure Manager, (iv) Lock Manager, (v) Query Processor, (vi) Query Optimizer, (vii) Plan Executor, (viii) Catalog Manager, and (ix) SQL Compiler/Interpreter. Basically, the design of every component is affected by the data type and applications that need to be supported. For example, the Access Structure Manager houses the index structure/structures for a database management framework and supports the different access mechanisms typical for the particular data type. For multimedia data, the index structure should be multidimensional and should support retrieval mechanisms, such as Content-Based Information retrieval (CBIR), meant to be their preferred access mechanism. Similarly,

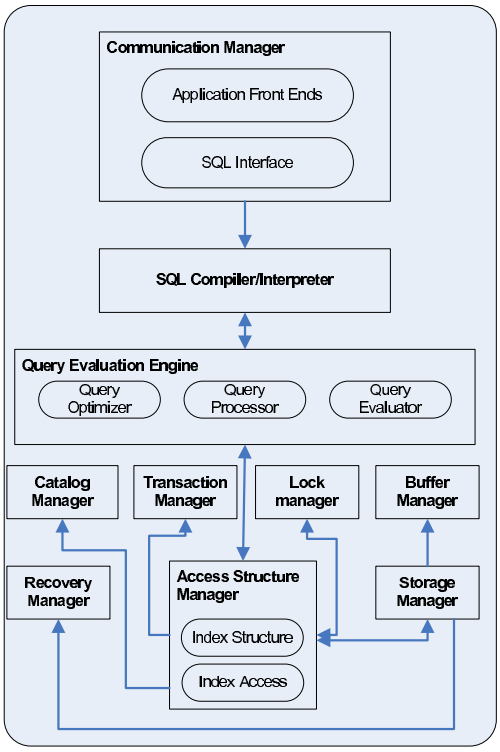


Figure 1.1: Database management architecture.

the Query Processor should be able to process CBIR queries issued for multimedia data objects; the Query Optimizer and Query Planner should utilize and consider new rules and cost models to evaluate the queries utilizing the multidimensional index structures and consider the spatial, temporal and semantic information about the data before developing optimized queries; Query Languages should be able to support interactive query by example [234] methodology and should translate the information need of an user and represent them as formal relations with the help of new or existing operators. A high-level illustration of a Multimedia Database Management architecture is presented in Figure 1.2 where the database management has been clearly divided into two main tasks, viz. the core database management systems (dbms) engine and the content-retrieval engine. These two components communicate among themselves and provide query results to the interface by searching the underlying storage system.

Till date, to the best of our knowledge there wasn't much progress towards developing a complete robust multimedia database management system supporting all these typical characteristics. The existing relational or object relational database models are usually used to accommodate the multimedia data by using BLOBs (Binary Large OBjects) to store images, videos, etc., compressing a multidimensional feature representation of a multimedia data into a single key using space filling Z-order curves [164] and then using single dimensional index structures such as B+-Tree [15]. With multimedia data gaining popularity everyday, such an ad-hoc management approach would fail to meet the quality of service required by the user.

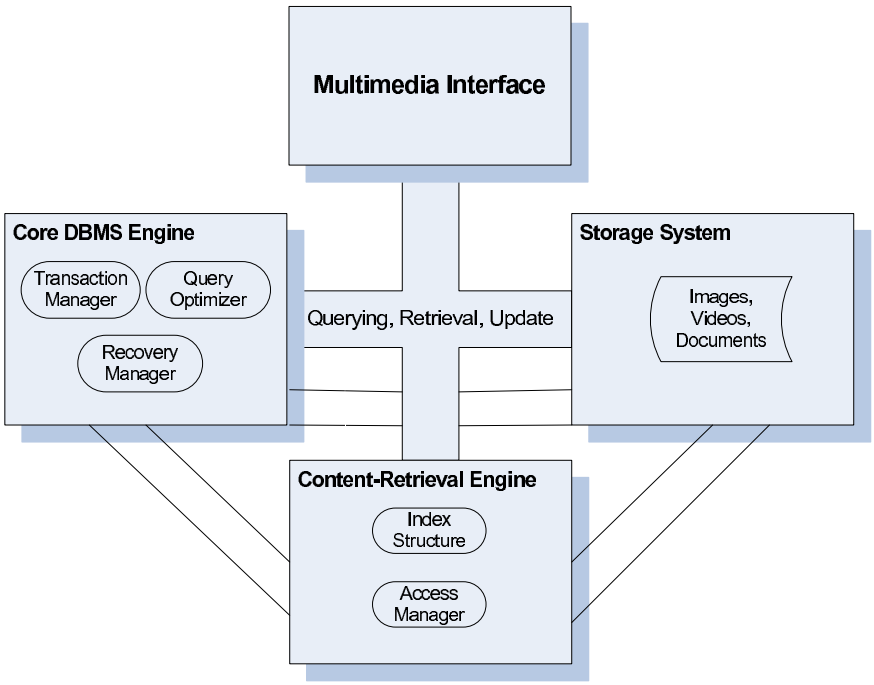


Figure 1.2: Different components of a multimedia database management framework.

The multimedia research community is moving towards attempting to lay down the foundation of an efficient, robust, extensible, and transparent database management framework dedicated for multimedia data. As noticed from the frameworks presented in Figure 1.1 and Figure 1.2, an index structure is one of the pivotal components. This dissertation proposes an indexing framework to index multimedia data and design similarity search algorithms to support popular retrieval mechanisms considering both the contents

that a multimedia data is made up of viz. low-level feature contents and high-level semantic contents. A graph-based social-network approach is also proposed to visualize the high-level semantic relationships between the data objects and to develop a Multimedia Data Network. It is also planned to use data mining and machine learning strategies to make the index structure intelligent and flexible. Developing Query Optimizers or Query Processors based on them should be the next research direction based on the proposed index structure. Thus, this dissertation is an important step towards designing a dedicated database management system for multimedia data with all the components of the framework optimized for the particular characteristics of the data and its effective retrievals.

The remainder of this chapter is organized as follows. In the next section, it briefly discusses the important challenges faced while developing efficient index structures for multimedia data which can support the atypical structures and retrieval strategies associated with them. The significance and major contributions of this dissertation are presented in Section 1.2. In Section 1.3, the scopes and limitations of this framework are discussed. Finally, section 1.4 gives the outline of this dissertation.

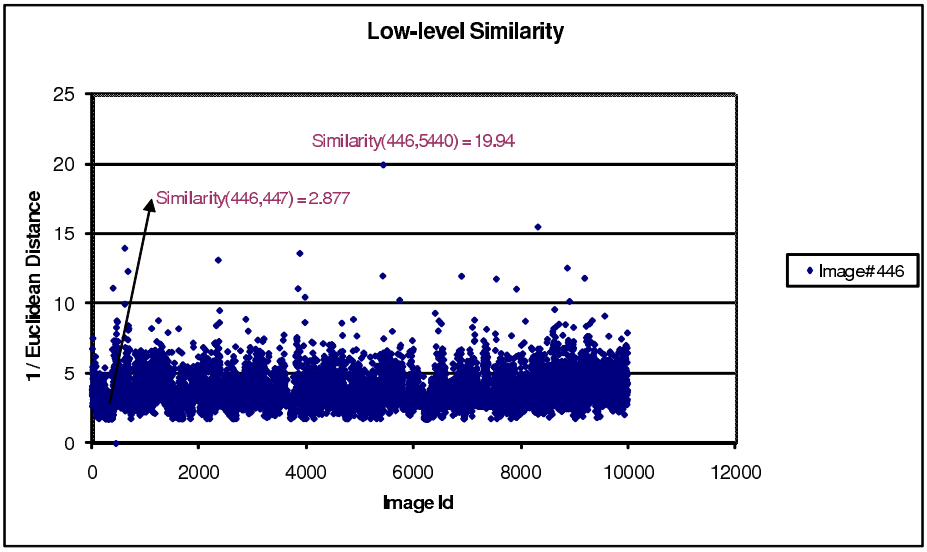
1.1 Challenges

The main challenges in developing an efficient index structure to organize multimedia data and support all the popular retrieval strategies are as follows:

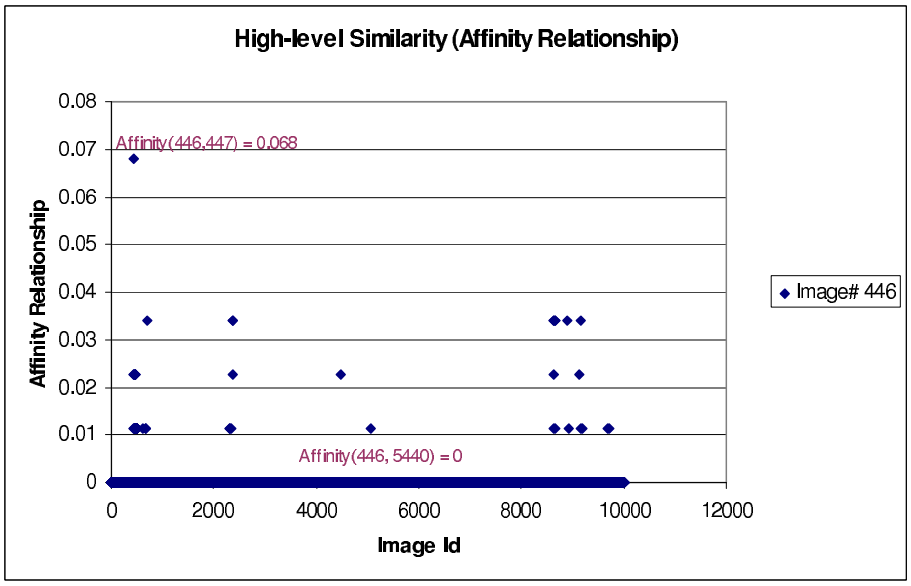
- Multidimensional Feature Set: A multimedia data is represented as a multidimensional feature vector where each field of the vector corresponds to components such as color, texture, shape descriptors, etc. Often the visual contents of a multimedia object cannot be expressed satisfactorily with a fixed length feature vector and a variable length feature representation is used [13]. In addition, though an image can be represented completely by a fixed length or a variable length feature vector,

a video has a more complicated structure. Usually a video is represented with a hierarchical containment structure whereby it is divided into different units such as frames, shots, concepts, etc. A video is usually represented as a collection of smaller units (for example, shots) and is expressed as a set of feature vectors, one vector for each shot. For each shot, each feature vector usually stores the low-level feature values combined over a number of frames (which can be defined as a group of images having a temporal relationship among them). Furthermore, videos contain additional characteristics such as audio feature, temporal information, motion vectors, etc. along with the color, texture, shape features used in images. All these characteristics of the low-level feature need to be accommodated and handled efficiently in the index structure. Obviously, a traditional single dimensional index structure such as B-Tree [14] cannot be utilized for such data and a multidimensional index structure such as R-Tree [98], KDB-Tree [179], Hybrid-Tree [35], etc. should to be utilized. But the existing multidimensional index structures do not support the typical retrieval approaches such as CBIR in their query structures efficiently and have been seen to demonstrate unsatisfactory query results in terms of relevance. Also, none of the existing index structures were found to handle different types of multimedia data, such as images and videos, seamlessly from within a single framework. Additionally, they do not support the hierarchical relationships between the video units and the different unit-level retrievals (such as frame-level, shot-level, video concept-level, etc.). Thus the index structures, to be developed for a MultiMedia DataBase Management System (MMDBMS), should be able to support the multidimensional feature representation, support different multimedia data types from within a single framework, accommodate the different retrieval strategies in their query structure and produce query results relevant to the users' information need.

- Semantic Gap: Every multimedia object, whether an image or a video has high-level semantic meaning attached to it. The relationship between the low-level feature content and the high-level semantic concept of a multimedia object is rather fuzzy. Most of the times it is quite difficult to capture and express the semantic information need of an user about a multimedia data via feature-level weights and thus there exists this semantic gap between them. Figure 1.3 explains it graphically where Figure 1.3(a) plots the similarity between every pair of image in a database and Figure 1.3(b) plots the number of times two images from the same database were voted to be semantically close by a group of users. It can be seen that the low-level similarity pattern does not match completely with the high-level similarity pattern for the same set of images. A particular case is highlighted which demonstrates this fact for a pair of images which has a low feature-level similarity, calculated as the inverse of the Euclidean Distance between the feature vectors, but a high semantic level closeness as has been pointed out by a large number of users. Thus query models proposed in [36][37] fail to produce satisfactory query results for datasets where there is no proper mapping between the features and the semantic concepts. Video data is different from image data and bear additional hierarchical semantic relationships among the different units, such as frames, shots, etc. The existing query models meant to bridge the semantic gap do not possess techniques to handle the typical characteristics of video data. Also, these query models for multimedia data (specifically designed for images) cannot be utilized and embedded into a distance based index structure as adjusting feature-level weights make the distance functions arbitrary. Thus there are three major limitations of [36][37] viz. (a) they cannot be used for distance-based index structures (b) they are not designed to be used to bridge the semantic gap for video data and (c) they fail to produce satisfactory query results if the low-level similarity does not follow the same pattern as semantic closeness between two multimedia data objects. Thus,



(a)



(b)

Figure 1.3: (a) Graph plotting the feature-level similarity calculated by euclidean distance function for image id 446 with all other images in the database. (b) Graph plotting the high-level relationships captured by MMM framework for image id 446 with all other images in the database.

the challenge is to develop generic query models to be embedded into the k-NN similarity search methods of index structures which precisely address the above three issues.

- Imprecise Query for Similarity Search: The perception subjectivity attached to a multimedia object makes the queries imprecise in nature. By perception subjectivity, it is meant that the semantic meaning attached to a multimedia object is interpreted differently by different users or even by the same user at different points of time. Thus the actual information need of an user issuing a query cannot be captured precisely in one iteration. Multiple iterations, with user in the feedback loop, capture the users' perception with greater accuracy. Such retrieval techniques are called CBIR with Relevance Feedback (RF), whereby the relevance of the query results are marked by the user in the form of feedbacks which are utilized in the next iteration by the query model to improve the query results. The process is called query refinement whereby the issued query is refined at every step to reduce the imprecise elements in it. Naturally, the index structure needs to accommodate the query refinement within its k-NN similarity search methods without incurring excess computational overhead. Though there are some query refinement techniques introduced to a feature based index structure as presented in [38][170][172], there were none developed for distance based index structures so far. Also, the RF method completely relied on the query model [36][37] where the refinement is done primarily on the feature-space while trying to express the semantic relationship with the feature-level weights. This might not always yield good results as discussed in the above paragraph. Thus a query refinement strategy is required, especially for a distance based index structure, that considers and refines both the contents of a multimedia objects, viz. the low-level feature content and the high-level semantic content, independently.

There are several other challenges in developing an index structure for multimedia databases such as indexing documents, which can be considered a different form of aggregated multimedia data consisting images and videos along with texts grouped together; indexing traditional alphanumeric data along with the multimedia data to be able to use a single database management system for all types of popularly used data; embedding the multidimensional index structure, with all its complex retrieval strategies, efficiently into the database kernel; developing query optimization and query execution plans which tune with the index structures, etc. But the above three challenges can be marked as the most important ones that need to be considered while developing an index structure to handle the common multimedia data robustly. In addition, there are some serious issues that are frequently faced in research related to multimedia data. Though they might not be directly related to the database management or indexing point of view, they are crucial. They are as follows:

1. Extracting appropriate low-level features from complex multimedia objects is a difficult task and falls into the content processing category. But, without good features and feature extraction techniques, the entire multimedia database management framework along with the index structures and the retrieval models would fail to produce satisfactory results. Thus, researchers dealing with multimedia database management scenarios need to make sure that they have a good repository and techniques to extract important features such as colors, textures, shapes, objects, audio, etc. correctly.
2. Developing the semantic relationships among multimedia objects require a lot of efficient preprocessing and training steps. These training and preprocessing of the data sometimes involve intense data mining or pattern recognition knowledge which a database researcher should master to be able to utilize them according to their needs.

1.2 Contributions

This dissertation addresses the above challenges and proposes a general framework that can index both images and videos and support content-based retrieval strategies for both the data types. Additionally, the framework is capable of handling the imprecise nature of such queries and the semantic gap issue. The main contributions of this dissertation are the following:

1.2.1 Generalized Multimedia Index Framework

Since there are different types of multimedia data, each differing in representation and retrieval requirements, it is necessary to device a generalized index structure capable of accommodating the different data types seamlessly. This dissertation proposes such a multidimensional index structure, called Generalized Multimedia Tree (GeM-Tree) [47], which is a flexible and robust structure indexing images as well as videos. It is the first of its kind to provide a single index structure for different media types. It is possible to extend GeM-Tree to support other genres of multimedia data such as documents as well with little modification to the basic framework. A novel data signature is used to represent the multimedia data. This signature stores the features of the individual multimedia data object, hierarchical relationships among the various units in case of videos, and other information to identify every multimedia object uniquely. Also, the signature ensures to preserve the inter-relationships between multimedia objects, for example among images and video units. A video unit, such as a shot, can be considered as a set of images, temporally related and together carrying a high-level concept. Thus, it is possible to determine the similarity between a shot and an image by comparing the features of the images, that the shot is made up of, with the image object. This type of cross-multimedia-object information retrieval is also possible in GeM-Tree. This is useful during conceptual retrievals when users wish to see all types of multimedia objects

(both images and videos) related to a particular concept and similar to one another. The proposed index structure supports almost all of the popular multimedia information retrieval strategies based on the contents including region-based information retrieval [121], CBIR with feedback [187], conceptual multimedia retrieval, unit-based video retrievals, etc. The similarity search algorithm of GeM-Tree implements a very flexible k-NN search technique to accommodate different varieties of retrievals with the same efficiency. To ensure the flexibility of the k-NN search method, the distance function used should be also flexible. Thus, Earth Mover's Distance is used which can accommodate variable length feature vectors, metric and non-metric instances of the distance function, and partial matching during similarity calculations.

1.2.2 Multimedia Similarity Queries

Basically, any index structure answers the similarity queries with either of the two approaches, the range search or the k-NN search. Due to the imprecise nature of multimedia queries, it is difficult for the user to specify his/her exact requirements while issuing a query. Hence, specifying a range/radius, required in a range search, is quite difficult. The preferable option is to let the retrieval routine of the index structure search the database for k closest objects (nearest neighbors) to the submitted query. Thus, it is crucial that a successful index structure, developed for multimedia data, supports similarity searches that are typical to multimedia data in its k-NN search routine. A unique query model for GeM-Tree is developed that considers both the low-level feature similarity as well as the high-level semantic similarity between multimedia objects independently. The semantic content and relationships of multimedia data is captured with a stochastic probabilistic network model whereby similarity between a pair of multimedia objects is determined from users' feedback over time. This semantic relationship is embedded into the similarity search routine to provide a CBIR approach which considers both the low-level feature similarity and the high-level semantic closeness individually. It is observed that

the relevance of the query results increases manifold with the proposed k-NN search of GeM-Tree in comparison to other multidimensional tree-based index structures. A novel algorithm to distribute the semantic relationship in the index tree structure is proposed which ensures that it preserves the metric properties of the index tree.

1.2.3 Multimedia Query Refinement

The issue of the imprecise nature of multimedia query is addressed by proposing a hybrid query refinement model for distance based index structures supporting multimedia data management and retrieval. Since the information content of multimedia data can be divided into the low-level and high-level categories, the query is refined from both aspects without attempting to deduce relationships between them (which is a rather erroneous process). A query expansion [37] approach is used to refine the feature space in each iteration and shift the query space to a region that has the highest chance of satisfying users' feature level information need. The users' feedback, in the form of query results marked relevant by him/her, is gathered in each iteration to form a multi-point query and similarity distance functions are redefined to accommodate such modified query structure. The high-level information need of the users are refined by dynamically updating a stochastic probabilistic semantic network, which captures the closeness or affinity between a pair of multimedia objects by keeping track of the frequency with which users have marked the pair similar in their feedback. The proposed query refinement model is the first of its kind to be embedded in a distance based index structure capable of handling multimedia data requirements. An evaluation technique is proposed to compare and evaluate the collective effect of computational cost and relevance of query result on the overall performance of a multimedia indexing and retrieval framework, mainly during the query refinement process, where numerous iterations are executed for each query. It has been seen that both the relevance of the query result as well as the computation cost required to achieve a particular relevance level should be considered while developing

a multimedia retrieval framework. One can be increased at the cost of other i.e. they are inversely proportional. Hence, the optimized performance is a balance between both where satisfactory (not perfect) query results are achieved at an acceptable computation cost. This evaluation score is called the Model Score which is a combination of F1-Score and number of distance computations.

1.2.4 Visualizing and Analyzing Multimedia Data Relationships

As it has been pointed out throughout this chapter, semantic interpretation is the most important factor that makes multimedia data stand out. Moreover, with the explosion of the popularity of social networking applications and each one of them using multimedia data heavily, management policies for multimedia data should consider the evolving relationships (which is derived from the user behaviors) between the multimedia data in such a collaborative environment. Thus, a modeling technique of the multimedia data relationship in a collaborative environment is proposed utilizing social network representation techniques. Many-a-times, such Multimedia Data Networks can be huge, which make their subsequent analysis a difficult task. To overcome this issue, a social network preview method using graph similarity is proposed which can be utilized to visualize Multimedia Data Networks as well. Additionally, such Multimedia Data Networks can be analyzed with analysis techniques used for Social Networks to understand the characteristics and behavior of the individual data element with respect to the entire network. Such holistic information about the individual data elements, with respect to the rest of the network elements, can be used to improve and customize the underlying multimedia database components. In this dissertation, we discuss the preliminary approaches by which these data characteristics can be used in the different design techniques of the proposed GeM-Tree. These concepts would be further investigated as future research as they hold tremendous potential.

1.3 Scope and Limitations

This dissertation makes some assumptions and has few limitations as follows:

1. In the proposed multidimensional indexing framework for multimedia data that supports different retrievals, there are several assumptions made. For example, it is assumed that the feature extraction codes are providing us with features that express the multimedia objects well.
2. Only soccer videos were used as the test bed for our indexing framework as the domain specific features and the semantic relationship training set for them were already available. Since the index structure can accommodate any set of low-level features and any form of high-level semantic relationship among the multimedia objects, the results produced by the index structure should be consistent for images or videos from any genres as long as the features extracted from them are meaningful.
3. Though this dissertation is limited to the organization and management of only images and videos, documents are tagged popularly to be belonging to the multimedia genre as well. A document has a different content than images or videos. Thus, to be able to build a complete index framework for a multimedia database storing all form of multimedia data, document indexing from within the same framework need to be considered in the future.
4. To index any multimedia data with the proposed index structure, the data need to be represented as a feature vector consisting of numerical values. Thus any nominal or qualitative representation need to be converted to numerical form.

1.4 Outline

The organization of this dissertation is as follows: In Chapter 2, the literature reviews are provided in the areas of multidimensional index structures, image and video feature

representations and extraction methodologies, content based image and video retrievals, query refinement designed for index frameworks, and graph similarity approaches.

Chapter 3 describes the proposed multimedia indexing framework with its various components where each component is briefly discussed.

In chapter 4, a detailed discussion of a flexible generalized indexing and retrieval approach, which can handle both images and videos from within a single framework, is presented. Extensive experimental results are provided which corroborates the goodness of the proposed framework.

Chapter 5 discusses how the proposed generalized index structure manages image retrieval in details.

In chapter 6, different video modeling approaches and how the proposed generalized index structure accommodates them along with the retrieval strategies is discussed in details.

In chapter 7, a query refinement technique for the proposed index structures is discussed which improves the query result relevance in multiple iterations without incurring excessive computation overhead. It also discusses the proposed evaluation technique to compare and evaluate the performance of different multimedia retrieval frameworks.

In chapter 8, a technique to visualize large social network graphs is proposed utilizing fewer number of nodes and edges. This chapter also discusses how the proposed approach of generating previews for social network graphs can be utilized to visualize dynamic relationships of multimedia data in a collaborative environment. It further investigates how such visualization and analysis would help in the design decisions of different multimedia database components.

Chapter 9 presents a distributed multimedia database management system over the Grid. It discusses different techniques to handle distributed content-based information retrieval across several nodes of a Grid architecture and further investigates the effects of load balancing.

Chapter 10 presents the conclusion and the future direction of this research.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, a detailed survey is presented which encompasses across the existing researches in fields that are related to the four important components of the proposed frameworks discussed in the rest of the dissertation viz. the index structure, the retrieval engine, the query refinement model and the multimedia data network (semantic modeling). They form the backbone of any successful database management system and hence they were chosen as the primary components to be designed towards the larger goal of a complete system. The multidimensional representation, the presence of semantic gap and perception subjectivity and the typical retrievals based on the contents are some of the important characteristics that make multimedia data different from traditional alpha-numeric data. The index structures designed for such data should be multidimensional in nature. Hence, in Section 2.1 the evolution of multidimensional index structures are discussed. Their performances and usability are compared and advantages and limitations are briefly mentioned. Since the job of retrieving appropriate multimedia data is a rather complex job and there are several retrieval approaches for images and videos, Section 2.2 is dedicated to discuss the different content-based image and video retrieval strategies. Though such retrievals are considered to fall within the content-processing genre and are not directly related to the database management framework, understanding them is very crucial for the design of any successful component of the framework. Section 2.3 discusses about the different existing query refinement strategies for multimedia data followed by Section 2.4 that discusses the important topic of how the query refinement models are efficiently embedded into the multidimensional index structures. Section 2.5 discusses and compares the different existing graph similarity techniques to further help choosing the appropriate direction for generating previews for large Social Network graphs and utilize the idea to generate Multimedia Data Networks for representing multimedia semantic modeling.

2.1 Multidimensional Index Structures

Multidimensional indexing can be broadly categorized into feature based or distance based techniques, each of which can be further classified as data-partitioned [19][60][98][229] or space-partitioned [148][179] based algorithms. Later, the Hybrid Tree [35] made an attempt to combine data-partitioned and space-partitioned index structures to overcome the drawbacks of each. In the feature-based approach, the data or the space partitioning is primarily done based on the values of the feature vectors along each dimension and does not depend on the distance function used in the index structure. A feature based indexing technique projects an image as a feature vector into a multidimensional feature space and indexes it. The basic feature based index structures are KDB-tree [179], R-tree [98], etc. In the distance-based approach, the partition of the data or space is done based solely on the distances of one object from one more more selected pivot points. Thus, it considers only the relative distance between image objects to organize and partition the search space rather than considering their representation in the multidimensional feature space. Some famous distance based index structures are M-Tree [60][229], vp-tree [227], etc. Though the above mentioned multidimensional index structures can accommodate the multidimensional structure of multimedia data, the semantic information attached to these data cannot be properly handled by them. Though some works like [37] and [36] tried to address the issue with a model that attempts to translate the high-level semantic relationship among image objects to feature-level equivalence, such technique is highly error prone as it is extremely difficult to map and interpret high-level semantic characteristics of image objects in terms of feature-level weights. We proposed a multidimensional index structure called GeM-Tree which is capable of embedding the high-level image object relationship without translating it, into its framework. The outcome was an increased relevance in query results without sacrificing excessive computational overhead.

The (dis)similarity between any two data objects, if computed based on a metric distance function, does not allow any correlation between feature values [78]. Now, when we

need to introduce some object level information in the index structure without intending to transform it into multidimensional vector representations, it becomes quite difficult to do so with feature-based index structures (which require all similarity computations be done on a feature-level). Thus, distance based index structures gained popularity with the introduction of the metric tree [211] which presented a more generalized approach to the similarity search paradigm. It considers only the relative distance between the image objects to organize and partition the search space rather than considering their representation in the multidimensional feature space. Some famous distance based indexes are M-Tree [60][229], vp-tree [227], etc. The vp-tree or the vantage point tree selects one data point as the vantage point and partitions the remaining data points based on their distance with it. The partition should be done as even as possible so that almost equal number of data points reside in each partition. The partitions are spherical in nature. Moreover, the data points are stored in a sorted manner which helps in answering nearest neighbor queries with ease. However, the major disadvantage is that with increased dimensionality of the data space, the number of branches of the tree that need to be searched increases which decreases its computational ability [227]. M-Tree does not have the above discussed vices and is can guarantee a balanced structure as it is built in a bottom-up manner. Every object starts as a leaf object and is later promoted to being routing objects. Each object can be promoted several times and moved up the tree by partitioning or voting. Moreover, M-Tree is capable of handling dynamic data and do not require frequent re-organization. In M-Tree, the objects are partitioned on the basis of their relative distances, measured by specific distance functions (which are metric in nature), and these objects are stored in fixed sized nodes [60][229]. The leaf nodes store all the indexed objects represented by their keys or features; whereas the internal nodes store the routing objects. Each routing object has an associated pointer which refers to the root of the sub tree called the covering tree. All objects in the covering tree are within a particular range from the routing object, called the covering radius. In addition, each

routing object is also associated with a distance to its parent object. The structures of the leaf objects are similar to the routing objects with the difference that the leaf objects do not have any covering radius and the pointer instead of storing the covering tree now stores the actual object identifier. M-Tree is not necessarily I/O bound but can also be CPU bound [229] which necessitates the requirement to lower the distance computation whenever possible.

In a space-partitioned index structure, the multidimensional feature space is recursively partitioned into disjoint subspaces represented as a hierarchical tree structure. Where as in a data-partitioned index structure, bounding regions are arranged in spatial hierarchy containing sub-region The bounding regions (also called Minimum Bounding Regions or MBRs) may overlap with each other and their shapes vary from rectangles in R-tree [98] to spheres in SS-tree [221]. In SS-tree, the MBRs can be represented by only its center and radius, thus having the obvious advantage over the R-tree which need to store the upper and lower values of the bounding rectangle for each dimension. But, the major drawback of the SS-tree is that due to the increased volume of the spheres, the overlap between the nodes increases. An example of feature based space-partitioned index structure is KDB-tree and that of data-partitioned is R-tree. Similarly, an example of a distance based space partitioned index structure is vp-tree and that of a data partitioned structure is M-Tree. The major drawback of a data-partitioned structure is that it splits a node based on all the dimensions making the fanout dependent on the dimensionality. The fanout decreases drastically as the dimension of the feature space increases making the index structure inefficient. On the other hand the main drawback of the space-partitioned structure is that the split needs to form mutually disjoint subspaces which deteriorates the node utilization. Later, the Hybrid Tree [35] made an attempt to combine these two structures and overcome the drawbacks of each. It splits the node based on a single dimension, making the fan-out independent of the dimensionality and allows overlapping whenever a clean split makes the tree cascade down, thus solving the

utilization problem. The space partitioning in each index node of a Hybrid Tree follows the same technique as KDB-tree but, to allow overlapping, the internal structure of the index nodes of the Hybrid Tree needs to be modified. A second split position is added, which represents the "left boundary of the right partition (rsp)" [35], to the already existing "right boundary of the left partition (lsp)". If lsp > rsp, a overlap between partitions is indicated. The Hybrid Tree is dynamic in nature and supports both bounding-box and distance based queries. Another advantage of Hybrid Tree is its capability to support arbitrary distance functions which makes it a popular choice for multimedia application involving relevance feedback where the distance function can vary from query to query.

A brief categorical description of the structures of few important tree-based index structures are presented below:

- K-DB-Tree [179]: In this tree structure, each of the internal node stores values to identify a section of the multidimensional data space. A set of pointers refer to their children. There is no overlap between the bounding regions of nodes and split always occurs along a single dimension. The dimension along which the split is to be done is chosen based on a round-robin fashion. The splitting is done with the aim of having a minimum number of splits and to make the splits as even as possible.

- R-Tree [98] and its variants: In these types of trees, each node contains a tuple of the form (I, ptr), where I indicates the rectangular bounding region and ptr is the address of the child node. There is a limitation on the number of tuples that each node can hold and the split policies are determined accordingly. Moreover, R-Tree is a balanced structure. There are few variants to R-Tree viz. the R+ [182] and the R*-tree. The difference between R-Tree and R+-Tree is that in R+-Tree, the bounding regions are not allowed to overlap. This is possible by letting the spatial data to split among different nodes. The absence of the overlap reduces the computation cost during search techniques as only one path needs to

be considered but adds complexity while deleting. R*-tree [17] allows overlapping but it determines the bounding regions differently from R-Tree. R*-Tree also uses forced insert which prevents splits by deleting an object of a full node and then re-inserting it.

- M-Tree [60]: It is a distance based index structure where only the distance between objects is considered as the criteria for indexing. The distance function used should be metric in nature i.e. it should follow the laws of positivity, symmetry and triangular inequality. It is a balanced structure as it it built in a bottom-up fashion. It supports both range query and k-NN search approaches. It has also exhibited good performance while handling high-dimensional data. It is a flexible structure and can support different feature representations.

- VP-Tree [227] and MVP-Tree [23]: VP-Tree is also a distance based metric tree, but differs from M-Tree in its partitioning technique. It selects a particular data point, called the vantage point, and partitions the rest of the data based on their distance with it. Similar to M-Tree, bounding regions are spherical in shape. One of usefulness of VP-Tree is its natural ability to answer nearest neighbor queries. The major drawback arises when the data used is of high dimension. The partition becomes thin and the number of branches to be searched increases. Hence, the performance is not good for high-dimensional data and are not very suitable for multimedia data in general. In order to overcome this drawback of VP-Tree, MVP-Tree was proposed which was based on the concept of using multiple vantage points instead of one. Thus, the fanout increases which reduces the search time.

2.2 Content-Based Image and Video Retrievals

There are mainly two approaches to study the retrieval strategies for multimedia objects, one being text-based and the other visual content based. The traditional approach of

text-based retrieval first annotates the video or image before retrieving them. Popular text-based retrievals are [43][44]. However, there exists two major limitations of such approach. Firstly, the huge manual effort required to achieve the annotation of every piece of multimedia object that is stored. Secondly, the perception subjectivity of the rich content of multimedia data, which varies from user to user, makes such annotation approach insufficient and erroneous. Thus, to overcome these issues, in the early 1990s, retrievals based on the content instead of the attached text gained popularity.

Feature Representation

Efficient feature extraction is the first step towards any successful content-based retrievals as the features basically represent the contents of an multimedia object formally. The features related to multimedia data can be broadly categorized into three groups, visual features, audio features and domain specific features. Where, visual and audio features encompass around features like color, texture, volume, energy, the domain specific features are application dependent and might include features like human faces and finger prints.

The color feature is the most versatile and basic feature used in the feature vectors of multimedia data. Color histograms [207] are the most commonly used color space representation. Several metrics were proposed in [150][163] etc. to measure the similarity between them. Besides color histograms, color moments [205] and color sets [193] are used as other forms of color feature representations. Texture is another important visual feature that assists in efficient multimedia retrievals. It refers to the visual patterns that have properties of homogeneity which do not result from the presence of a single color or intensity [194]. It carries information about the structural arrangement of surfaces [99]. [99] proposed the co-occurrence matrix representation of texture features and laid down the gray level spatial dependency of textures. To represent the psychological aspect of human perception, [208] developed computational approximations to visual textures

such as coarseness, contrast, directionality, lifelikeness, regularity etc. The QBIC system [71] and MARS system [112] further made improvements in this direction. In addition, approaches like wavelet transform [192], Markov random field representation [62], etc. are utilized to represent features depending upon the data set and the application at hand. The audio features that are common for multimedia data like videos, mainly fall into two categories: time domain and frequency domain [144]. The three main types of volume features that help in retrieval processes are volume, energy and flux [206].

Image Retrieval

Once the features are extracted from an image, they are represented as multidimensional feature vectors where each component of the vector specifies to what extent the particular component is present in that particular image. Subsequently, in order to determine the similarity between two different images, a distance function like Euclidean or Manhattan is utilized to calculate the (dis)similarity between the two feature vectors of the two images. There are several commercial research prototypes that had been built to support the Content-Based Image Retrieval (CBIR) like QBIC [71], RetrievalWare [65], Photobook [167], VisualSEEk [195], WebSEEk [197], Netra [151], MARS [112], etc. Among them QBIC is the first commercial CBIR system. It supports queries based on example images, user sketches, selected textures and colors etc. One of the unique features of QBIC is its ability to take into account the indexing and multidimensional tree structures. It uses R*-Tree [17] in its system and can combine text-based and content-based similarity search from within one single framework. RetrievalWare [65] applies neural nets to image retrieval approaches whereas Photobook [167] is very useful for interactive image retrieval where a human factor is introduced into the retrieval technique and multi-modal features were used to represent the information need of the user. VisualSEEk [195] and WebSEEK [197] uses visual feature and web-based search engines for images, where the spatial relationships of the image regions are considered and the features are

extracted from compressed domains [219]. The main characteristics of Netra [151] are the combined use of Gabor filter based texture analysis [6], neural net-based image thesaurus [154] and edge flow-based region segmentation [152]. MARS [112] differ from the aforementioned systems in both the research scope and the techniques utilized. It uses concepts from computer vision, database management systems and information retrievals and merge them together to use an approach where the main focus is not on finding a single best feature representation but rather on how to combine different features which can self-adjust with the changing application and user requirements.

Region-based approaches [54][121] are another technique for image retrievals where each region roughly corresponds to an object and is represented by image features local to it. The similarity calculations are applied based on individual regions or objects. Blobworld [33] is an example of one of the first region-based applications that segments the images in blobs and queries the blobs using multidimensional index structures. One major drawback of this system is its inability to handle multi-region queries. SSIMPlicity [216] uses an integrated region matching technique to overcome this problem and can handle multiple regions in the similarity queries. WALRUS [160] is another popular region based approach where regions are segmented using wavelets. The main usefulness of region-based approaches are its ability to allow partial match and concentrate on a particular area of an image rather than considering lots of useless information.

One of the major obstacles in successfully retrieving relevant multimedia objects based on the content is the presence of semantic gap between the low-level contents and the high-level semantic meaning carried by them. Thus, to bridge this gap, the user is included into the feedback loop in the form of Relevance Feedback [187]. The main goal of this approach is to adjust the query representation and the query interpretation with the help of feedbacks. Though this technique is used popularly is several image retrieval systems like MARS, WebSEEk, they have two major limitations. Firstly, it is assumed that the features-level similarity will be completely able to identify the specific semantic

relationship that the user is looking for. This is not true in most of the cases as explained with an example in Figure 1.3 and it negatively affects the query results manifold. Secondly, all the adjustments made to the feature level weights are for particular queries and all the work put into it is lost as soon as the query is over. There is no long term learning technique attached to it and a new query need to be started from the scratch.

Video Retrieval

With the popularity of video data, investigations were made to design ways in which CBIR techniques can be adapted for video data as well [8]. The first step towards video organization is to classify them into several units like scenes, shots, frames, etc. Each shot is generally represented with a representative key-frame and the complete set of key-frames for the video forms the storyboard which can then be either manually annotated for further text-based retrievals or stored in a database for subsequent content-based retrievals. There are several video retrieval systems that can accommodate content-based video retrievals. Also, since the hierarchy of classification units as depicted in Figure 3.2 for videos is an important piece of information, different levels of retrievals are also a popular variant. Some important content-based video retrieval systems are as follows:

1. VDBMS [9]: It was developed at Purdue University and it supports video content processing, representing, indexing, storage and content-based retrievals. It supports both search-by-content and search-by-streaming approaches and implements query operators, query execution engines etc. However, the indexing technique mentioned here is purely high-level and doesn't address indexing from low-level feature and storage point of view.
2. Goalgle [200]: It is a search engine specially for soccer videos. It implements an web-based interface for search and querying of soccer videos and allows users to

retrieve video segments from a collection of existing soccer matches. Further, it performs semantic event classification in multi-modal video contents.

3. IBM MARVEL [114]: It uses machine learning techniques to automatically label multimedia contents and supports query by example in both low and high-level model vector space. Multi-modal features like visual clue, sounds, speech transcripts are employed for automatic annotation and has an internal multimedia analysis and search engine for advance support.

4. CuVid [63]: It is a search platform for broadcast news videos and implements techniques such as video story segmentation, semantic concept detection, multi-modal retrieval, interactive browsing interface etc. In this system, the story segmentation algorithm utilizes the information bottleneck principals and the duplicate scenes across different news sources can be detected.

2.3 Query Refinement

Content-Based Image Retrieval (CBIR) is one of the most popular retrieval strategies for multimedia data objects as discussed in Section 2.2. Unlike traditional database queries, performance of multimedia data retrieval like Content-Based Image Retrieval (CBIR) depends largely on the efficiency with which the users' similarity concept is interpreted. Thus, the most important aspect of such similarity queries is user subjectivity which makes it difficult for the user to specify his/her exact requirements in a single iteration. The main cause for this are the gap of the semantic interpretation of an image with its feature-level representation and the starting examples incapable of capturing the complete information needs of the user. The most widely used solution to tackle this problem is query refinement in which the user provides multiple feedback to the system, which the system analyzes and refines the query by attempting to interpret the users' requirements via adjusting the query representation, feature weights etc. [170] pointed out two

major steps to refine a query viz. (1) query modification and (2) query re-weighting. Query modification refines query representation to better suit the users' information need. Query re-weighting learns the users' notion of similarity by adjusting the weights of the features in an attempt to bridge the gap between the semantic interpretation of an image with its feature-level values.

In [170], two query modification techniques were proposed viz. (1) Query Point Movement and (2) Query Expansion. Query Point Movement (QPM) allows only a single query object per feature space. When in each iteration of the Relevance Feedback method, the user marks several objects as relevant, the weighted centroid of the relevant image objects is used as the new query. The weights are associated depending upon the relevance level (rank) as attached by the user. The weighted centroid C is defined as:

$$C[j] = \frac{\sum_{i=1}^{n} w_i E_i[j]}{\sum_{i=1}^{n} w_i} \tag{2.1}$$

Where, $E_i[j]$ is the feature vector of image i along the j^{th} dimension and w_i is the weight associated with the image i.

The similarity distance functions are modified with the new query point represented as the centroid of multiple relevant image points. The above QPM technique is utilized in [165] and in [115]. Its goal is to choose a single point and re-weigh its dimension such that the sum of its distance from the relevant points become minimum [115]. But, the QPM method results in some information loss as the characteristics of each relevant image is lost and their collective representation is treated as the refined query.

On the other hand in the query expansion approach, multiple objects marked relevant in a particular iteration are all considered in the refined query. Such queries are also called Multi-point Queries. In this method, clustering of the relevant points may be done and the centroids of the clusters can be used as the representative query points. The representative points are used to form the new query. The weights get added to the multi-point query and the distance function of the multi-point query is the summation of the weighted distances of each representative query from an image object in the feature

space. [172] performed extensive experiments over large image collections and concluded that Query Expansion approach outperforms Query Point Movement approach in retrieval results based on precision and recall measures. Another important advantage of the query point expansion technique pointed out in [170] is its enhanced general applicability as compared to the query point movement approach. Query expansion is usable even when the feature space is not defined but the metric space corresponding to the particular feature space is known. Later, [145] pointed out that the Query Point Movement technique has some additional limitation like having local maximum traps which results in poor improvement of query results for refinement iterations. Thus, [145] proposed four target search techniques viz. Naive Random Scan, Local Neighboring Movement, Neighboring Divide and Conquer Method and Global Divide and Conquer Method to improve it.

2.4 Query Refinement in Multidimensional Index Structures

In [170], a feature-based multidimensional index structure, called Feature Index or F-Index, was used to demonstrate the technique by which multi-point refined queries were supported during retrievals. The similarity queries were executed using a k-nearest neighbor algorithm which accesses nodes in an increasing order of their distances (similarity measurement) from the query point. A priority queue is implemented for the ordered traversal over the index structure. There are two approaches proposed by [170] and [36] to implement similarity searches with the refined queries since there are two types of query modification techniques. One approach of evaluating the refined queries is to retrieve nearest neighbors of a single point C (the centroid as defined in equation 9.1) and still guarantee that the set of answers S returned are the k-nearest neighbors of M (multipoint query) i.e. $D(M, S_i) \leq D(M, T)$ for any $S_i \in S, T \notin S$. An alternative approach is to retrieve results based on the distance from all the points in the multi-point

query instead of the centroid only i.e use the query expansion methodology. The distance between any intermediate node (N) and the multi-point refined query (M) was defined as:

$$MINDIST(M, N) = \sum_{i=1}^{n} w_i D(P_i, NP(P_i, N)) \tag{2.2}$$

Where,

$$NP(P_i, N)[j] = \begin{cases} l_i & \text{if } P_i[j] < l_j \\ h_j & \text{if } P_i[j] > h_j \\ P_i[j] & \text{otherwise} \end{cases}$$

$NP[j]$ denotes the position of NP along j^{th} dimension of the feature space R_F and P_i is the i^{th} point of the multi-point query.

The distance between a leaf node and the multi-point query can be defined in a straightforward manner as depicted in [36]:

$$D(M, N) = \sum_{i=1}^{n} w_i D_F(P_i, N) \tag{2.3}$$

To achieve the query re-weighting aspect of query refinement for F-Index in [170], one need to capture visual features that best describes the users' concept of high-level similarity and attach/modify weights to these features to get refined query results close to user perception. Extensive research was performed in this field to better capture users' perception and translate the high-level semantic interpretation of multimedia data to a feature-level model like in [110] and [186], where an interactive mechanism was devised to include a human in the retrieval loop, in [64], where an interactive region segmentation was employed etc. Later, [187] proposed another technique called Relevance Feedback [RF] in which humans and computers interact to refine high level information to low-level representations. It is the process to automatically adjust and modify an existing query by using the feedback of the user about the relevance of the image objects retrieved in a previous iteration. In the relevance feedback based approach, as in [188], the burden of specifying the weights is removed from the user and the weights corresponding to the features (inter feature or intra-feature) are dynamically updated to represent the high-level

concepts and perception subjectivity. Later, the RF method was improved and modified by using Support Vector Machine (SVM) as in [28]. Several classification-based RFs were also proposed [109] to overcome some of the problems of traditional RF like the heuristic nature of it, not able to consider the negative feedbacks etc. Later [143] proposed a modified SVM based RF technique to overcome the problem of SVM classifier's unstable behavior for small training sets and the problem of the kernel machine. All these research proceeds towards improving and modifying the RF method and aims to enhance the procedure of determining and attaching weights to low-level features to better capture user's similarity concepts. Any one of them, depending upon the available data and user requirement, can be utilized as the query re-weighting approach in the query refinement techniques. The major limitation of the above query re-weighting technique is that if the semantic interpretation of an image cannot be represented completely in terms of feature-level weights, the above discussed technique cannot generate satisfactory results. Also, the above query-re-weighting techniques cannot be used for distance-based index structures as the intra-feature weighting strategies make the distance functions arbitrary.

2.5 Graph Similarity

In this section, related works on the topics of Graph Similarity and Matching are discussed. There are several approaches to determine the similarity between a pair of graphical structures. The first law of thermodynamics and heat kernel is used in [128] by transforming the problem to finding the difference in the thermal energy between two graph structures. The technique utilizes a normalized graph Laplace and relates it to the edge-weights. Thus for un-weighted graphs, this approach cannot be used. Using the concept of maximum subgraph is another technique to compute the similarity between a graph pair. It is derived from the concept of graph isomorphism and is largely used in chemical and biological fields. The maximum common subgraph can be classified as

connected or disconnected [177]. Usually, the process of determining a maximum common subgraph between a graph pair is a NP-complete problem [123]. Thus, attempts have been made to work around this problem of NP-completeness by designing algorithms which would result in approximation of the final maximum common subgraph rather than attempting to find an exact subgraph. The exact matching algorithms (NP-complete) can be further categorized as *maximum clique-based algorithms* [34][140], *backtracking algorithms* [155][12] and *dynamic programming* [4]. Similarly, the approximate algorithms (with a lower computation cost) can be categorized as *genetic algorithms* [212][91], *combinatorial optimization* [11], etc. The main disadvantage of this genre of graph similarity calculation is that the exact guaranteed match is NP-complete and the approximate matching approaches do not guarantee that the approximated subgraph will be close in size or structure to the optimum representation. A spectral method of Graph Matching is proposed by [180] where the graph adjacency matrix represents the transition probability of a Markov Chain. The main challenge in using this method is that it depends solely on the structural information of a graph and there is no way to incorporate semantic information in the form of node/edge labels. A graph similarity determination technique is proposed in [180] using maximum common edge subgraphs. In this method too, edges or vertices need to be labeled and thus cannot be utilized for un-labeled or unweighted graphs. In [233], an approach of conceptual graph matching is presented which searches a conceptual graph by considering the similarity between concepts and relations in them. It uses a distance function and assigns a value (*milestone*) to every node in a conceptual hierarchy. Such approaches alone will not be beneficial in graph similarity calculations as they do not consider the graph structures. But, if combined with efficient structural similarity computation methods can yield desired results in identifying the overall similarity between graphs with both structural as well as semantic (conceptual) information in them. [61] presents an Ontology Similarity Measure for conceptual graphs while considering structural information as well. It takes into consideration both the re-

lation adjacency in the bipartite graphs as well as the relation hierarchy. The proposed method is yet to be utilized in practice and might prove to be useful for future graph similarity determinations. However, for the above two discussed methods, an explicit conceptual hierarchy graph, developed from the semantic information, need to be determined. Such clear definition of conceptual relations may not be available for most graph structures and hence using these methods might be a challenge. Thus considering all the different methods for computing similarity between a pair of graph, the node-edge coupled method [228] combined with the proposed semantic similarity generation matrix was found to be most appropriate for social network graphs while considering both the quality of the generated results as well as the computation. The main drawbacks of other existing approaches can be summarized as: (1) a large family of the existing methods are NP-complete and hence were not appropriate for our purpose where graph sizes are generally very large, and (2) many of them were not generic enough and needed compulsory additional data such as edge/vertices weights and labels. The approach proposed in [228] was found flexible enough and computationally more economic than most of these approaches and hence was chosen for the similarity computation of social network graphs.

CHAPTER 3

OVERVIEW OF THE FRAMEWORK

A traditional database management system can be considered to be composed of two major parts: a data organization component and a data retrieval component. Other components such as a query engine or a query interface eventually assists either the data organization or the data retrieval component. Multimedia data has an additional property, the semantic information attached to them, which is absent in the traditional alpha numeric data. Hence, a third component, the semantic relationship component, should be included into the multimedia database management framework as well to generate semantically relevant retrieval results. An index structure is the backbone and the first step towards the satisfactory development of any database management system as it acts as a bridge between the data organization and the retrieval component. This dissertation is dedicated towards developing the indexing framework that would aid in better organization and retrieval of different types of multimedia data. The complete framework is depicted in Figure 3.1 which lays down the overview of the ultimate goal and motivation of this research which is developing a *dataspace* management system that can organize and retrieve different genres of data, from recent multimedia to traditional alpha numeric, with the same ease from within the same platform while providing a transparent interface to the user. The components and sub-components only related to multimedia data indexing and retrievals are covered in this dissertation while other design aspects included in the framework are to be addressed in the future.

As presented in Figure 3.1, the three major components of the multimedia database management framework are: (i) Multimedia Data Organization Component, (ii) Multimedia Data Retrieval Component and (iii) Semantic Relationship Component. The multimedia data is divided into three categories viz. images, videos and documents. Each data type has a specific composition and representation. Thus, each of the above specified components should support all these three types of multimedia data seamlessly.

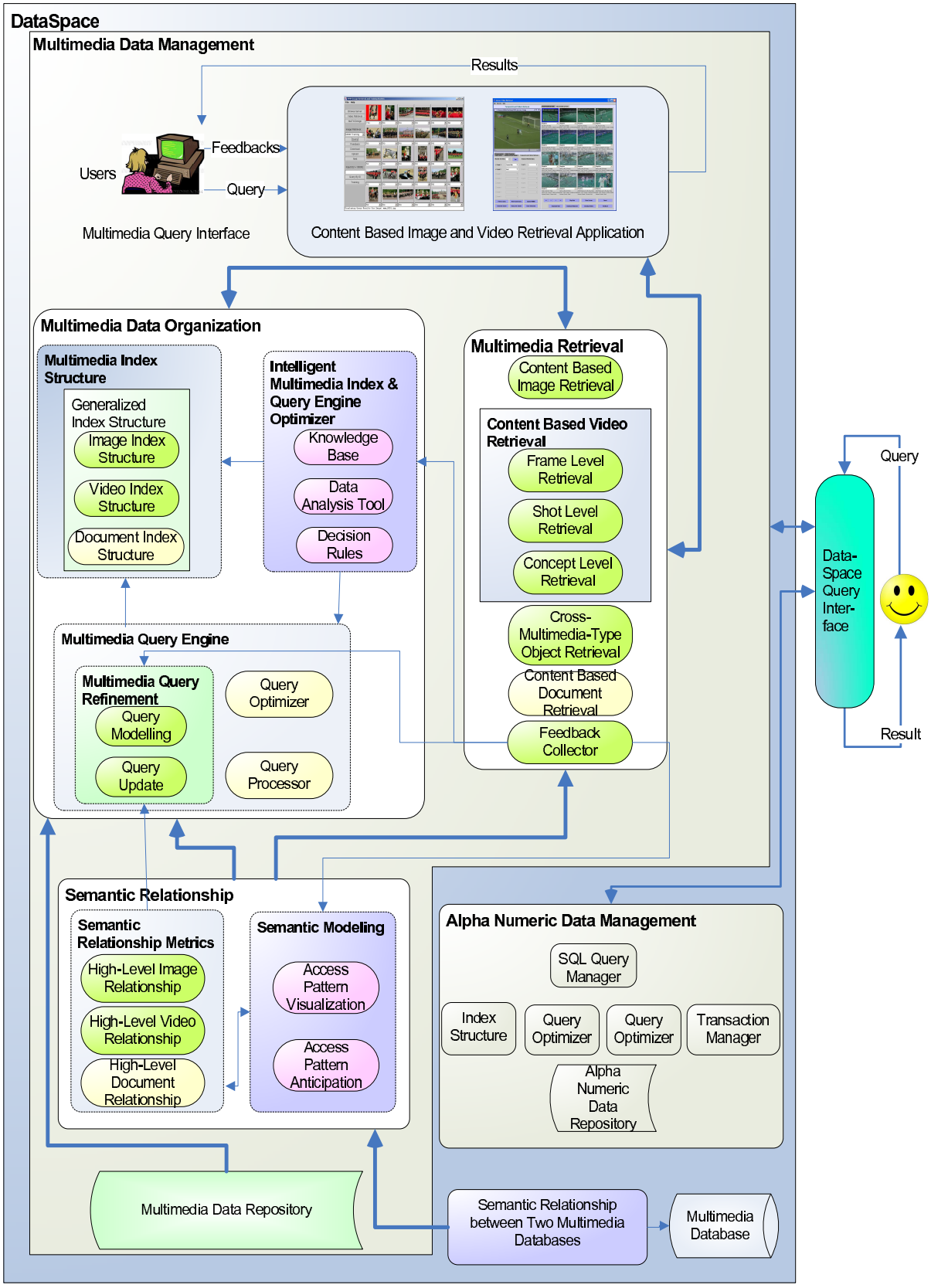


Figure 3.1: Overview of the proposed framework.

This dissertation covers images and videos and leaves the document handling part as a future research agenda.

3.1 Multimedia Data Organization

The multimedia data organization component has three sub-parts viz. an index structure, a query engine and an intelligent index and query engine optimizer.

3.1.1 Multimedia Index Structure

Multimedia data such as images and videos are represented as multidimensional feature vectors where each feature component represents the multimedia data object in terms of color, texture, shape descriptors, audio features, etc. Videos are further classified into a hierarchical containment relationship as depicted in Figure 3.2 where each video is represented as a collection of shots. Further each shot is expressed in terms of feature values of consecutive frames, which are combined (their average, maximum, minimum etc. are considered).

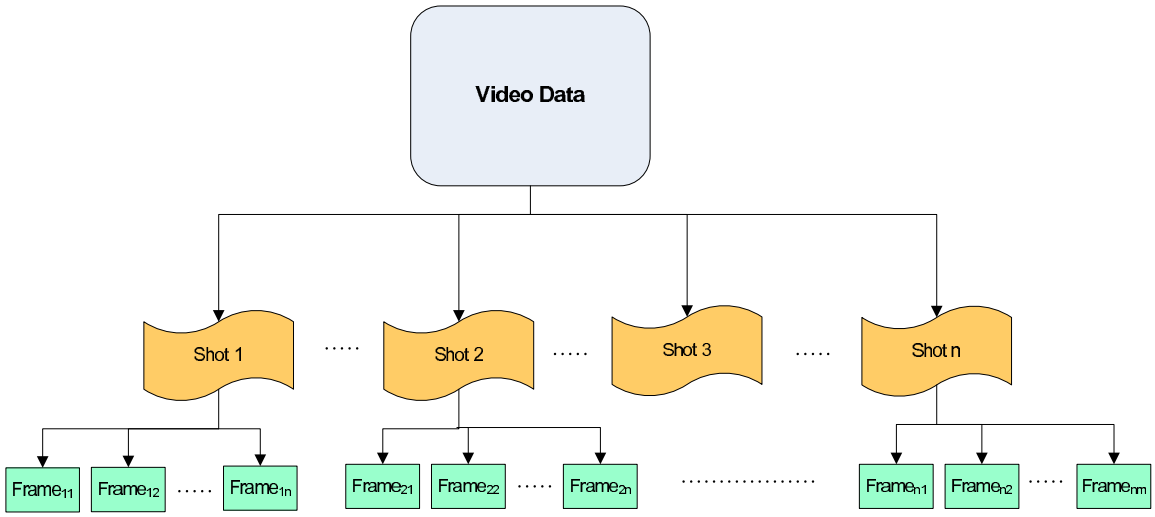


Figure 3.2: General video structure.

In this research, shot is used as the smallest unit of video representation and shot detection algorithms like [52] is utilized to represent a video as a collection of shots.

Traditional index structures meant for alpha numeric data like B-Tree [14] cannot accommodate such data type and multidimensional index structures are utilized to handle them. But the existing multidimensional index structures like [35][179] are not sufficient to be used in a multimedia database management system because they are not designed to support the hierarchical structure of video data, they do not embed the high-level semantic relationship in the similarity searches efficiently and none of them can be used to index both images and videos together from within one single framework. The third factor is particularly crucial as having separate index structures for different data types inside one single database management framework is inefficient and error-prone. There are several other components which are linked to the index structure and are optimized and tuned based on it. Thus, if there are more than one index structure, there might be conflicting issues while trying to tune and optimize the related components with respect to each of the index structures utilized. In order to overcome these existing issues, a generalized index structure, the GeM-Tree (Generalized Multimedia Tree) [47] is proposed in this dissertation, that can handle both images and videos efficiently and as per our preliminary investigation has great potential to index documents as well. The framework also links the index structure to another important component of the multimedia database management framework, the Semantic Relationship component, during similarity searches to generate query results close to the human perception of similarity.

3.1.2 Multimedia Query Engine

The multimedia query engine is an integral part of the multimedia data organization component. When an user issues a query to the multimedia data management interface, it is submitted to the query engine which formulates it, processes it, optimizes it, and submits it to the index structure. Once, the index structure searches the data repository with the help of the retrieval engine using the query, it produces the results to the user to collect the feedback regarding the relevance of the results. The feedback thus collected

is further utilized by the multimedia query refinement component to refine and improve the query representation with the help of the query modeling and the query update processes. This dissertation investigates the Multimedia Query Refinement process and leaves the development of a Multimedia Query Optimizer and a Query Processor for future investigation.

As discussed in Chapter 1, multimedia queries are imprecise in nature. Thus the proposed Multimedia Query Refinement method helps to capture the users' information need from multiple feedback by modifying the queries and updating the matrices that capture the semantic relationships between the multimedia objects, thus attempting to capture the users' information need precisely. This approach comprises of two major methods viz. the Query Modeling method and the Query Update method. The proposed query refinement model is mainly designed to be embedded into a distance-based index structure as there are few models [36][37] for feature based index structures, but none has been developed for distance based index structures so far. It can be safely argued that distance-based index structures are equally important as feature-based index structures. Thus the lack of query refinement methodologies for distance-based index structures limits their utilization for practical purposes. Apart from serving as a query refinement component for GeM-Tree, this query refinement methodology can be also used for other distance-based index structures as well.

3.1.3 Intelligent Multimedia Index and Query Engine Optimizer

In order to make the index structure and query engine able to optimize and adjust themselves according to the changing data types and access patterns, it is planned to introduce a novel sub-component, the intelligent multimedia index and query engine optimizer. It has three major parts viz. a knowledge base, a data analysis tool and a

repository of decision rules. The relevant historic information about the access patterns and user feedbacks will be stored in the knowledge base and it will use a collection of data mining tools like association rule mining, classification techniques and clustering methodologies to analyze these historic data. These data will serve as the training data and once there are association and classification rules, they will be used to tune the index structure and the query engine.

3.2 Multimedia Retrieval

The Multimedia Retrieval Engine takes care of all the data access needs of the user. Since, the multimedia data can be of multiple types, each having different characteristics and retrieval requirements, the retrieval engine should accommodate multiple retrieval approaches, with equal efficiency. The Multimedia Retrieval Engine is intrinsically related to the Multimedia Organization module and the index structure must support all the different retrieval approaches in its search methods. This dissertation focuses on mainly three different retrieval approaches viz. content-based image retrieval; content based video retrieval that further has sub-categories of unit-level retrievals like frame-level video retrieval, shot-level video retrieval and video-concept level retrieval; and mixed multimedia data type retrieval. Another retrieval genre, the content based document retrieval, is planned to be pursued in the future.

3.2.1 Image Retrieval

A content based retrieval approach, that considers the similarity between both the low-level features as well as the high-level semantic contents of images, is implemented in this framework. The GeM-Tree [47] embeds this content-based image retrieval technique in its k-NN and range search routines. To capture the high-level semantic relationships between the submitted query image and the stored images, a stochastic construct called

Markov Model Mediator [191] is utilized during the retrievals. The k-NN search algorithm, implementing the content based image retrieval, recursively searches the index tree from the root, via the intermediate nodes till the leaf nodes. It maintains a priority queue to store the candidate nodes that might contain the result images. It traverses the index tree and checks each intermediate node in terms of both low-level feature distance and the high-level semantic closeness with the query object. If the k^{th} node in the priority queue already has an object with a distance lesser and a semantic closeness higher than the node under examination, it is discarded or else it replaces the k^{th} node in the priority queue. The priority queue is re-organized so that the nodes in it are arranged in an order with the node of lowest overall similarity score placed as the k^{th} element. The candidate nodes in the priority queue are examined in subsequent rounds. The same approach is taken for leaf nodes with the only difference of directly adding the examined leaf node in the result set (also a priority queue) if it satisfies the required similarity conditions. The approach of separately considering both the high-level and low-level similarity measures without violating the metric property of the search space ensures high relevance of query results with acceptable computation cost.

3.2.2 Video Retrieval

A similar k-NN based retrieval algorithm is proposed for video data as well, which is supported by GeM-Tree. The main principle remains the same i.e. searching video objects similar both in terms of low-level features as well as high-level semantic content to the submitted query. But, since a video object can be classified into different units like frames, shots and video concepts, thus the retrieval algorithms should accommodate different unit retrievals. Hence, there are three sub-categories of video retrievals i.e. frame-level retrieval, shot-level retrieval and video concept-level or entire video-level retrieval. Additionally, since a hierarchical semantic relationship metric is maintained in the semantic relationship module, cross-unit similarity can be evaluated as well. For

example, if the submitted query is a video shot but one wishes to retrieve complete videos similar to the concept/feature represented in the submitted query video shot, it can be achieved by traversing up the hierarchical semantic model and finding similar videos corresponding to the video that the shot belongs to. Similarly, other inter unit similarity can be calculated indirectly. In GeM-Tree, such logical hierarchy is maintained with the help of the unique data signature and a single tree structure is enough to accommodate and retrieve all the units efficiently.

3.2.3 Mixed Multimedia Data Type Retrieval

This approach of multimedia retrieval is mainly utilized during conceptual queries where the user wishes to extract multimedia objects, irrespective of data types, that are similar to a submitted query. For example, an user might submit a query that comprises of a shot of a soccer scene. If he/she wishes to retrieve all types of multimedia objects (both images and videos) that are related to soccer or depicts some scenes related to soccer, this type of query needs to be initiated. Such queries can turn out to be very useful for search engine type applications and for scenarios when the user doesn't have a clear idea of what media-type he/she wants but starts off with some random query and refines it in subsequent iterations. Such mixed multimedia data type retrievals can be implemented in the k-NN search routines of GeM-Tree with the help of the multimedia data signature used. Essentially, this dissertation deals with images and videos only where each video unit is represented as a collection of two feature sub-sets. The first subsets represents the characteristics of the video unit with respect to features that are common to both images and videos such as color, texture, etc. The second part comprises of features related to videos only such as temporal relationship, audio features, etc. For images, this second part is absent as they do not have any temporal or audio information attached to them. Thus, while calculating the inter multimedia object similarity, the parts of the multimedia object feature set that have the same feature types are compared. A separate

concept level semantic relationship metric is used to determine the high-level similarity between two different types of multimedia objects.

3.3 Semantic Relationship

In this dissertation, a stochastic model is chosen to capture the semantic relationships between all types of multimedia objects, called the Markov Model Mediator (MMM) [191]. The model collects the user access patterns during the feedback steps over time and generates a similarity matrix that basically provides how frequently two images/videos are marked similar to one another in the past. This value along with the relative access frequencies of other pairs are utilized as the primary semantic relationship determining factor in this research.

3.3.1 Semantic Relationship Matrices

As discussed above, the MMM approach is used to generate the semantic relationship matrices which are called *affinity relationships*. They are usually square matrices where each cell represents a score depicting how similar the pair of multimedia objects, identified by the corresponding row and column of the cell, are to each other. The process follows a continuous learning approach. After each query result is presented to the user, the feedback from the user is utilized to update the affinity relationship matrix dynamically. Thus, the matrix is never biased on any single users' feedback. For videos, a hierarchical version of the above concept called the Hierarchical Markov Model Mediator (HMMM) [191] is used which preserves the hierarchical relationship between the different units that the video is made up of. These semantic relationships are used in the similarity search methods of the proposed index structure and they help to increase the relevance of query results in all the above discussed types of retrievals.

3.3.2 Semantic Modeling

In order to better visualize and analyze the semantic relationships between multimedia objects, a graph based social network representation approach is utilized. This would help the database designer to get a clear relationship map and identify pivotal data objects in the network. Also, it would help to refine and improve the index structure and the query refinement modules immensely as it would assist in understanding how the different functionalities of these components, apparently affecting few immediate multimedia objects, can have indirect but important effects on the entire network. Further, graph similarity methods [228] can be applied on these semantic similarity networks to determine the relationships between two semantic models of two different multimedia databases. Determination of such relationships or similarity would help to design or optimize the various components of one multimedia database based on the successful design decisions taken in a related one. Another important aspect of developing such semantic modeling based on the user behavior is its ability to represent the data relationships in a collaborative environment where multiple users are handling the same data corpus. Such understanding would in turn help to design multimedia database management frameworks for applications using collaborative approaches. Practically, all social network frameworks can be considered as collaborative applications. Thus such semantic modeling can be utilized in developing the database components for such applications, where multimedia data is a common medium of communication.

CHAPTER 4

GEM-TREE: A GENERALIZED MULTIDIMENSIONAL INDEX STRUCTURE SUPPORTING IMAGE AND VIDEO RETRIEVAL

As it is said, "a picture speaks a thousand words", the expressiveness of multimedia data and the varied information that it can carry has increased its popularity manifold in the recent years. With the increased use of search technology, paradigms involving multimedia search based on their content instead of the attached annotations are emerging as the future research direction. Thus, retrieval approaches such as Content-Based Information Retrieval (CBIR) have gained importance. Efficient management of multimedia data and supporting the different retrieval strategies seamlessly are crucial for the success of the above research direction and usability of the multimedia data in a commercial scale. As pointed out in Section 1.1, there are several issues with designing and implementing a robust multimedia data management framework and laying down a suitable index structure is one of the pivotal and crucial parts.

In this chapter, a generalized multidimensional index structure, called GeM-Tree (Generalized Multimedia Tree) is discussed to manage multimedia data such as images and videos seamlessly from within one single framework. It is necessary to have a generalized index structure which can accommodate different types of multimedia data along with their different retrieval strategies because having separate index structures for different types of multimedia data poses two major problems. First, integrating an index structure into a database kernel needs the modification of the Query Optimizer, Query Processor, SQL Compiler/Interpreter and other database components as illustrated in Figure 1.1. to tune the performance of these components with the corresponding index structure to be embedded. The process itself is complicated, tricky and time consuming [103][175]. Thus, modifying the database kernel components to support multiple different index structures and access methods is not a welcoming idea and might have conflict issues regarding performance (for example, modifying a Query Optimizer for one particular

index type for a certain multimedia object might have a degrading effect on the performance of another index structure for a different multimedia data object). Second, for some applications (like multimedia concept search) where the similarity between videos and images needs to be determined to answer queries, having separate index structures is inconvenient and inefficient. The next two chapters are extensions of the current chapter where retrievals of images and videos, as handled by the proposed index structure, are discussed in details.

GeM-Tree uses Earth Movers Distance (EMD) [185] as the distance function to calculate the (dis)similarity among the multimedia data objects in a metric space. To capture and utilize the high-level semantic relationships among the multimedia data objects and to introduce the relationships between the different levels of video units, a probabilistic mathematical construct called the Hierarchical Markov Model Mediator [51] is used. Further, a flexible k-NN based similarity search algorithm is introduced that can support different approaches of content-based image and video retrievals while considering both the high-level semantic relationships and the low-level feature similarity with equal efficiency. Though EMD was used as a distance function in VP-tree [224] to develop VP-EMD tree [225], VP-EMD tree does not have the capability to index videos. It was meant to serve as an index structure supporting only content-based image retrieval for feature sets with variable lengths. Also, VP-Tree is not a balanced structure as it is built in a top-down fashion. GeM-Tree, on the other hand, is a balanced structure as it is built from the bottom following an approach similar to M-Tree [60].

4.1 GeM-Tree

As discussed, EMD is used as the distance function to build GeM-Tree by utilizing the (dis)similarity between the multimedia data objects in a R^q metric space, where q is the number of features used to represent a multimedia object (an image or any video unit). Euclidean Distance function (L_2) is chosen as the ground distance to

keep EMD metric. The main benefits of utilizing EMD as a distance function are its capability of calculating the (dis)similarity between variable sized distributions [185] allowing for partial matches, and its ability to better match perceptual (dis)similarity [185] by providing the flexibility to use different approaches of content based retrieval such as region-based [13] methods. The EMD has been used to measure image similarity with respect to color and texture [13][185], but to the best of our knowledge, EMD was not previously utilized to calculate video similarities or to determine the relationship between two different types of multimedia objects.

4.2 Earth Movers Distance

The Earth Movers Distance (EMD) [185] is a general and flexible distance function to compute the similarity between two distributions and is based on derivation of the minimum cost that is incurred to transform one distribution to another. It was derived from the transportation problem viz. the Monge-Kantorovich Problem [174] which determines the minimum cost of transporting goods from a set of m sources or suppliers to a set of n destinations or demander. To use a EMD function, a multimedia object is represented as a signature or a finite distribution x as follows:

$$x = (x_1, w_1), (x_2, w_2), ..., (x_n, w_m) \equiv (X, w)\epsilon D^{K,m} \tag{4.1}$$

Where,

$$X = [x_1, x_2, ..., x_m]\epsilon R^{k\times m} \tag{4.2}$$

and m is the number of points.

Given two distributions $x = (X,w)\ \epsilon\ D^{K,m}$ and $y = (Y,u)\ \epsilon\ D^{K,n}$, a flow between x and y is a matrix defined as:

$$F = (f_{ij})\epsilon R^{m\times n}. \tag{4.3}$$

The main approach is to find a flow between x and y that minimizes the overall cost of the work done in order to displace values between x and y as presented in Equation 4.4.

$$Work(x, y, F) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij} \tag{4.4}$$

Four conditions need to be satisfied by the f in Equation 4.4 as presented in Equation 4.5 to 4.8.

$$f_{ij} \geq 0 \tag{4.5}$$

$$\sum_{j=1}^{n} f_{ij} \leq w_x \tag{4.6}$$

$$\sum_{i=1}^{m} f_{ij} \leq w_y \tag{4.7}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = min(\sum_{i=1}^{m} w_x, \sum_{j=1}^{n} w_y) \tag{4.8}$$

Where, w_x and w_y are the weights of the distributions of x and y, respectively, $1 \leq i \leq m$ and $1 \leq j \leq n$. An optimal flow F is calculated using the solution technique of the transportation problem [174] and EMD is defined as:

$$EMD(x, y) = \frac{(\sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij})}{(\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij})} \tag{4.9}$$

Where, $d_{i,j}$ is the ground distance expressed as:

$$d_{ij} = d(x_i, y_i) \tag{4.10}$$

EMD is a metric i.e., it follows the laws of symmetry, positivity, and triangular inequality when the total weights of the distributions are equal i.e. $\sum_{i=1}^{m} w_i = \sum_{j=1}^{n} w_j$ and the ground distance d_{ij} is a metric [185].

4.2.1 Fixed-Length Multimedia Data Signatures

For traditional content-based retrieval, where the entire multimedia object (an image or a video frame) is treated as a single class/region, the fixed length multimedia data signature is used for representation. Both visual and audio features are used in the feature distributions. Techniques described in [52] are applied for shot boundary detection and grouping a bunch of consecutive video frames as a shot. It is worth pointing out here that any feature set (visual or audio) can be represented collectively with F_A, F_B and F_C (as presented in Equation 4.11, 4.12, and 4.13) without any loss of generality. It completely depends upon the application, data types and the user preference. For example, while representing videos, if the retrieval application addresses frames as the lowest level of video units, F_A stores the low-level features of individual frames. However, if the retrieval applications do not require frame-level information, F_A can have all 'zero' values. In this paper, we use 'frame' as the lowest level of video unit. For the rest of this section, we discuss the signature parameters particularly with respect to the retrieval application and dataset used in our system.

For the sake of clarity, the feature distributions for each multimedia data object are divided into three sub-distributions as follows.

$$F_A = \{x_1, x_2, ..., x_i\}. \tag{4.11}$$

$$F_B = \{y_1, y_2, ..., y_j\}. \tag{4.12}$$

$$F_C = \{object_{id}, v_{id}, s_{id}\}. \tag{4.13}$$

The feature vector representing the distribution of each multimedia data object is a union of the three sub-distributions and is represented as given in Equation 4.14.

$$F = \{(F_A \cup F_B \cup F_C), F_{wt}\}, \tag{4.14}$$

Where F_A is the color and texture features for each image and each frame in a video shot, x_i is the value of the color/texture feature normalized in a 0-1 scale, and F_B has the visual and audio features for a video shot.

Examples of the visual features for video shots are the average percentage of the changed pixels between consecutive frames in a shot, the mean value of the frame-to-frame histogram difference in a shot, etc. [52]. For audio features, a sampling frequency of $16,000$ H_z is used and the audio track is divided into clips. Each audio feature is calculated at the frame-level and synchronized with the visual features. Audio features used in this framework are divided into three basic types: volume, energy, and spectrum flux. F_C captures two important information about the multimedia data: the unique identification number and the hierarchical relationship (if any). The hierarchical relationship is only meant for video data where it exists between the different video units. v_{id} stores the $object_{id}$ of the video data object of which a particular frame or a shot is a part, and s_{id} stores the $object_{id}$ of the shot of which a frame is a part. For images and entire video objects, both the fields are set to 'zero'. For video shots, v_{id} is set to the $object_{id}$ of the video object to which the video shot belongs; whereas the s_{id} field is set to 'zero'. Similarly, for video frames, v_{id} is set to the $object_{id}$ of the video object, and s_{id} is set to the $object_{id}$ of the video shot to which the particular video frame belongs. F_{wt} is a set of cardinality 1 as only one feature/distribution class is utilized in the fixed-length representation, and the value is set to 1 for all the Data Signatures.

4.2.2 Variable-Length Multimedia Data Signatures

For variable-length Multimedia Data Signatures, each image/video frame is modeled as a Gaussian Mixture distribution in the feature space. Each image/video frame is divided into homogeneous regions, and regions are represented by Gaussian Mixtures [168]. In our proposed framework, the RGB color space is used, but any color feature space can be utilized without any loss of generality. Moreover, feature spaces other than colors, such as textures, can be utilized as well. The Expectation Maximization (EM) algorithm [18] is used to determine the maximum likelihood parameters of a mixture of k Gaussian in a 3-D feature space (RGB). EM algorithm is generally used when there is a missing data.

In this case, the missing data is the region where each pixel in the image/video frame belongs. To choose k (i.e., the number of clusters/regions in each image/video cluster), a goodness of fit measure is considered. There are three popular goodness of fit measures: (i) The Akaike Information Criterion [3], (ii) The Bayes Information Criterion [190], and (iii) Minimum Description Length (MDL) [178]. Here, the Akaike Information Criterion is adopted to get the k value that will divide the image/video frames into an optimum number of regions.

To set the weight F_{wt} for each region, the fraction of image pixels that belong to the particular cluster/region is determined. Thus, the summation of F_{wt} for each image/video shot is always equal to one. The mean of each cluster/region is used to represent the feature vector for that particular region of the image/video frame. For the dataset we used, k ranges from 2 to 5. Thus, each region of an image/video frame is represented as $F_{A_{cluster2}}$=(0.35,0.45,0.39), $F_{A_{cluster3}}$=(0.54,0.62,0.23), etc. The feature distribution is thus F_A=$\cup_{i=2}^{5}(F_{A_{clusteri}})$. For videos, the hierarchical relations and the weights are represented as F_B=$\{0,0,....0\}$, F_C=$\{5,0,0\}$, and F_{wt}=$\{0.23,0.37,0.25,0.16\}$, respectively. Figure 4.1 represents the different stages of the clustering method using Gaussian Mixture Models with Expectation Maximization Algorithm. This example uses k=5 (i.e., the number of

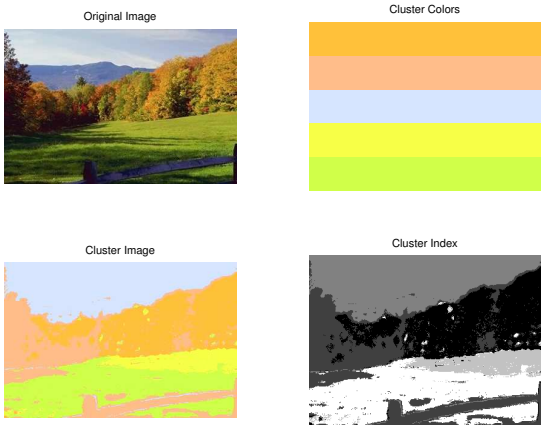


Figure 4.1: Clustering using gaussian mixture models with k=5.

clusters/regions is 5). The top-leftmost is the original picture. The top-right figure shows the dominant colors or the cluster colors. The bottom-left figure represents the clustered image and the bottom-right depicts the cluster index. Thus, the image is represented into five clusters and the mean color values of the pixels present in each cluster forms the feature distributions of the image. It should be noted that videos are collections of frames which can be treated as images and clustered in the same way to generate the variable-length distributions for each video frame.

4.2.3 Node Structures of GeM-Tree

GeM-Tree has two main node types, the leaf nodes storing the actual indexed multimedia data objects and the intermediate nodes which maintain the sub-tree structure within a tree. Further, depending upon the signature of the particular multimedia data object that the intermediate or the leaf nodes are storing, they can be subdivided into *image_intermediate* and *image_leaf nodes*, *frame_intermediate* and *frame_leaf nodes*, *shot_intermediate* and *shot_leaf* nodes, and *video_intermediate* and *video_leaf* nodes respectively. Each intermediate node contains the pointer to the sub-tree it points to; a covering radius, which is the distance between the root of the sub-tree under consideration and its farthest child and four place holders for the promoted high-level similarity. These four place holders for the high-level semantic relationship value hold the values for the four possible types of multimedia data object viz. an image, a frame, a shot or a video. These values change with each query issued but the covering radius and the pointer to the sub-tree remains the same after the GeM-Tree is built until the structure of the tree is modified via an insertion or deletion operation. Each leaf node contains the $object_{id}$ of the indexed database object along with the storage information of the particular object.

4.2.4 Node Insertion

To insert a node into a GeM-Tree, the tree is recursively traversed until a candidate leaf node is identified. A particular sub-tree leading to the leaf node is chosen by selecting an intermediate node for which there is no (as in Equation 4.15) or minimum increase (as in Equation 4.16) in the covering radius.

$$d(O_r, O_n) \leq r(O_r) \tag{4.15}$$

$$d(O_r, O_n) - r(O_r) \tag{4.16}$$

is minimum.

In case of a tie, the sub-tree, whose object type matches with the object to be inserted is chosen i.e., $O_{candidate} = O_r$ for which $O_r \rightarrow object\_type = O_n \rightarrow object\_type$. Essentially, a new object is inserted at the leaf node, and if it is full, a split is required followed by a rearrangement of the tree. Thus, it can be seen that GeM-Tree grows in a bottom-up manner and hence maintains the balanced structure.

4.3 Similarity Search

GeM-Tree uses a metric distance function, Euclidean Distance (L_2) as the ground distance of the EMD to determine the (dis)similarity between multimedia data objects. The number of features used to represent each multimedia data object in GeM-Tree is the same, only that some features may be absent in case of some particular type of data object and have zero values. In this case, the images do not have the features relevant to videos only and have zero values for all the 19 values for F_B. The applied signature representation for multimedia data objects makes sure that the similarity between two multimedia data objects is correctly translated and projected into the metric space, thus creating an effective index structure where similar data objects can be retrieved with minimum computation overhead and false dismissals.

For example, the signature of an image and a video shot can be represented as F_{image} and F_{shot} as follows:

$$F_{image} = \{\underbrace{\{x_1, x_2, ..., x_i\}}_{F_A}, \underbrace{\{0, 0, ..., 0\}}_{F_B}, \underbrace{\{1, 0, 0\}}_{F_C}, \underbrace{1}_{weight}\} \tag{4.17}$$

$$F_{shot} = \{\underbrace{\{z_1, z_2, ..., z_i\}}_{F_A}, \underbrace{\{y_1, y_2, ..., y_j\}}_{F_B}, \underbrace{\{1, 1, 0\}}_{F_C}, \underbrace{1}_{weight}\} \tag{4.18}$$

The similarity between an image (image1) and a shot (shot1) can be related to the similarity between two shots (shot1 and shot2 respectively) as follows: if,

$$d(F_{image1}, F_{shot1}) \leq d(F_{shot1}, F_{shot2}) \tag{4.19}$$

We can conclude,

$$d(F_{A\_image1}, F_{A\_shot1}) \leq d(F_{A\_shot1}, F_{B\_shot2}) \tag{4.20}$$

and

$$d(F_{B\_image1}, F_{B\_shot1}) \geq d(F_{B\_shot1}, F_{B\_shot2}) \tag{4.21}$$

From Equations 4.19, 4.20, and 4.21, it is clear that in terms of similarity measure, if the Euclidean Distance measurement indicates that the similarity between an image and a video shot is more than that between two video shots, it is correct to organize the image and the video shot together rather than the two video shots (even if they belong to different categories of multimedia data). As can be seen from Equation 4.21, the similarity between the video parts of the signature (F_B) between image1 and shot1 is always less than that of shot1 and shot2. This is because the F_B part of an image has all zero values, thus the Euclidean Distance between it and the F_B of any video shot having non-zero values will be always more than the Euclidean Distance between that shot and any other shot (which always has a non-zero F_B values). Also, this dissimilarity could not override the similarity between the image part (F_A) of image1 with shot1 as depicted in Equation 4.20. Thus, it can be concluded that shot1 is more related to image1 in terms of F_A than it is with shot2 in terms of F_B. Also, in this case, a signature

consisting of only one distribution is used, the EMD will be directly proportional to the ground distance when the weights are equal since there will be only one possible flow between two signatures [185]. For scenarios when multimedia data objects might need to be represented with a variable length feature distribution, the weight assignment is a crucial step and should be carefully handled so that the ground distance translates to meaningful relationships when the entire EMD between two multimedia data objects is calculated.

4.3.1 High Level Semantic Relationship

The high level image relationship used in GeM-Tree is captured using a stochastic construct called the Markov Model Mediator (MMM), that maps the low level features and high level concepts in CBIR [191] by capturing the image relationships as perceived by the user. MMM is a probabilistic mechanism that adopts the Markov model framework and the mediator [191]. The MMM mechanism is represented as a 5-tuple $\lambda = (S, F, A, B, \pi)$, where S is the set of images, A is the state transition probability distribution, B is the feature vector and π is the initial state probability distribution. From this tuple, our point of interest is the state transition matrix denoted by A, where each entry *(i, j)* corresponds to the relationship between image i and j captured by a training processes. The MMM mechanism builds an index vector for each image in the database and considers the relationship between the query image and the target image. The main idea is *the more frequent two images are accessed together, the more related they are.* The relative affinity measurement ($aff_{m,n}$) between two images m and n is defined as follows:

$$aff_{m,n} = \sum_{k=1}^{q} use_{m,k} \times use_{n,k} \times access_k \tag{4.22}$$

Here, $use_{m,k}$ denotes the usage pattern of image m with respect to query q_k per time period, and $access_k$ denotes the access frequency of query q_k per time period. The state transition probability matrix is built by having $a_{m,n}$ as the element in the $(m, n)^{th}$ position

of A. The $a_{m,n}$ value is defined as

$$a_{m,n} = \frac{aff_{m,n}}{\sum_{n \epsilon d} aff_{m,n}} \tag{4.23}$$

In a similar fashion, the semantic relationship among videos considering the different units is captured using an extension of the above explained MMM model called HMMM (Hierarchical Markov Model Mediator [53]). It is represented by an 8-tuple, where each element of the tuple is discussed in details in [53]. The element n represents the number of levels in an HMMM and the purpose and representation of the other elements varies with the level under consideration. In this approach, n is set as 2 and we are mainly concerned with the following three elements of the tuple viz. B, A and F during the similarity searches. B represents the set of distinct features in level 0 and semantic concepts in level 1, A represents affinity matrix which denotes the similarity measurement between video units as perceived by the users and collected over time and F represents the low-level feature information for each frame at level 0 and concept matrices at level 1. The matrices are constantly updated for each iteration through a learning process by utilizing users' feedback.

It is worth mentioning here that HMMM is an extension of MMM into different levels where the tuples carry different information depending upon the level. Thus, the semantic relationships of images can be easily integrated into HMMM at the level that stores information about the video frames (as video frames and images are essentially similar units with video frames having additional video specific features such as audio and temporal information). Any high level image/video relationship capturing mechanism can be used in the proposed index structure without loss of generality.

4.3.2 Incorporation of Affinity Values

GeM-Tree embeds the high level image relationship in the metric space of the index structure. A metric tree indexes a metric space formed by points which are related to

each other by a metric distance function. A distance function is called metric if it obeys the laws of *symmetry*, *positivity* and *triangular inequality* as discussed below:

Let O_x, O_y and O_z be three indexed objects. They are considered to belong to a metric space if the following conditions are satisfied:

$$d(O_x, O_y) = d(O_y, O_x) \tag{4.24}$$

$$0 < d(O_x, O_y) < \infty, O_x \neq O_y, d(O_x, O_x) = 0 \tag{4.25}$$

$$d(O_x, O_y) \leq d(O_x, O_z) + d(O_z, O_y) \tag{4.26}$$

In the proposed GeM-Tree, the *affinity relationship* (the high-level multimedia data relationships) could not be incorporated within the distance function without violating its metric characteristics. If the *affinity relationships* between the data points are used as a factor to scale the distance between them (higher the affinity value, lower is the computed distance), the *triangular inequality* of the distance function demands the factor to be the same. The affinity value is a high level concept depicting the similarity between each pair of images as perceived by the user and hence cannot be equal to each other. If the affinity, or in other words the user concept of similarity, could have been projected in the feature space, the distance function could have been scaled using them by attaching different weights to the feature values as discussed in [36]. However, it will then make the distance function arbitrary. Moreover, our goal is to embed the semantic relationship as it is in the index structure. Hence, the affinity relationship is incorporated after the index structure is formed during each query for pruning the tree by a method called affinity promotion. The detailed derivation of the above claim is described in Lemma 4.3.1.

Lemma 4.3.1 *The affinity relationship cannot be involved while constructing the GeM-tree as it no longer keeps the search space metric.*

Proof. Let O_1, O_2 and O_3 be three objects in a metric space. Let D_{13}, D_{12}, and D_{23} be the original distances computed before the introduction of the affinity values. Let K_{13}, K_{12} and K_{23} be the affinity factors used to scale down the distance and β_{13}, β_{12} and β_{23} be the newly computed distance functions. Therefore,

$$\beta_{13} = \frac{D_{13}}{K_{13}}, \beta_{23} = \frac{D_{23}}{K_{23}}, \beta_{12} = \frac{D_{12}}{K_{12}} \tag{4.27}$$

Thus,

$$D_{13} = K_{13}\beta_{13}, D_{23} = K_{23}\beta_{23}, D_{12} = K_{12}\beta_{12} \tag{4.28}$$

According to *triangular inequality*,

$$D_{13} \leq D_{12} + D_{23}, D_{12} \leq D_{13} + D_{23}, D_{23} \leq D_{12} + D_{13} \tag{4.29}$$

Thus,

$$\beta_{13} \leq (\frac{K_{12}}{K_{13}})\beta_{12} + (\frac{K_{23}}{K_{13}})\beta_{23} \tag{4.30}$$

$$\beta_{12} \leq (\frac{K_{13}}{K_{12}})\beta_{13} + (\frac{K_{23}}{K_{12}})\beta_{23} \tag{4.31}$$

$$\beta_{23} \leq (\frac{K_{12}}{K_{23}})\beta_{12} + (\frac{K_{13}}{K_{23}})\beta_{13} \tag{4.32}$$

Hence, to maintain the triangular inequality of the weighted distance function, From equations (4.30, 4.31 and 4.32),

$$\frac{K_{12}}{K_{13}} = 1, \frac{K_{23}}{K_{13}} = 1 \tag{4.33}$$

The above proves that the affinity factor should be the same to maintain the triangular inequality. □

4.3.3 Affinity Promotion in GeM-Tree

As discussed in the above section, the affinity relationships cannot be introduced into any distance based index structure during building it in order to satisfy the metric condition of triangular inequality of the metric search space.

Table 4.1: Affinity promotion for GeM-Tree

```
Promote_Affinity (Q) {
   object_type = Find_Object_Type(Q);
   //Determine object type of the query.
   Traverse_to_leaf();
   maxAffinity = 0;
   A = Load_Affinity_Matrix(object_type); //Load affinity matrix.
   ∀ N do: {
    ∀ O_r in N do: {
     if (object_type ≠ N → object_type) {
      Traverse the HMMM Model to get the affinity value
      between cross-multimedia data types;
     }
     temp_affinity = A(O_r, Q);
     if (temp_affinity ≥ maxAffinity) {
      maxAffinity = temp_affinity;
    } }
   N → parent → affinity = maxAffinity;
   Goto_Level(N → parent → level);
  } }
```

They need to be promoted from the leaves to the intermediate nodes before each query. The main idea behind the affinity promotion is to ascertain that there is no false dismissal and no unnecessary sub-tree traversal as discussed in Definition 4.3.2.

Definition 4.3.2 *Let N_a and N_b be the leaf nodes of GeM-Tree containing the indexed objects O_a and O_b respectively represented by their keys or features, and N_r be the parent of N_a and N_b. Let $aff_{a,q}$ and $aff_{b,q}$ be the precomputed affinity values between the query object and the objects at the leaf level. Hence, the affinity value of the parent of N_a and N_b (i.e., N_r) with respect to the query O_q is equal to max($aff_{a,q}$, $aff_{b,q}$).*

When a query is submitted, at first the object type of the query is determined to find out if the submitted query is an image, a shot or a video. Then the appropriate affinity matrix is loaded and the leaf nodes are examined to obtain the affinity relationship between the each one of them and the query object. If the examined leaf node holds an object that is of the same type as the type of the query object, the corresponding affinity value from the affinity matrix is stored. If the object type of the examined leaf is different from the object type of the query, the corresponding affinity is set to zero. Once all the leaf nodes have a particular affinity value with respect to the query object, the maximum of the affinity values among the sibling leaf nodes is determined. This becomes the affinity of the parent node to which the set of leaves belong. The affinity is stored in the appropriate place holder according to the object type of the submitted query. This process continues till the root of the GeM-Tree is reached and it ensures that if there is at least one object belonging to the object type of the submitted query and possessing an affinity value greater than or equal to the required affinity, which is the child of a particular node, this fact gets reflected in the affinity value stored at the parent. Thus, false dismissals will be avoided during the similarity search. Additionally, it also ensures that if there is no object satisfying both the object type and the affinity matching, the parent node can be confidently pruned without any further consideration, thus saving huge computation overhead during the similarity searches. If cross multimedia data type need to be extracted, more than one affinity matrices are considered simultaneously and multiple affinity place holders need to be filled before the similarity search.

4.3.4 k-NN Search

The k-NN algorithm for GeM-Tree supports both content-based image and video retrieval considering the high-level semantic relationships between the multimedia data objects. In addition, GeM-tree is capable of answering queries that involve both images and videos

together. The pseudo-code for the k-NN search of GeM-Tree is presented in Table 4.2. In this routine, first the affinity promotion is done as discussed in Section 4.3.3.

Table 4.2: k-NN search algorithm for GeM-Tree

```
k-NN_GeneralSearch (Q, N, k), {
  Promote_Affinity(Q); //affinity promotion for
    // image_affinity, shot_affinity and video_affinity
   if (N ≠ leaf) {
    ∀ O_r in N do: {
     if (| d(O_r, Q)  r(O_r) | ≤ d_k) {
      if (O_r → object_type = Q → object_type) {
       if (aff(O_r, Q) ≥ aff_k) {
        Update(d_k);
        Update(aff_k);
        k-NN_GeneralSearch (Q, T(O_r), k);
        //T(O_r) points the root of the subtree of O_r.
       } }
      elseif (Q is a video) {
       if (O_r is a shot) {
        if ((aff(O_r → v_id, Q → object_id) ≥ aff_k)) {
         Update(d_k);
         Update(aff_k);
         k-NN_GeneralSearch (Q, T(O_r), k);
        } } }
      else if (Q is a shot) {
       if (O_r is a video) {
        if ((aff(O_r → object_id, Q → v_id) ≥ aff_k)) {
         Update(d_k);
         Update(aff_k);
         k-NN_GeneralSearch (Q, T(O_r), k);
        } } }
      Update(d_k);
      k-NN_GeneralSearch (Q, T(O_r), k);
     } } }
   // For the leaf node, perform all the checks as the intermediate
   // nodes and if it qualifies but instead of recursion,
   // add the node pointer to the result set and update d_k.
  }
```

For each intermediate node in the GeM-Tree, the feature similarity and the affinity value (if similar object types) with respect to the query object is checked. If both

the similarities of the candidate node is greater than the nodes examined so far, it is stored in a priority queue of possible nodes for future recursion. The priority queue is updated and so are the dynamic threshold distance and the threshold affinity value. The process continues in an recursive manner. Now, if the object type of the candidate node does not match with the query object type, the hierarchy relationship is traversed upwards/downwards to find an affinity value between the two data objects. For example, if the query is a video object type and the node examined is a shot, the indirect high-level similarity between them (if any) is determined by checking the affinity between the video to which this shot belongs to and the video object of the submitted query. Similarly, other hierarchical relationships are utilized to gather the indirect affinity relationships. If there is no available hierarchical relationship between the query object type and the object type of candidate intermediate node and thus no available affinity relationship, the search procedure is continued depending on the feature-level similarity only. For the leaf nodes, the same steps are undertaken with the difference of adding the candidate objects to the result set without further recursion if they satisfy both the low-level and high level similarity requirements. Thus the k-NN algorithm of GeM-Tree is flexible and can accommodate different kinds of video unit classifications. For this application, a shot is used as the lowest unit of a video and the algorithm is adjusted to reflect it. The next two chapters discusses the content-based image retrieval and different types of content-based video retrieval algorithms respectively.

Another flexibility of the GeM-Tree is that it allows for only video or only image searches as well. For example, if one wishes to search only videos, the distance function can be modified to compare the feature similarity of only the video part, i.e., F_B of the multimedia object signature, and the k-NN search algorithm will automatically pick up the k nearest videos or shots to a submitted query. For dedicated content based image retrieval, the above technique is slightly modified and though F_A of the signature is used in the distance function, the result might contain both shots as well as images. In this

case, a second stage of refinement is performed, where for each shot in the result set, the frames, of which the shot is made up, are checked for similarity with the query image, and the k nearest images/frames are picked up from the multimedia database as the result set.

4.4 Empirical Study

In this section, the detailed of the implementation of the GeM-Tree and analysis of the experimental results is provided. The H-Tree and M-Tree packages [41][171] were used as a framework upon which the GeM-Tree application was built using C++ in an Linux environment. A node size of 4 Kbytes was used. The image database used has 10,000 color images from the Corel dataset of 72 semantic categories. The feature matrix is developed by obtaining the color information for each image from its HSV color space. Twelve color features viz. 'black', 'white', 'red', 'red-yellow', 'yellow', 'yellow-green', 'green', 'green-blue', 'blue', 'blue-purple', 'purple' and 'purple-red' are considered which makes the feature matrix 12-dimensional. If the number of pixels of a particular color is less than 5% of the total number of pixels, the corresponding color has a value 0 in the feature vector. An affinity relationship matrix of dimension $10,000$ X $10,000$ is used which is precomputed from a training set capturing the user perception.

An extensive study of the computation cost in terms of distance calculations and the relevance of query results in terms of accuracy is performed for Gem-Tree for different multimedia data types, viz. images and videos. Three different types of queries are executed, namely queries involving only images, queries involving only videos, and queries involving both images and videos. The results obtained from GeM-Tree are compared with a distance-based index structure for *only* images [45] (labeled as I in Tables 4.3 to 4.5), a distance-based index structure for *only* videos [48] (labeled as II in Tables 4.3 to 4.5), and with a sequential search approach (labeled as III in Tables 4.3 to 4.5).

Essentially, I and II have the same framework as GeM-Tree. The only difference is that in this experimental setup, while GeM-Tree indexes a data corpus having both images and videos, I and II indexes a data corpus having only images and videos respectively. Since the sequential search method essentially computes the distances between every pair of multimedia data objects present in the system, it has the highest accuracy and is used as a bench mark to determine the relevance of the query results obtained from the frameworks with index structures. Its distance computation is also presented to show that the high accuracy comes at the cost of a high computation overhead. Also, no matter what object is being searched, the sequential search goes through the entire dataset to generate the results. Thus, the number of distance computations is always the same for method III. The performance of the proposed index structure is compared with other distance-based multidimensional index structures (here, M-Tree) in Chapter 5 for images and with other video retrieval systems in Chapter 6.

We performed the experiments on two different Data Signature types. Table 4.3 and Table 4.4 use a fixed-length feature distribution where each image/video frame is considered as a single class/region. Nineteen color and texture features are extracted from them and the F_A part of the Signature is formed. F_B is formed from the nineteen video related features, and F_C captures the hierarchical relationships among the video shots. The results for 10 queries of each category are averaged. The first query type consists of querying the database, consisting of mixed type multimedia objects, for only images. The second type consists of querying the same database for only videos, and the third type comprises of cross-queries, where any multimedia object (images and videos) similar to a submitted query needs to be retrieved. To indicate that a particular index structure is incapable of handling a particular query type, the corresponding location in the table is marked with an 'X'. It can be seen that the computation cost for GeM-Tree in all the three types of queries is slightly higher than those of index structures for only images (I) and only videos (II). This is because GeM-Tree indexes more types

of multimedia data objects as the underlying database consists of both images as well as videos. The accuracy of GeM-Tree was slightly lower than those of method I and method II because of the same reason. Since the underlying database has both media types, while retrieving only one type of media, the search parameters are made stringent and only nodes satisfying the object type of the query are considered. These nodes might have object types, matching the query object, as their children. This is possible as it has been pointed out before that similarity of the low-level feature content is given a higher priority over object types while organizing the multimedia data objects in GeM-Tree. Thus, there can be some false dismissal which might affect the overall accuracy to a little extent. Such a problem can be easily overcome by making the query parameters more loose whereby a node having a different object type than the query object should be considered if it has at least one child with the same object type as the query. However, this might result in a slight increase of the computation cost.

However, it should be noted that though GeM-Tree has slightly lower performance in comparison to dedicated image-only and video-only index structures, it has the added capability to answer concept-based queries involving both images and videos.

Table 4.3: Distance computations during querying the index trees for fixed-length feature distributions

Query	Distance Computations			
	GeM	I	II	III
Only Image	98	80	X	1000
Only Video	63	X	50	1000
Cross Query	80	X	X	1000

Table 4.4: Accuracy for fixed-length feature distribution

Query	Accuracy			
	GeM	I	II	III
Only Image	90%	93%	X	98%
Only Video	90%	X	91%	95%
Cross Query	80%	X	X	90%

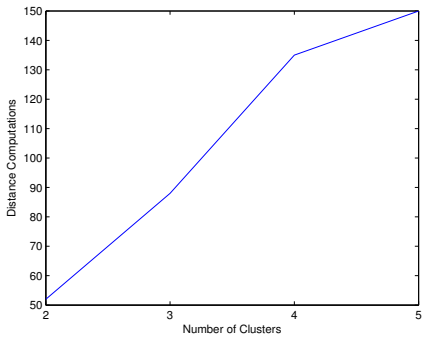


Figure 4.2: Distances vs. numbers of clusters for variable-length feature distribution.

Table 4.5: Distance computations during index tree formations for variable-length feature distributions

Data	Distance Computations		
	GeM	I	II
Images	145	X	X
Only Video	240	X	X
Both	960	X	X

The second set of experiments were performed on a variable-length feature distribution. As discussed in Section 4.2.2, each image/video frame is represented as clustered regions using Gaussian Mixture Models and Expectation Maximization techniques. We conducted a preliminary test on the HSI and the RGB color spaces and used various cluster sizes ranging from 2 to 5. Using the Akaike Information Criterion to determine the goodness of fit, we found that the optimum cluster size for most of the image/video frames was 4. Figure 4.2 presents the relationship between the distance computations and the number of clusters during constructing the GeM-Tree. It can be seen that the computation overhead increases with the increase of the number of clusters, which is obvious as with the increase of the clusters, the number of instances of each feature corresponding to each multimedia data object also increases. Thus, the total number of feature distributions representing the entire data object increases and the number of distance computations, necessary during the tree formation, increases as well. Hence, a

careful choice of the number of clusters should be made. If the number of clusters is large, though the subsequent similarity measurements will be more precise, it is at the cost of an increased computational overhead. Similarly, if in order to reduce the computation overhead, too few clusters are chosen, the data object will not be represented properly. This will further lead to poor relevance of query results.

Table 4.5 presents the comparison of the computation cost for variable-length data signatures between the different index structures. It uses about 500 multimedia objects consisting of images and videos. Among them, about 200 are images and 300 are video frames. It can be observed that both index types I and II are incapable of handling variable length feature distributions (indicated by 'X') as they do not use EMD as the distance function. Thus, compiling observations from Table 4.3, 4.4 and 4.5, it can be concluded that GeM-Tree has added capabilities over dedicated multimedia index structures with a comparable computation cost. It should be also pointed out that Table 4.5 represents the performance of GeM-Tree during the tree formation stage and thus the results presented should not be compared with Table 4.3 and 4.4, which represent the performance during the query stage and uses different datasets and representations.

4.5 Conclusion and Future Work

In this chapter, a common platform for indexing multimedia data objects with the help of a distance based multidimensional index structure called the GeM-Tree is discussed. GeM-Tree is a flexible structure and can accommodate different techniques of content-based retrievals by utilizing a variable length multimedia feature distribution and using EMD as the underlying distance function. To the best of our knowledge, GeM-Tree is the first attempt to organize two different types of multimedia objects with a single index structure and support queries that involve both. It is a very promising framework in the multimedia index genre and has ample potential to be improved and utilized for different

applications. As a future work, it is planned to use Gem-Tree to index documents as well. Documents can be considered as the third genre of multimedia data apart from images and videos. Thus, to be able to develop one seamless framework for indexing and retrieving all the different multimedia data, as per the main motivation of this research, supporting document indexing should be the next step.

CHAPTER 5

CONTENT-BASED IMAGE RETRIEVAL UTILIZING A MULTIDIMENSIONAL INDEX STRUCTURE

5.1 Content-Based Image Retrieval in GeM-Tree

In this chapter, the way by which GeM-Tree handles images and their retrieval strategies are discussed in details. Content-Based Image Retrieval (CBIR) is by far the most popular retrieval and search technique for images. Index structures have two major query types: range query and k-NN query. The main criteria for an index structure to be considered fit to organize images is its ability to support content-based image retrieval via the range and the k-NN query, both in terms of low-level contents and high-level contents. This chapter is dedicated to investigate the detailed procedure by which GeM-Tree is able to answer queries for content-based image retrieval. It should be pointed out here that though while discussing GeM-Tree, in general it has been mainly presented as a metric tree, but there is a space-based filtering technique that can be utilized before the metric-space is indexed. Thus all the discussions in the previous chapter is made assuming that the metric space is already developed after filtering via the space-based indexing. However, in this chapter, the space-based indexing is considered during the query phases as well.

5.2 Similarity Queries

The proposed GeM-Tree supports both the popular queries supported by index structures viz. the range queries and the k-NN queries. Before going into the detailed algorithms, the tree traversal and the node information processing are discussed in each case. A query is represented as a collection of features Q(F), where F is the same set of feature vectors as that of the stored images and are extracted in the same manner. Once the

feature vector of the query image is obtained, the GeM-Tree is traversed from its root to the subspaces of the feature space containing data points related to the query object. The metric trees corresponding to subspaces having the maximum number of data points are merged and the affinity relationships of all the nodes in the metric tree are computed by affinity promotion technique. By computing the distance and the affinity between the query object and the tree objects of the metric tree, the query result is obtained. A detailed pseudo-code of the range and k-NN queries for the GeM-Tree-Tree is presented in the following subsections.

Table 5.1: Implementation of range query in GeM-Tree

```
GeM_Range_Query(R(Q):query_region, r(Q):search_radius, Q:query_object,
                aff:affinity_value, N:node) {
  if (N is Null) {terminate;}
  else
  {
     Let page=root page;
     R_NF=BR corresponding to N;
     Space_Search_Images(R(Q), N, R_NF); //space search sub-routine.
  } //end of space search.
  Set N_child=Root_Metric;
  Metric_Search_Images(Q, N_child, r(Q), aff); //metric search.
}//end of GeM_Range Search.
```

5.2.1 Range Queries

A range query (Q, $r(Q)$) traverses through the GeM-Tree and selects all the appropriate database objects (O_i) which satisfy the following condition:

$$\forall\ O_i,\ d(O_i,\ Q) < r(Q). \tag{5.1}$$

The GeM_Range_Query as discussed in Table 5.1 for the GeM-Tree is developed to implement the range query in the feature space as well as in the metric space. Since the space-based indexing technique requires a search range and a metric-based indexing technique requires a search radius to implement the range search, both the values are provided

while initializing the range search algorithm for the GeM-Tree. The algorithm for the range query is described in details in Table 5.1, 5.2 and 5.3. The GeM_Range_Query first performs range search on the feature space to get the feature sub-spaces within the supplied range of the query using the function Space_Search_Images(R(Q), N, R_{NF}) as discussed in Table 5.2. Once the feature subspaces are obtained, in order to increase the metric search space, neighboring feature subspaces are combined in a step-wise manner, depending upon the users' feedback, starting with just the original result obtained from the space search. The metric search method includes the introduction of the affinity concept. For the router objects i.e. the intermediate objects, the similarity distance is first evaluated against the search radius. If satisfied, the affinity of the routing object with the query object is checked against the required supplied affinity value. Upon satisfying both the conditions, the metric search is iterated for the subtree of the routing object. The metric search is implemented in the function Metric_Search_Images(Q, N_{child}, r(Q), aff) discussed in Table 5.3. For data objects residing at the leaf nodes, similar evaluation is performed except that the image objects are added to the result set (also a priority queue) directly when evaluation is successful instead of initiating an iterative metric search. For image objects without affinity values (possible if a new image object is introduced whose affinity value is not yet available), simple metric search is performed depending upon the classical similarity evaluation.

In many cases, providing an appropriate search radius with the query is rather difficult for the user and might not result in satisfactory query output. Moreover, at times, the similarity distance computation techniques fail to capture the users' similarity perception when it do not follow any feature-level similarity pattern or the query image is of a *'hard-to-interpret'* kind. Such a scenario is taken care by maintaining a second parallel result set, which is populated depending upon only the affinity requirement even when the similarity measurement criteria fails. This result set is used if the user is not satisfied with the earlier result set and it gives the high level image relationship a greater importance

Table 5.2: Implementation of space search for images in GeM-Tree

```
Space_Search_Images(R(Q), N, R_NF){ //Space Search.
  if (N ≠ Space_Data_Node)
  {
   I = R_NF ∩ R(Q);
   if (I ≠ ⊘)
    {
      ∀ child nodes in N {
         Compute R_child from R_NF;
         Set R_NF=R_child;
         Space_Search_Images(R(Q), N_child, R_NF);
       }
     }
   }
  } //end of Space Search Subroutine.
```

in determining the query result under such circumstances. Moreover, in this query result improvement technique, no additional computational overhead is incurred. An image object in the GeM-Tree qualifies for the original result set if it satisfies both the similarity and affinity criteria. If the similarity measurement fails, the object, instead of being discarded, is still checked against the affinity value. If it satisfies the affinity check, it is pushed into the parallel result set instead of the original one. Thus, it is clear that there is no need of additional computation to achieve the above described process except for operations on a priority queue.

5.2.2 k-NN Queries

The GeM_k-NN_Image_Search algorithm as discussed in Table 5.4 retrieves k nearest neighbors from the GeM-Tree for a query object Q. The GeM-Tree uses a branch-and-bound technique similar to the one designed for the R-Tree [98]. The algorithm proposed here to implement the k-NN query on the GeM-Tree first determines the k-nearest subspaces to a given query point. Then it merges the metric trees corresponding to each space, ultimately performing k-NN search on the combined metric tree

Table 5.3: Implementation of metric search for images in GeM-Tree

```
Metric_Search_Images(Q, N_child, r(Q), aff) { //Metric Search.
  Affinity_Promotion( ); //promotion of affinity value.
  if (aff(O_r, Q)≠ 0) { //affinity value available.
    if (O_r is a routing object){
     ∀ O_r in N_child do: {
      if (| d(O_M, Q) - d(O_r, O_M)| ≤ r(Q)+r(O_r)) {
        Compute d(O_r, Q) and aff(O_r, Q);
        if ((d(O_r, Q) ≤ r(Q)+r(O_r)) && (aff(O_r, Q) ≥ aff)) {
         Metric_Search_Images(ptr(T(O_r)), Q, aff);
            //T(O_r): pointer to the subtree.
        }
        elseif (aff(O_r, Q) ≥ aff){ //giving affinity relationship
                //greater priority over similarity measurement.
         Metric_Search_Images(ptr(T(O_r)), Q, aff);
            //T(O_r): pointer to the subtree.
        }
      }//end of search for O_r satisfying metric condition.
     }//end of search for all O_r in N_child.
    }//end of internal node search of the metric tree.
    elseif (O_r is a leaf object){
     If the object qualifies the distance function and the affinity,
     add to the result set;
    }
  } //end of search for query object with affinity.
  else {
   Metric_Search_Images with the absence of the affinity comparison;
  } //end of search for query object without affinity.
} //end of Metric Search Subroutine.
```

thus formed, to get the k nearest neighbor of the submitted query. The search algorithm implements an ordered depth-first-search on its feature space using the function Space_Nearest_Search_Images(N, nearest, Q) in Table 5.4. During traversal, at each non-leaf node, the metric bounds are calculated between the query point and all its Minimum Bounding Regions (MBRs) and stored in an ordered list. The list is pruned depending on the similarity measure and the search iterates upon this list until it is empty. In each iteration, the next sub-tree belonging to the particular MBR is selected. On reaching the data nodes of the feature based index structure, the value of the nearest distance is

updated and the iteration continues until k feature subspaces are obtained. The metric tree corresponding to each feature subspace thus obtained is then combined. The affinity values of the combined metric tree is promoted from the leaf levels. A priority queue is maintained which points to the active sub-trees of the metric tree. The function Metric_Nearest_Search_Images(N, k, Q) in Table 5.4 implements the metric search in the metric space. The search radius and the affinity value now become dynamic in nature and are defined in Definitions 5.2.1 and 5.2.2, respectively.

As in range search, k-NN query method of GeM-Tree attaches greater importance to high-level image relationship over the similarity distance computation when initial iterations of the similarity search fails to produce satisfactory query results. A parallel result set is maintained containing images which passes only the high-level image relationship condition even if the distance criteria is not met. As in range-search, such a technique is found to be particularly helpful when the user's concept of similarity does not depict enough feature level similarity among the images or the query image is of a *'hard-to-interpret'* kind.

Definition 5.2.1 *The search radius is defined as the distance between the query point and the current k-th nearest neighbor.*

Definition 5.2.2 *The affinity value is defined as the affinity between the query point and the current k-th nearest neighbor.*

The pseudo-code is described in Table 5.4.

5.3 Experimental Analysis

Extensive experiments are performed to evaluate the performance of the GeM-Tree during its construction and during queries involving content-based image retrieval. The implemented GeM-Tree is compared with the performance of M-Tree for both the query

Table 5.4: Implementation of k-NN search for image retrieval in GeM-Tree

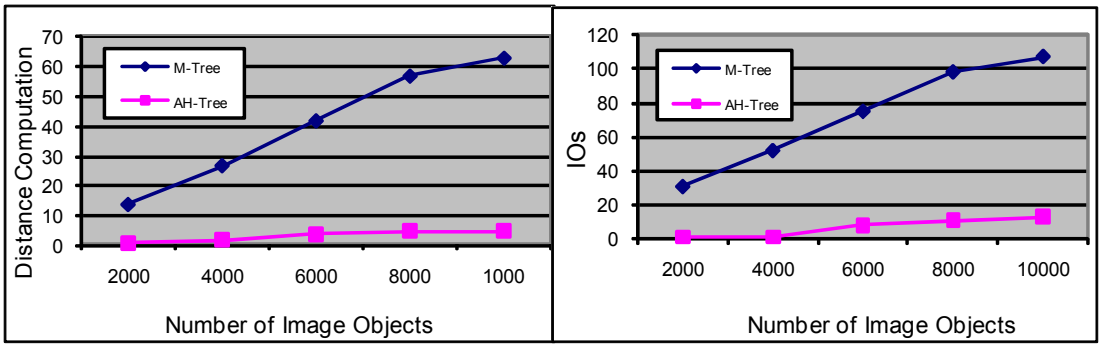
```
GeM_k-NN_Image_Search(N:node, nearest:distance, Q:query point,
        k:number of nearest neighbors){
  if (N is Null) {terminate;}
  else
  {
    Space_Nearest_Search_Images(N, nearest, Q);
    //Returns k nearest space, searching the Space Indexing Tree
    //by generating Available Node List and Sorting them
    //based on the similarity measure iteratively.

    //Combine the metric trees corresponding to k Spaces.

    //Search on the corresponding metric tree.
    Affinity_Promotion( );//promotion of the affinity value.
    //Perform the metric search as explained in range search with the
    //difference of making the search radius dynamic by making
    //it the distance between Q and the current kth nearest neighbor
    //and storing all the non leaf nodes with the required similarity
    //measurement in a priority queue.
    Metric_Nearest_Search_Images(N, k, Q);
  }
}//end of k-NN search.
```

types. The GeM-Tree structure is not compared with the Hybrid Tree or any other Spatial Access Mechanism (SAM) as it has already been discussed that the high-level image relationship introduced in the GeM-Tree cannot be utilized in any Space-based indexing technique without translating it to its low level equivalence. Also, the main purpose of introducing the feature-based index structure was for filtering the underlying metric space. The distance-based index structure performs the major search related work. Thus, introducing semantic relationships between the images in the metric space should be sufficient to provide satisfactory results. However, if desired, existing approaches like [37] can be used to introduce the user perception in the feature space during search methods without disturbing the proposed similarity search techniques. It should be noted that the choice of the number of feature dimensions and their types should not have any

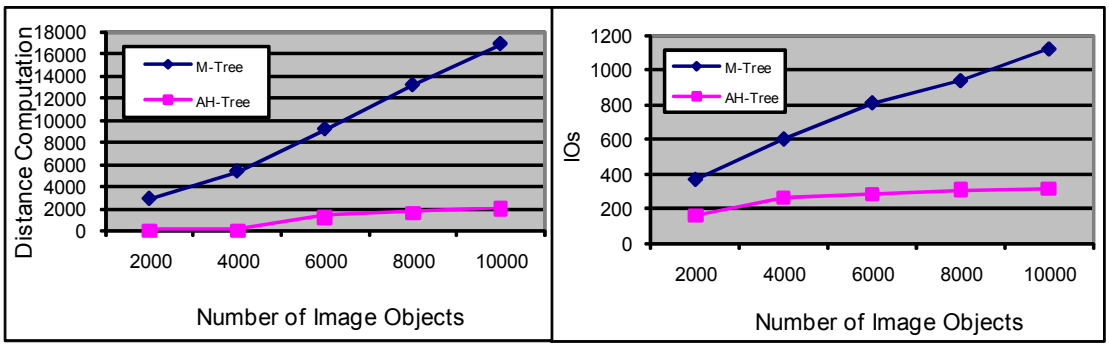
drastic effect on the observed results. This is because, with the increase of the feature space or by using more sophisticated methods of representation, though the quality and relevance of the result set might improve, it will have the same degree of improvement in both the proposed GeM-Tree and the compared index structure i.e. M-Tree. Thus, the relative performance of GeM-Tree with respect to other frameworks, with which it is being compared to, would remain the same.

The experimental results imply that GeM-Tree is capable of reducing the computation costs by combining the space-based and distance-based indexing structure. Figure 5.1(a) depicts the distance computation and number of I/O vs. number of objects for M-Tree and GeM-Tree. It clearly indicates that by using the space based indexing structure to filter the feature space prior to building the M-Tree for each subspace, there has been a noticeable reduction in computation overheads. Experiments are carried to implement both range as well as k-NN queries using 10 query images each for both GeM-Tree and M-Tree with $k = 10$. The distance computations and number of I/O averaged over 10 queries of the range and k-NN query is plotted in Figure 5.1(b) and Figure 5.1(c). The results also demonstrate that GeM-Tree performs far better as far as overhead is concerned. The labels of the graphs indicate AH-Tree to refer to the tree structure of GeM-Tree when only images are indexed.

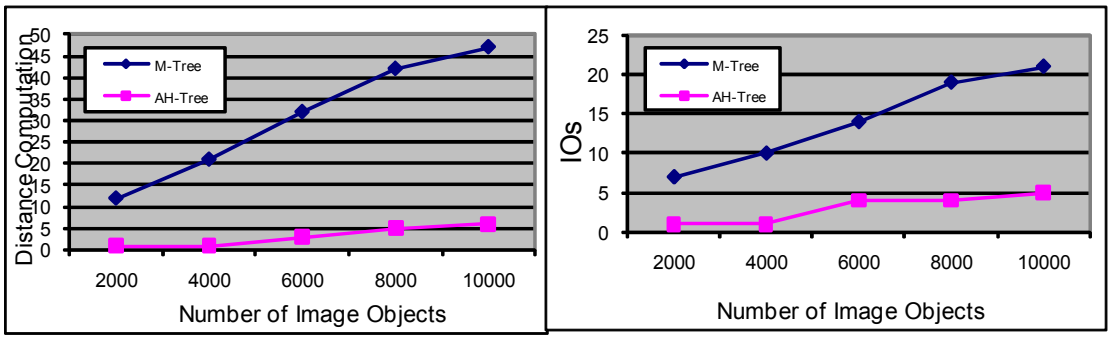
Since one of the major contributions of the proposed approach is the introduction of the high level image relationship in the index structure to facilitate getting semantically related query results without translating them into their low-level equivalence, hence the query results obtained from GeM-Tree as well as those obtained from M-Tree are checked for accuracy against images annotated manually. The *accuracy* is defined as *the percentage of the retrieved images that are semantically related to the query image as marked by the user*. It is noted that the results obtained from the M-Tree do not exhibit any regular pattern of semantic relationships and have accuracy as low as 10% on an average as depicted in Figure 5.2. GeM-Tree on the other hand has an average



(a)



(b)



(c)

Figure 5.1: Distance computation and number of I/O during (a) Building the index trees (b) Range queries (c) k-NN queries

accuracy over 80% which is depicted in Figure 5.3 for an example 10-NN query where the query image is at the top left-most corner highlighted in red. It can be seen that

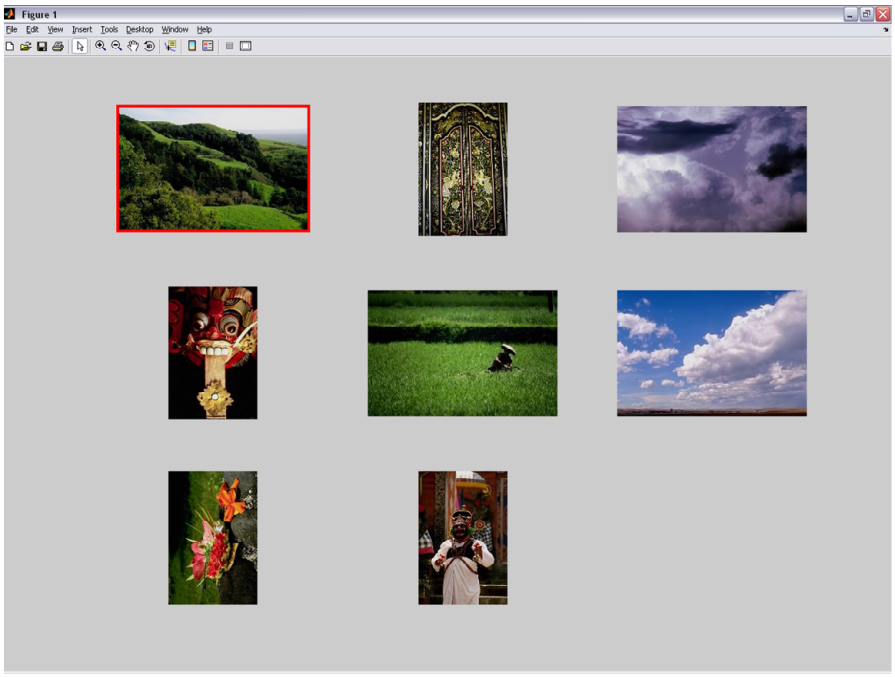


Figure 5.2: Query results without including the affinity value

about 8 among the 10 retrieved images have a close semantic relationship (animals in natural surroundings) and hence possess an accuracy of 80% for this example. The result is ranked in an order of decreasing similarity from left to right and top to bottom. Such stark improvement in the accuracy of obtained query results is clearly due to the introduction of high level image relationship.

In general, satisfactory result is obtained for range queries too as illustrated in Figure 5.4. Here, a query radius of 0.2 and an affinity of 0.23 is used. It is seen that the result relevance is in general better in case of nearest neighbor queries than in range queries since in the later the user need to supply the range radius, which at times is difficult to determine. Thus, if the range radius is small, it limits the search space in contrast to the nearest neighbor query where the entire search space is considered to get the top-k

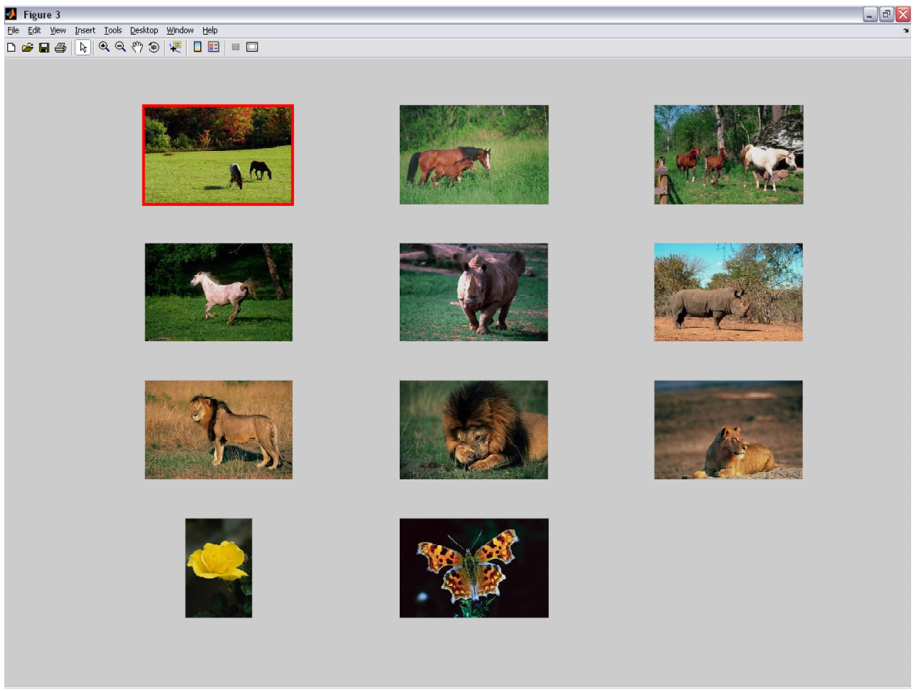


Figure 5.3: Query results for 10-NN query in GeM-Tree

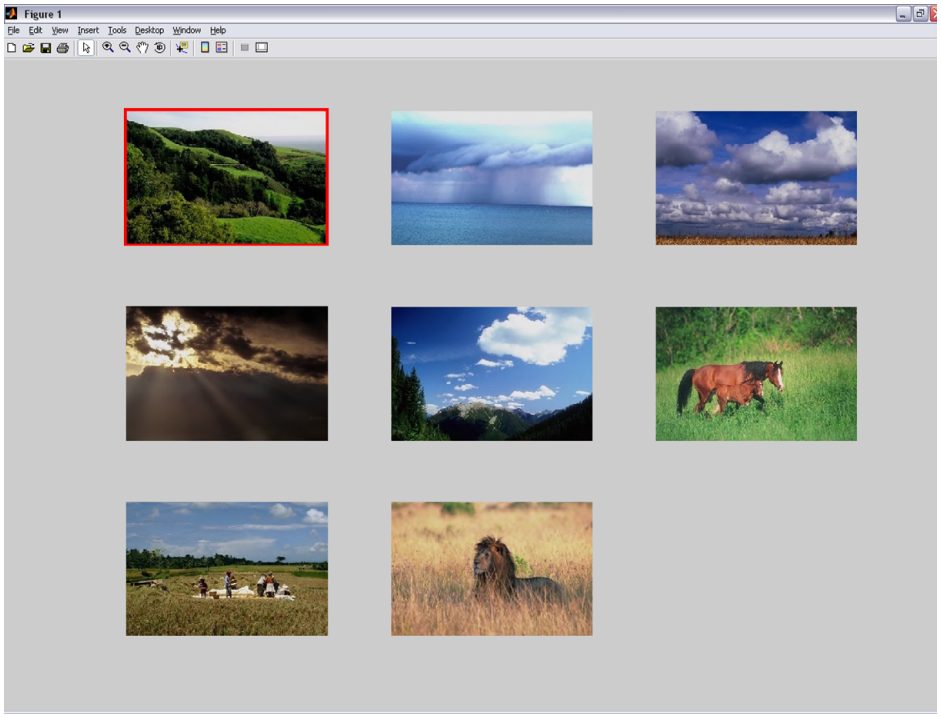


Figure 5.4: Query results for range search with radius and affinity relationship equal to 0.2 and 0.23, respectively

match. On the other hand, if the radius is too large, too many results are returned along with several false positives which reduce the result accuracy.

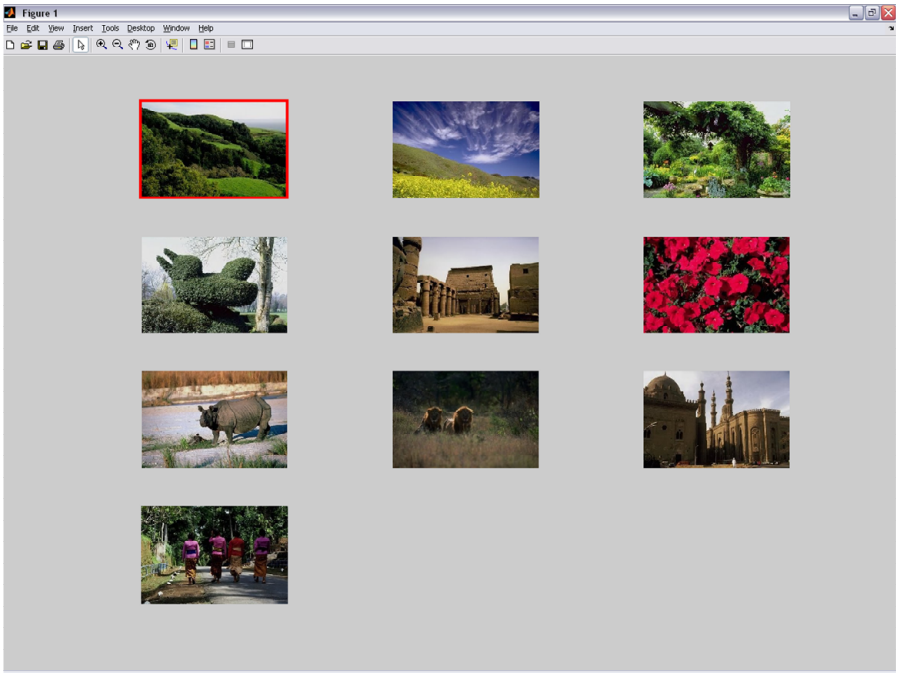


Figure 5.5: Query results obtained giving equal importance to similarity measurement and high-level image relationship

As pointed out in this chapter, both during range and k-NN queries of the GeM-Tree, a parallel result set is maintained giving the high-level image relationship greater importance over similarity criteria. Experiments are performed to corroborate the claim that indeed at times the users' perception of similarity is not represented by the feature-level closeness of the image objects. Better results are obtained while attaching greater importance to high-level image relationship. Figure 5.5 shows a nearest-neighbor query result while attaching equal importance to distance criteria and affinity relationship. It can be seen that for this particular query object (at the top leftmost corner highlighted in red), the results obtained has a low accuracy of 40% (3 out of 8 retrieved images are relevant which are the first three images of the result). But if the parallel result set is used instead (giving priority to affinity relationship), a query result is obtained with a

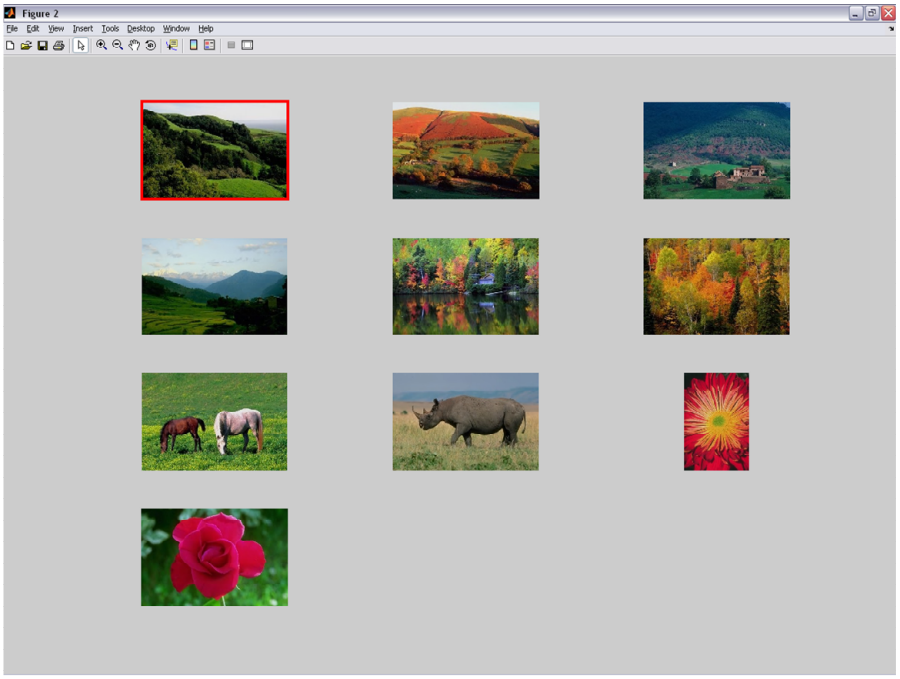


Figure 5.6: Query resultds obtained giving more importance to high-level image relationship

higher accuracy of 67% as depicted in Figure 5.6 (query image is at the top leftmost corner highlighted in red and the relevant images are the first six of the result). It should be noted that in this example, a particular *'hard-to-interpret'* image example is chosen to explain the typical scenario which is the cause of the low overall accuracy level. A *'hard-to-interpret'* image, as discussed in earlier sections, are images whose feature vector representation is incapable of distinguishing it properly and the semantic concept is not related closely with its low-level feature representation.

The above analysis of the experimental results help us to conclude that the proposed GeM-Tree indeed performs better both in terms of computation overhead and relevance of the query results. Moreover it has some additional features like maintenance of a parallel result set, which produces better query result with no major additional computational overhead. Thus, it achieves the two essential goals of any multimedia indexing structures.

First, it reduces the computation time in retrieving multimedia objects and second, it makes the retrieved result as close to human perception as possible.

5.4 Conclusion

In this chapter, the technique of managing images and accommodating CBIR into the GeM-Tree is discussed in details. Detailed search algorithms viz. range search and nearest neighbor query is presented which are capable of taking care of the complex *hybrid* nature of the framework. Moreover, a parallel result set maintenance mechanism is proposed to improve the query result without incurring additional computational overhead. The experimental results demonstrate that the proposed GeM-Tree is a promising indexing mechanism to bridge the gap between the low level features and the high level image relationship, and has potentials for future research and development. As a part of future work, it is planned to introduce data mining approaches to analyze the access patterns of users and use the analysis to modify the tree structure by utilizing the knowledge in determining the split policies. Thus, a factor of intelligence will be introduced to the tree structure and it will be able to self adjust its structure to improve the query results in future iterations.

CHAPTER 6

CONTENT-BASED VIDEO RETRIEVAL UTILIZING A MULTIDIMENSIONAL INDEX STRUCTURE

Videos are considered more complex than images as they carry more information. As a consequence, the retrieval strategies need to consider additional aspects to cover all these additional information stored in videos. Thus, content-based video retrievals supported by the range and k-NN queries of an index structure require a dedicated discussion. The generalized retrieval algorithm is presented in Chapter 4. The retrieval algorithms presented here can be achieved by extending those presented in Chapter 4, without any loss of generality. The main focus of this chapter is to discuss the different modeling techniques of videos and how they affect the retrieval strategies. It should be also pointed out here that all the algorithms presented here assumes that the index structure is handling only videos. Basically, algorithms presented in this chapter along with those presented in Chapter 5 should be considered together while developing the general query handling as discussed in Chapter 4, Section 4.3.4.

Traditionally, the term *video indexing* translates to the process of classifying the video contents and assigning content-based labels to them for the ease and precision of retrieval processes as depicted schematically in 6.1 [169]. Thus, it usually dealt with issues of inserting new videos into existing repositories, segmenting the video data into smaller pieces, extracting the features from the video units and analyzing the contents. So, indexing was synonymous with classifying video units and using them during the retrieval phase for relevant results.

As pointed out in [201], three main issues arise while classifying the video content viz. granularity, modality and type. There are different video indexing techniques like [5][57][97] etc. from the traditional video classification point of view. For example, [57] tends to index videos based on single modality whereas [5][10][69] uses a more advanced multi-modal approach to index the videos. [97] proposes another content-based video

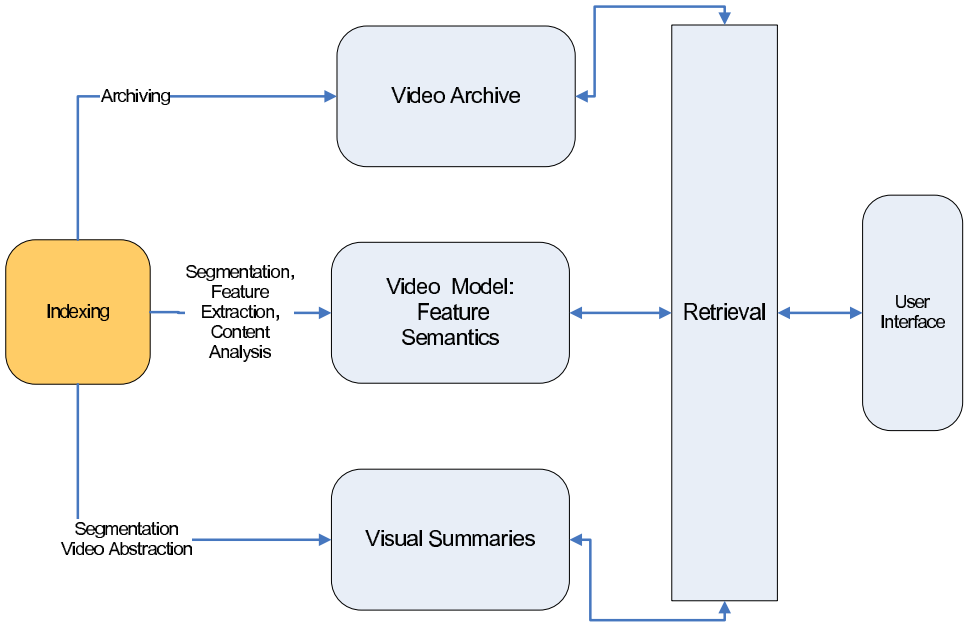


Figure 6.1: Traditional concept of indexing in video databases from video classification point of view

indexing/classification system which achieves the purpose of automatic management of video data by syntactic and semantic features. Other similar video techniques were proposed in [95][166] where concepts such as virtual image and Dublin core meta data [166] were used and statistical frameworks [95] were engaged for modeling and segmenting video content into coherent space-time segments. But none of the above techniques attempted to address the issue of indexing the video data from the true database or storage point of view. They classify the video data into units and design a way to identify useful information from them, but internally they need to perform exhaustive sequential search of the entire database to locate the video objects of interest. This increases the computation overhead and has increasing negative effects on the overall retrieval performance, especially for large video retrieval systems. Thus due to the absence of any storage-level index structure, the computations costs, number of I/Os and other database management related components' (for example query engine, retrieval engine, etc.) performances would be severely affected in those frameworks.

As will be discussed in details in Section 6.1, video data is modeled in three different ways viz. (i) Hierarchical Unit-Based Modeling, (ii) Feature-Based Modeling and (iii) Video Semantics Modeling. To develop a robust multimedia database management system, designing an index structure just as efficient and useful as index structures like kDB-Tree [179], R-Tree [98], is crucial which would accommodate all the above three categories of modeling to capture the different characteristics of video data. A set-based nearest neighbor approach applied on a multidimensional index structure, to index and retrieve videos based on their feature information, was proposed in [125]. Though [125] attempts to enable index structures like SR-Tree [126] to support video indexing, i.e. it indexes the video data from the database storage point of view, it has major drawbacks as it does not consider the first and third modeling approaches of video data. Thus it neither supports the different units of retrievals for video data nor does it consider the high-level semantic interpretation of multimedia objects in the similarity searches. The absence of both the characteristics limits the usability and performance of the index structure manifold. GeM-Tree supports the different aspects of video modeling along with the various levels of video-unit similarity searches (namely frame-level similarity search, shot-level similarity search and entire video-level similarity search) efficiently. and retrievals . To define and accommodate the high-level similarity among different units of video objects and bridge the gap between the feature and semantic information, a framework called HMMM [53] is embedded seamlessly within the k-NN similarity search.

6.1 Video Modeling

Understanding the different categories of video representation and video modeling is imperative for successfully designing a robust index structure and an efficient database management framework for video data. There are three major approaches of video modeling viz. Hierarchical Unit-Based Modeling, Fine-Grained Feature-Based Modeling and Video Semantics Modeling. The novelty of GeM-Tree is that it combines these three

approaches seamlessly in its index framework, and implements similarity searches that considers all the three approaches of video modeling in its k-NN methods.

6.1.1 Hierarchical Unit-Based Modeling

Early video modeling techniques [184] involved time-line that gave explicit, static, temporal relations between video elements on the time axis. They organized video units in chronological sequences and the inter- and intra-modality synchronizations are defined by the time-line binding. But the approach complicates and limits the editing process and does not support structures like high-level abstractions of basic video elements which groups videos into semantically related units. Thus, hierarchical approaches of video model representations were used [226], where complex video units were created by recursively combining smaller simpler units. A simple way to compose a hierarchy of video units is to utilize nesting [218] as the nested relationships between the nodes allow the user to explore the context in which it appears.

Temporal segmentation of a video sequence into meaningful units is called video unit classification. There are various levels of video units that have been proposed viz. shot level, frame level, scene level and clip level as presented in Figure 3.2. Among them shots are the most self-contained and well defined units. A shot-based approach categorizes a video sequence into a collection of frames where each collection represents a continuous camera action in time and space while sharing a close high-level semantic as well as low-level feature similarity. This dissertation uses video-shots as the lowest conceptual unit of videos. The video shot detection is mainly performed by adopting the three-level filtering architecture viz. pixel-histogram comparison, segmentation map comparison and object tracking as discussed in [52]. Each video shot consists of a number of temporally related video frames, one of which called the key frame, serves as a representative of the shot. For the purpose of ease, in this work, the first frame of each shot is identified as the key frame but other techniques can be used as well like selecting the frame which best

describes the overall concept of the shot. The combination of the low-level features of all the frames comprising a shot is used to represent each shot's feature vector.

A hierarchical structure is also utilized by video retrieval approaches that extract objects and events appearing in a video [142]. They segment and classify a video based on the common objects in it. They usually associate an object or group of objects with a set of frame sequences. Such a video modeling can be embedded into the video index structure without altering the basic framework.

6.1.2 Feature-Based Modeling

Early approaches of video retrieval borrowed ideas from image retrieval techniques and only added functionality for key-frame extractions. Then they applied similarity measurements on them based on the low-level features like color, texture, etc. But, such approaches were not satisfactory as video is temporal in nature and the sequential relationships among the frames comprising a logical unit, should be preserved. In addition, another feature mode, the audio features, are considered as the important source of information carried by videos which were absent for image data. So, static features comprising of single modality, as used in images, are insufficient for representing videos completely. Thus, this dissertation considers the multi-modal features on each frame and combine them to get the feature representation of the video shots. The multi-modal features (visual and audio) are extracted as proposed in [50] for each shot. Some important shot-level visual feature descriptors utilized in this work to represent the feature vector for each video shot are pixel change, histogram change, average volume, average energy, flux, etc.

6.1.3 Video Semantics Modeling

To model the semantic content of a video is far more difficult than the above two approaches of modeling. At the physical level, a video is a temporal sequence of pixel regions

without any obvious direct relation to its semantic content. In addition, if one considers multiple semantic meanings like metaphorical, associative, hidden etc, the problem becomes even more complex. The simplest way is to use text-based annotations. But such approaches have limitations like using only one annotation of the original video data, error in annotation techniques, not being able to handle the perception subjectivity efficiently, etc. Other methods like utilizing spatio-temporal approaches [120], using key words or key events [142], etc. limits the usability of the approaches as firstly they are complex and error prone and secondly using cues reduces the flexibility of the framework. Thus, in this research, in order to capture and utilize the high-level relationship among the different video units and bridge the gap between the low-level features and high-level semantic concepts attached to each video unit, a mathematical construct, called Hierarchical Markov Model Mediator [53] is used.

6.2 Similarity Search

Similarity searches for video datasets are mainly based on two different similarity criteria viz. low-level feature similarity and high-level semantic or conceptual similarity. When a query in the form of a video shot or a complete video is submitted, the k-NN search algorithm traverses the Gem-Tree and produces k most similar video objects to the user. During querying the GeM-Tree, the proposed k-NN search algorithm considers both the distance or (dis)similarity between the indexed nodes (video_node, key_shot_node) and the query object (also represented as feature vectors with the same data structure as the index tree nodes), as well as the high-level semantic relationships among them. A threshold value (affinity), specifying the minimum high-level similarity expected in the query result, is supplied with the query. This value is utilized to further prune the candidate nodes which have passed the distance criteria or the low-level similarity condition. It should be mentioned here that the k-NN search algorithm for GeM-Tree can

handle different video units as queries. For example, GeM-Tree can be queried frame-level, shot-level or entire video-level. It completely depends on the video units chosen by the users to classify the videos. Frame-level query may be issued to find frames similar to the submitted frame from within the same video or across multiple videos. Searching within the same video is useful when a video is large in size and users may be interested to find similar frames from within the video itself. Similarly, shot-level and video-level queries can be issued with the same efficiency.

The k-NN algorithm, supporting CBVR, starts with extracting features from the key shots representing the videos and the frames constituting a shot of the submitted query object, and represents them as multidimensional feature vectors. They may be of fixed length or variable length representation, depending upon the feature extraction technique. Then, depending upon the video unit of the submitted query, it proceeds to the corresponding portion of the retrieval algorithm as presented in Table 4.2. If an entire video is submitted and the routing object of the GeM-Tree being examined matches with the object type, the corresponding affinity value is checked from the appropriate matrix of the appropriate level of the HMMM (specifically obtained from the A matrix for level 1 of HMMM). The dynamic distance value is updated which stores the radius of the current k^{th} nearest neighbor. If the currently examined object do not match the object type of the query object, i.e. if it is a shot of a video, the affinity is computed in a roundabout manner where the affinity of the video to which this shot belongs is checked against the candidate affinity value (since the query is a video). The HMMM model enables to identify the indirect affinity relationships between cross-video-unit types. The algorithm have to just ensure that it points to the correct level of the HMMM framework and identifies the appropriate matrix relating the video unit of the routing object to the query object. Any video-unit can be handled seamlessly by this retrieval algorithm as long as it is included into the HMMM framework and represented explicitly in the feature signatures.

6.3 Experiments

In our experiments, 10 soccer videos were collected from different sources. They have total time duration of almost 2 hours. For each video, shot boundary detection was performed utilizing the concepts presented in [52]. For each shot, the key frame is set as the first frame and 20 multi-modal features (as discussed in Section 4.2.1) are extracted. Ten queries are executed comprising of video-level and shot-level queries. Since, to the

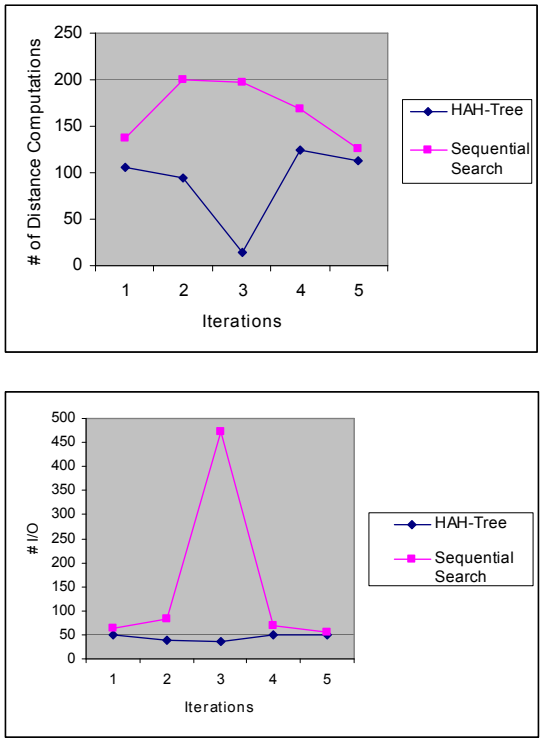


Figure 6.2: Distance computation and number of I/O of GeM-Tree compared with sequential search

best of our knowledge, there is no comparable video indexing framework like GeM-Tree that combines all the three characteristics of a video model into a multidimensional index structure, the performance could not be compared with any other tree-based video indexing strategy. However, this system is compared with the traditional exhaustive video retrieval strategy which does not have any underlying index structure from the storage point of view and depends only on the video classification techniques to provide

search result. Essentially, the exhaustive search traverses the entire video sequentially to provide the query results. The results presented in and Figure 6.2 demonstrates tremendous improvement in computation time and # of I/O for GeM-Tree over the traditional indexing concept method. The accuracy, a very subjective indicator for video retrievals, of GeM-Tree was satisfactory with an average value of $70\% - 80\%$. This value is lower than the exhaustive sequential search framework where it is achieved at the cost of very high computation overhead. The label of the graph in Figure 6.2 indicates HAH-Tree to refer to the tree structure of GeM-Tree when only videos are indexed.

6.4 Conclusion

In this chapter, the way GeM-Tree handles video data is discussed is details. The different video modeling techniques are presented with a detailed discussion of how the index structure seamlessly accommodates all the modeling requirements. The k-NN search algorithm for GeM-Tree is presented which supports CBVR by amalgamating low-level feature similarity and high-level semantic closeness among videos. The experimental results demonstrate encouraging outcome in terms of low computation overhead and a satisfactory accuracy of query results. As a part of future work, it is planned to include temporal relationships and event information in the index structure to improve its performance and broaden its domain.

CHAPTER 7

HYBRID QUERY REFINEMENT: A STRATEGY FOR A DISTANCE BASED INDEX STRUCTURE TO REFINE MULTIMEDIA QUERIES

An index structure is one of the major components of a database management system as it assists in efficiently organizing the data and enables quick and accurate retrieval. As discussed earlier, there are multidimensional index structures like [19][35][60][98][179][76][94] which can accommodate the atypical multidimensional representation of multimedia data but enabling them to efficiently support the popular retrieval strategies like content-based image and video retrievals is still a challenge due to the semantic information carried by such data types. The semantic interpretation of a multimedia data is very subjective and varies from user to user or even from iteration to iteration for an individual user. This makes the similarity queries issued to multimedia data *imprecise* in nature and a single iteration or a fixed query representation is not enough to capture the users' requirements during the retrieval process. Thus, attempts to capture the users' interest pattern are made with a strategy called *query refinement* which adjusts a query over multiple feedbacks from the user to better capture the users' information need. It has two major components viz. query modification and query re-weighting [170]. In query modification, the query representation is modified in each iteration to represent a region in the feature space which best describes the feature components of the users' query. In query re-weighting, the semantic interpretations of a query is modified, in each iteration, to better reflect the users' high-level perception. Thus, the index structures, in order to be able to handle the imprecise nature of the similarity queries of multimedia data, need to support query refinement with both its components efficiently.

As was discussed in details in the related work in Chapter 2, multidimensional index structures can be broadly divided into two categories viz. feature-based and distance-based. Both categories are useful depending on the dataset in hand and the applications that need to be supported. Hence, both categories of index structures need to have a

query refinement strategy to answer imprecise multimedia similarity queries more accurately. Though query refinement strategies has been designed for feature-based index structures like [36] and [170], to the best of our knowledge there are no query refinement strategies for distance-based index structures like [60]. Another major drawback of the existing approach is that if the semantic information of a multimedia object such as an image cannot be interpreted completely in terms of the inter and intra feature weights, refinement strategies like [170] fail to produce satisfactory results. This has been illustrated in Figure 1.3 for an image database where the feature-level similarity (calculated with a distance function) failed to capture users' high-level semantic perception.

In this chapter, a hybrid query refinement strategy is discussed for distance based index structures which organizes and manages mainly images [49]. It should be noted that the basic query refinement model used here can be utilized for indexing other multimedia objects like videos as well as long as the underlying distance-based index structures can accommodate the particular data type. The proposed query refinement strategy is called hybrid because it refines and adjusts both the low-level feature space as well as the high-level semantic interpretations individually and independent of each other during refining the queries in each iteration. It should be pointed out that in existing approaches like [170] basically only the feature space is refined, since the query is attempted to be refined by adjusting solely the feature attributes. But, in this proposed approach, specifically two separate refining methods, combined into one seamless approach is used. To refine the semantic interpretation of a query with each iteration, the proposed approach uses a dynamic parameter adjusting technique of a stochastic construct called Markov Model Mediator [191] while it adopts a query expansion approach to refine the feature space. This hybrid query refinement ensemble is introduced in a distance-based index structure and the similarity searches are designed to utilize it by adjusting the distance functions to be able to use the refined query structure. A new evaluation score determination technique is also proposed, called the Model Score, that can compare the overall performance

of the framework in terms of both computation time and F1 Score (relevance). Both the response time and the relevance of a query result is important in case of similarity queries based on contents for multimedia data. Thus, while evaluating and comparing the performance of an index structure for multimedia data supporting query refinement, one should be able to view the combined effect of both these criteria on the retrieval process and how each affect the performance of the other. The Affinity Hybrid Tree [45][46] (AH-Tree) was chosen as the distance-based index structure where the proposed hybrid query refinement strategy is introduced. The main cause for choosing Affinity Hybrid Tree is because it can handle the two different similarity concepts applicable to multimedia data, viz. low-level feature similarity and high-level semantic similarity, independently without trying to express one in terms of the other, as practiced by the existing approaches like [39][170]. Thus we found it as the right candidate to demonstrate the successful implementation of the proposed query refinement to a distance based index structure as both the approaches share the same basic philosophy of separating the low-level feature components of a query from the high-level semantic information that it carries.

7.1 Hybrid Query Refinement in a Distance-Based Index Structure

The hybrid query refinement technique is applied to the AH-Tree structure in two steps as there are two types of index structures embedded in it. Since, by intuition, it can be assumed that every user at every iteration will not change his/her information needs drastically, the proposed query refinement technique is literally applied for each iteration only to the metric space. This metric space is obtained for the starting query by filtering and combining feature spaces as explained in [46]. If it is seen that the metric space at hand does not contain the user's preferred points (deduced if the number of query results labeled relevant by the user falls below a certain threshold), the feature space is searched

with multiple query points and the feature spaces obtained for each query point are combined to build up the metric space. In this way, computation overhead in querying the entire database for each iteration is saved and a much smaller filtered metric space needs to be searched for subsequent iterations. Also, the purpose of demonstrating the performance of the proposed hybrid query refinement in a distance-based index structure is served.

7.1.1 The Refinement Model for Semantic Relationships

Experience shows that on several occasions, feature-wise similar objects do not share close semantic relationships. Thus the ideal query refinement techniques should explore methods to refine the high-level similarity concepts, independently from the feature-level similarity. This is implemented in the proposed hybrid query refinement technique where the high-level semantic relationship is dynamically refined and adjusted along with the feature space based on users' feedback in each iteration. The MMM framework [191] is utilized to introduce the high-level semantic relationships between the multimedia data objects in the index structure.Thus its constructs need to be manipulated to refine the users' information need and reflect the access pattern of the user with each completed iteration. In the next sub-section, the MMM framework is briefly introduced followed by a discussion on the dynamic manipulation of the semantic constructs.

Markov Model Mediator

Markov Model Mediator (MMM) [191] is a stochastic framework which provides an alternative retrieval mechanism for CBIR process. This method captures the high-level image relationship, called the affinity relationship, among image objects and use it during searching the image database and providing query results. It is an effort to bridge the gap between low-level features and high-level concepts. It is an alternative to Relevance Feedback method of refining queries to better capture users' perception during

CBIR. In [191], the method builds an index vector for each image within the database and based on the relationship between the query image and candidate images, produce results. But, unlike popular methods like RF, the users' perception is captured with the help of training data such as access patterns and access frequencies of the images in the database. The MMM mechanism is represented as a 5-tuple $\lambda = (S, F, A, B, \pi)$, where S is the set of images, A is the state transition probability distribution, B is the feature vector and π is the initial state probability distribution. From this tuple, the point of interest for the query refinement is the affinity matrix denoted by A, where each entry *(i, j)* corresponds to the relationship between image i and j captured in the training process. It is used during the similarity search routines of the AH-Tree along with the distance measurement to produce semantically close results.

The MMM mechanism can assist in retrieving very accurate results and is robust in the sense that it doesn't solely rely on feature-level similarity to provide query results. Thus, it can be used as a retrieval mechanism even when features cannot capture users' perception of semantic relation. Also, since it doesn't attempt to adjust feature weights, it do not make distance functions arbitrary during similarity computations and can be used in distance-based index structures easily to embed and refine the high-level semantic relationships.

Dynamic Refinement of Affinity Values

This is achieved by dynamically manipulating the $aff_{m,n}$ value (as explained in Equation 4.22) for each iteration. In each iteration, when a number of images from the result set for a query k are marked relevant by an user, the access frequency $access_k$ is increased by 1 while $use_{m,k}$ and $use_{n,k}$ are set to 1 for a pair of images m and n in the result set which are marked relevant. Thus, if the access frequency between two images at $(t-1)^{th}$ iteration be $access_{t-1}$, the affinity value at t^{th} iteration is refined as:

$$aff_{m,n_t} = 1 \times 1 \times (access_{t-1} + 1) \tag{7.1}$$

Also, if the query image itself belongs to the database, the row of the affinity matrix corresponding to it is selected and all the entries in that row are refined according to Equation 7.1, where m is set as k and $n \; \varepsilon \; p_1, p_2, ..., p_x$, where p_x is the image id marked relevant by the user and $x \; \varepsilon$ ids of relevant images in each iteration. A normalization of each row, affected in the feedback process, of the affinity matrix is performed after each iteration as Equation 4.23 and the modified affinity relationship score is utilized during the similarity search in the next iteration. The original affinity value before normalization is stored to get the actual access value for refinement in subsequent iterations. With Equation 7.1, the affinity relationship scores between the relevant images are refined (increased) dynamically whereas the scores between all other pairs remain the same. Hence, those relevant images will have a greater probability of retrieval than the rest of the images during subsequent retrieval iterations and also for similar query types issued by different users later. Thus, the query is refined independently in terms of the high-level semantic relationship by manipulating the probabilistic measure of access frequencies without relying on the relationship between the semantics and the feature space.

The refinements or modifications of the affinity values that occur in a particular row of the affinity matrix for a particular feedback from the user, is distributed through out the affinity matrix. The distribution takes place to pairs of affinity relationships that are directly or indirectly related to the modified/refined pairs of the affected row. For example, let, the $(i, j)^{th}$ pair of the affinity matrix be refined in iteration n. To distribute and reflect the update in the similarity perception throughout the affinity matrix, the affinity relationships for those elements in row i and j are refined with Equation 7.1, which have been marked relevant in the feedback in iteration n. The process is repeated for each pair marked relevant in a particular iteration. Thus with a single user feedback, quite a few number of semantic relationships are refined in a single go.

In addition, a separate data structure known as *profile_affinity_relationship*$_x$ is maintained, which copies the original Affinity Relationship matrix into a local profile specific to the x^{th} user and updates it according to the users' perception and preference. Normalization is done for each updated row after each iteration. Such *profile_affinity_relationship* structures for different users are evaluated after a certain interval and the universal affinity relationship matrix is updated accordingly. In this way, high-level similarity is not biased by any one user's preference and each user need not take the heavy burden of correctness of his/her response. Since the universal affinity relationship matrix update takes place off-line, after a considerable amount of data is accumulated in the *profile_affinity_relationship* all the rows and every image pair that are affected by the feedback can be updated without causing considerable delays in query processing.

7.1.2 Refinement Model for the Feature Space

The hybrid query refinement model, for refining the feature space, uses a multiple point query representation as it has been pointed out in the literature [170] that multiple point query representations (query expansions) better capture users' perception than aggregated single point query representation (query point movement). Such a query is represented by the following tuple: $Q = (n, P, W)$, where n represents the number of image points present in the refined query, P stands for set of the feature vectors for each points and W the weights attached by the user with each refined query point to rank them in order of their relevance with the original submitted query. Thus, with each submitted feedback, the query representation is modified. The above method of representing the refined query modifies the requirement of the user in terms of the target feature space. For example, for traditional queries when a k-NN query is submitted with a single image object, it meant "*search the database and give me k images which are nearest to the query in terms of features*". With the refined query, the requirement changes to "*search the database and give me k images which are nearest to* ***all*** *the query images in terms*

of features". Thus, the above query representation expands the feature search space and adjusts it with each iteration.

To utilize the the above representation of a refined query in a distance-based index structure to answer similarity searches, the distance function need to be modified. The distance function aids to calculate the (dis)similarity of an indexed object in the database with the query point. Since traditional metric distance functions, such as Euclidean Distance, were proposed with a single query point in mind, hence they need to be modified and the correctness need to be proved for multi-point queries. A distance function is correctly defined if after searching the metric space (containing the indexed data objects), based on the similarity score produced by the distance function, the top k results are indeed the k most nearest objects to the query point in the entire database. There are two basic search paradigms implemented by any index structures viz. range search and k-NN search. For the imprecise nature of multimedia query, k-NN search is preferred over range search since determining the range of an imprecise query is rather error-prone while simulating CBIR in the search routines for an index structure indexing multimedia data.

The k-NN search of AH-Tree follows the classical branch and bound technique [181] and needs to determine (i) the distance between the query object and the image object in the leaf nodes and (ii) the distance between the query object and the intermediate index nodes. Since the metric space of our index structure is Euclidean, hence each intermediate node which serves as a bounding region for the child nodes is represented by a sphere with a centroid C (an image object with a routing role) and a radius r (covering radius). Each data node (at the leaf level) is represented by a centroid with covering radius equal to zero. The distance function, DIST, is used to calculate the (dis)similarity between the query object and the index tree nodes (both intermediate and leaf nodes). Generalizing the definition of MINDIST in [181], the DIST(O,P) in AH-Tree of an object O from

Table 7.1: Refined_k-NN search algorithm

```
Refined_k-NN-Search(Q, N_child, r(Q), aff) { //Metric Search.
  Affinity_Promotion_RefinedQueries( ); //promotion of affinity value.
  if (((aff(O_r, Q_1) ≠ 0)|||((aff(O_r, Q_2) ≠ 0)||| ... ||| ((aff(O_r, Q_n) ≠ 0)))
   { //affinity value available for at least one of the query points.
     if (O_r is a routing object){
      ∀ O_r in N_child do: {
          if ((Σ_{i=1}^{n} W_i|C - F_i|^2 - r) ≤ r(Q)+r(O_r)) && ...
             ((aff(O_r, Q_1) ≥ aff)|||(aff(O_r, Q_2) ≥ aff)|||
              |||(aff(O_r, Q_n) ≥ aff))) {
                //update and reshuffle the k least distance values by inserting
                //(Σ_{i=1}^{n} W_i|C - F_i|^2 - r) in the correct position.
                aff = max_{i=1}^{n} (aff(O_r, Q_i));
               Refined_k-NN-Search(ptr(T(O_r)), Q, aff);
                //T(O_r): pointer to the subtree.
              }
        }//end of search for O_r
         //satisfying metric condition.
     }//end of search for all O_r in N_child.
  }//end of internal node search of the
   //metric tree.
  elseif (O_r is a leaf object){
        if ((Σ_{i=1}^{n} W_i|C - F_i|^2) ≤ r(Q)+r(O_r)) &&
           ((aff(O_r, Q_1) ≥ aff)&&(aff(O_r, Q_2) ≥ aff)&& ... &&(aff(O_r, Q_n) ≥ aff))) {
           //add to the result set.
           //update and re-shuffle the k least distance values by inserting
           //(Σ_{i=1}^{n} W_i|C - F_i|^2) in the correct position.
           aff = max_{i=1}^{n} (aff(O_r, Q_i));
     }
   } //end of search for query object with affinity.
 else {
    Refined_k-NN-Search with the absence of the affinity
    comparison;
  } //end of search for query object without affinity.
} //end of Metric Search Subroutine.
```

query object P is defined as:

$$DIST(O, P) = |C - P|^2 - r. \tag{7.2}$$

If, the object is a leaf node, the covering radius is zero and Equation 7.2 is reduced to:

$$DIST(O, P) = |C - P|^2 \tag{7.3}$$

The above technique can be extended to query expansion for metric space with a multiple point refined query as follows: The DISTMULTI between an intermediate object O and an expanded query Q is defined as:

$$DISTMULTI(Q, O) = \sum_{i=1}^{n} W_i |C - F_i|^2 - r, \tag{7.4}$$

Where F is a feature vector consisting of features of each image object the user has marked to be a potential query object or relevant to a submitted query, n denotes the number of marked query objects and W corresponds to a set of weights attached to each returned additional query point.

As mentioned above, the correctness of the modified distance function or DISTMULTI with the multiple query points should be proved. It is correct if indeed it provides the k closest results in response to a similarity query, in each iteration. Thus, to ensure that there will be no false dismissal, it should be proved that DISTMULTI(O, Q) lower bounds $D(T, Q)$, where T is any node of the subtree of O.

Lemma 7.1.1 *DISTMULTI for a multi-point query is correct iff DISTMULTI(O, Q)* $\leq$ *D(T, Q) for any node T in the subtree of O.*

Proof. Let, O be an intermediate node of the metric tree. Since, T is any object under O, it belongs to the subtree/covering tree of O. Since all nodes of the covering tree are within its bounding radius, we have:

$$D(P_i, T) - r(O_T) \geq D(P_i, O) - r(O_o) \tag{7.5}$$

Where P_i is the i^{th} query point and $r(O_T)$ and $r(O_o)$ are the covering radii of the subtree T and O respectively. Which implies,

$$D(P_i, O) - r(O_o) \leq D(P_i, T) - r(O_T) \tag{7.6}$$

Which implies,

$$\sum_{i=1}^{n} W_i D(P_i, O) - r(O_o) \leq \sum_{i=1}^{n} W_i D(P_i, T) - r(O_T) \tag{7.7}$$

Thus,

$$DISTMULTI(O, Q) \leq D(T, Q) \tag{7.8}$$

□

The summation of the distances over n refined query points ensures that the pruning is performed based upon the collective impact of each of the refined query points. The effect of each of the refined queries on the collective MINDIST function is determined by the weights assigned by the relevance attached by the user in each iteration.

7.1.3 Similarity Search With Hybrid Query Refinement Model

The above discussed technique of hybrid query refinement is embedded in the metric space (distance-based index structure) of AH-Tree during similarity search as discussed in Refined_k-NN-Search algorithm in Table 7.1. The Refined_k-NN-Search explores the image database with a refined query Q, consisting of multiple image points marked as relevant by the user, and a refined affinity matrix to return k image points most similar in both feature-level similarity (computed by DISTMULTI) and high-level relationship (computed from the promoted affinity values). A priority queue of the sub-trees (for which at least one qualifying object has been found) is maintained and elements (intermediate nodes which are the roots of the subtrees) are popped from its top and checked for similarity criteria (both in terms of low-level features and high-level semantic relationship) until the priority queue consists of leaf nodes which are the k nearest neighbors

of the multi-point query Q. During a k-NN search of the metric space, for each non-leaf node, the DISTMULTI is calculated between the node and each of the query points using Equation 7.4. If the DISTMULTI of the node is greater than the distance of the current k^{th} neighbor and does not have an affinity with any of the query point greater than or equal to the required affinity, the node is pruned. Otherwise the examined node is added to the priority queue and the entire queue is sorted based on the DISTMULTI function. Similarly, for each leaf-node, DISTMULTI is calculated using Equation 7.4 with r set to 0 since for leaf nodes there is no bounding region and the covering radius is 0 [45]. Here, the affinity condition is an *AND* (Equation 7.9) rather than an *OR* (Equation 7.10) as used in the intermediate node evaluation.

$$((aff(O_{leaf}, Q_1) \geq aff)\&\&(aff(O_{leaf}, Q_2) \geq aff)\&\& \ldots \&\&(aff(O_{leaf}, Q_n) \geq aff)) \tag{7.9}$$

$$((aff(O_{router}, Q_1) \geq aff)\|\|(aff(O_{router}, Q_2) \geq aff)\|\| \ldots \|\|(aff(O_{router}, Q_n) \geq aff)) \tag{7.10}$$

Where aff(O_{leaf},Q_n) and aff(O_{router},Q_n) are the affinity between the leaf node and the intermediate node with the n^{th} query point respectively. Q is the multi-point query and aff is the required affinity value.

This condition is utilized to push all the intermediate nodes into the priority queue with even a slight possibility to match the high-level similarity with at least a part of the multi-point query (at least with one of them). Thus, an optimistic guess is done to avoid any false dismissal. But for the leaf nodes, the final result set is determined based on the distance and the affinity criteria. Hence, the conditions are made more stringent and only those image points from the database are chosen which have the required high-level closeness with every query point. If there are not k image points satisfying the criteria, the refined query is re-executed with a more flexible *OR* condition for the leaf nodes and the intermediate nodes. Furthermore, it should be noted that based on the weights

attached to each query point by the user, DISTMULTI is actually a weighted summation of the corresponding distances (bounding region or point). These weights (in the range of 0-3 as discussed in Section 7.2 are attached to each query point during the user feedback process.

Table 7.2: Affinity promotion for refined queries

Affinity_Promotion_RefinedQueries( );
Start from the leaf nodes of the distance-based index structure of the AH-Tree.
For each leaf node O_l:{
Set aff(O_l, Q) = $max_{i=1}^{n}$ (aff(O_l, P_i));
// Where, aff(O_l, Q) is the affinity of the leaf node with respect
// to the multi-point query Q and aff(O_l, P_i) is the affinity of
// the leaf node with respect to each query point P_i in Q.
}
Traverse the tree bottom-up.
For each intermediate node O_r:{
aff(O_r, Q) = $max_{i=1}^{n}$(max(*aff(t_a, q_i)*, *aff(t_b, q_i)*, ..., *aff(tz, q_i)*,
max(*aff(t_a, q_{i-1})*, *aff(t_b, q_{i-1})*, ..., *aff(t_z, q_{i-1})*)) ;
// Where, aff(O_r, Q) is the affinity of the intermediate node
// with respect to the multi-point query Q consisting of q_i, aff(t_a, q_i)
// is the affinity of each children t_a with each query point q_i and
// z is the number of children of O_r.
}

The high level semantic relationships among the images in the form of affinity values cannot be embedded in the metric space as it makes the distance functions arbitrary [45]. Instead the affinity relationship is introduced in the metric structure prior to issuing a query, through a novel affinity promotion technique. In the affinity promotion routine, the affinity values are promoted from the leaf level to the intermediate nodes up to the root and thus distribute it through out the tree structure so that each node has an affinity value with respect to the query. Since the query representation is modified, the affinity promotion technique should also be changed from to support multiple points in a query.

For a query space consisting of n query points $q_{i=1}^{n}$, the affinity relationship of an intermediate node p with child nodes a and b is set as:

$max_{i=1}^{n}(\max(affinity_{a,q_i}, affinity_{b,q_i}), \max(affinity_{a,q_{i-1}}, affinity_{b,q_{i-1}}))$,

Where $affinity_{a,q_i}$ represents the affinity value of child node a with the query point q_i. The above representation sets up a routine whereby the maximum of the affinity values among the values between all the child nodes with all the query points is promoted as the affinity value of the parent node p with respect to the multi-point query q. It is done by an iterative process whereby in the i^{th} iteration the maximum of the affinity between all the children with the i^{th} query point is compared with the maximum of the affinity between all the children with the $(i-1)^{th}$ query point. The maximum of these two values is set as the affinity value at the i^{th} iteration and the process continues. Also, it should be mentioned here that for simplicity of representation, the above formula is presented for an intermediate node with only two children. But for implementation purposes, it could be extended for n number of child nodes without any loss of generality. This formula ensures that if there is any candidate node in the subtree which has an affinity value with each of the query points greater than or equal to the required affinity, the parent node is traversed and there is no false dismissal. It also ensures that if none of the child nodes has the required affinity, the parent node can be pruned altogether without any further investigation. The method executes bottom-up i.e. in the AH-Tree, first, for each leaf node, the affinity value between it and each of the query points are calculated and the maximum among them is determined. This maximum affinity value is assigned to each leaf node and then the index tree is traversed one level up to promote these affinity values to the intermediate nodes till the root is reached, using the above formula. The algorithm employed for the multi-point query promotion technique is presented in Table 7.2.

7.2 Empirical Study and Evaluation Metric

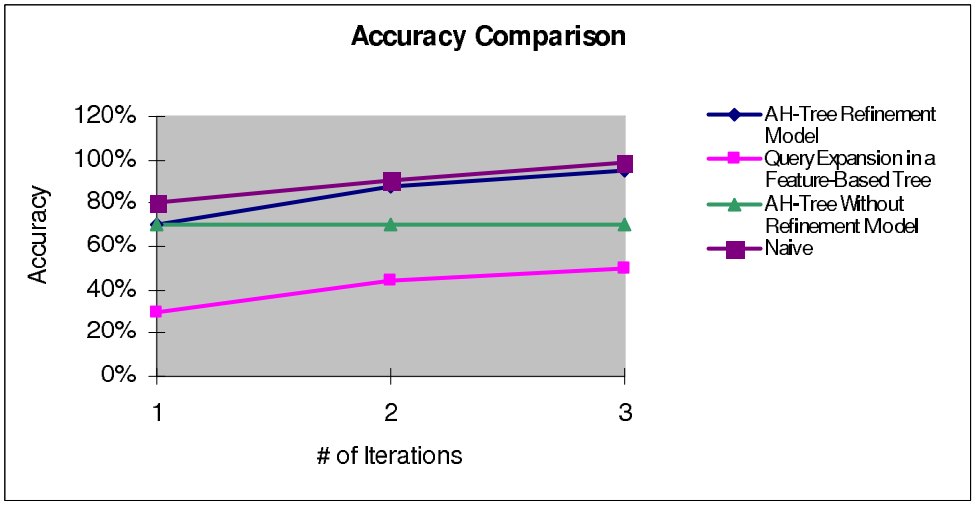
In the experiments, the image database used consists of $10,000$ color images from Corel dataset belonging to 72 semantic categories. The system allows the user to rank query

results as 0 (Not Relevant), 1 (Very Close), 2(Perfect), and 3(Set as new query). These weights are utilized in formation of the refined queries. Extensive experiments are performed with 3 rounds of iteration for 10 query images randomly picked from the database. Four systems viz. AH-Tree Refinement Model, Feature-Based Refinement Model, AH-Tree Without Refinement Model and a Sequential Search Model(one which doesn't have any index structure but searches through the entire image database in terms of both low-level feature-wise similarity and high-level affinity relationships), are compared with one another. The comparison is performed in terms of 4 criteria viz. Accuracy, Computation Time, F1 Score and Number of Distance Computations (required to determine the feature-level similarity). The accuracy is measured as the percentage of retrieved results that were marked relevant by the user, computation time is the time taken to execute the query, F1 Score can be considered as the weighted average of the precision and recall and is expressed in Equation 7.11 and number of distance computation measures the computation overhead contributed by each model during the similarity calculation using distance functions (like Euclidean Distance Function).

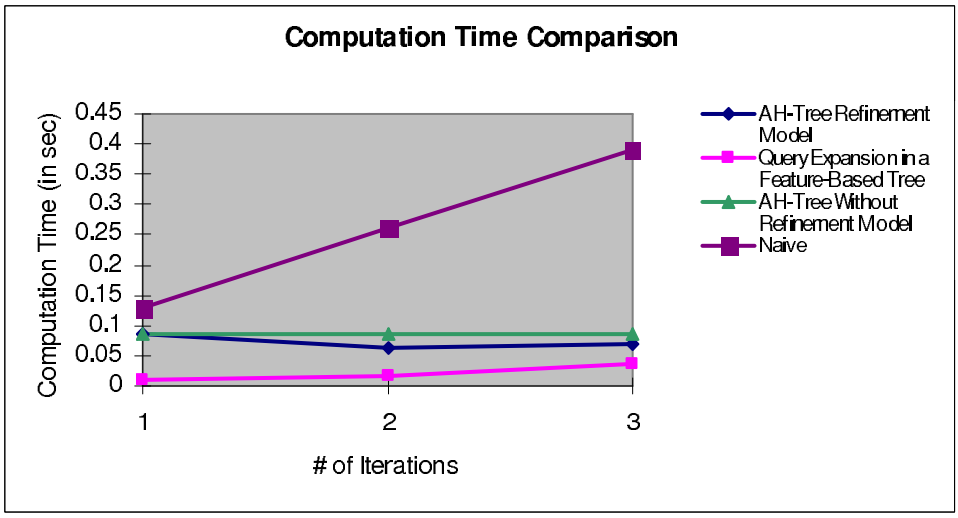
$$F1\ \ Score = \frac{2 \times p \times r}{p + r} \tag{7.11}$$

Where, p is the precision and r is the recall.

The experimental results for 3 iterations averaged over 10 random queries are summarized in Table 7.3 and their graphical representations are presented in Figure 7.1, Figure 7.2 and Figure 7.3. It can be seen from the results that the Naive system, having no index structure at all, performs the best in terms of Accuracy and F1 Score (since it searches through the entire database to provide the result) and thus obviously performs worst in terms of computation time and number of distance computations. On the other hand the feature-based index structure using a RF-based refinement model produces the best results in terms of computation time but the worst in terms of F1 Score (the feature-level weights failing to capture the users' similarity concept in this case where it has been seen that the query images are rather hard to distinguish in terms of low-level features alone).



(a)



(b)

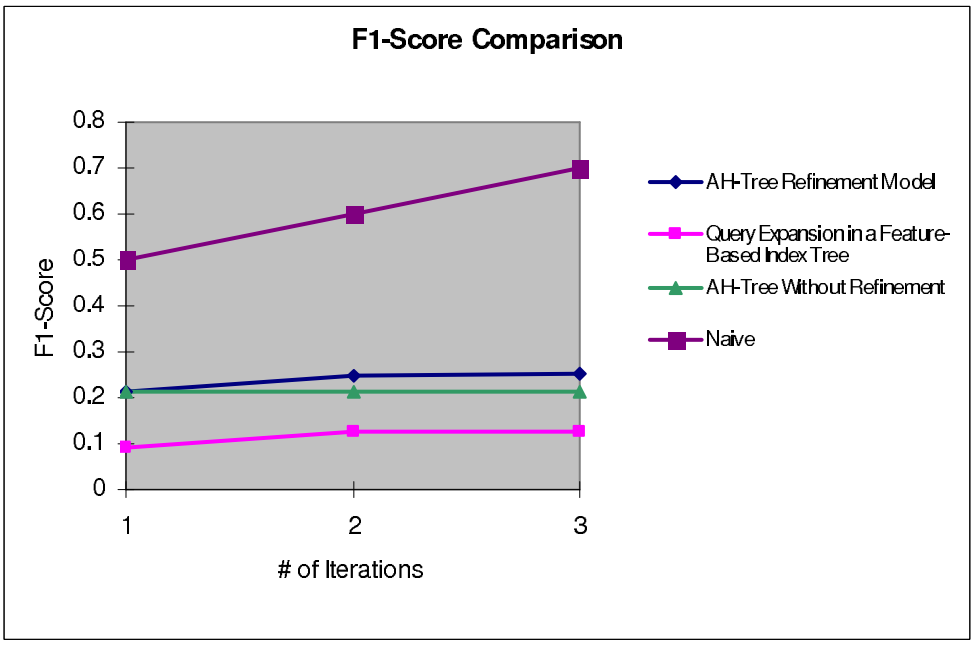
Figure 7.1: (a) Accuracy compared over three iterations, (b) Computation time compared over three iterations.

AH-Tree Refinement Model has a computation time far less than the Naive Model but the computation time is greater that the Feature-Based Index Structures Refinement Model (since an additional similarity factor, the affinity relationship need to be considered). The AH-Tree without any refinement model obviously has a fixed value for each of the criteria for all the three iterations due to the lack of any refinement model. From all the different criteria, it becomes rather difficult and confusing to determine which is the best model. Hence, an aggregate model score is proposed in terms of computation time (the main reason to introduce an index structure) and the F1 Score (the main reason for the requirement of an efficient refinement model). The main purpose behind proposing an efficient multidimensional index structure supporting multimedia retrieval strategies is twofold. First, to reduce the computation overhead and second, to produce query results as close to human perception as possible.

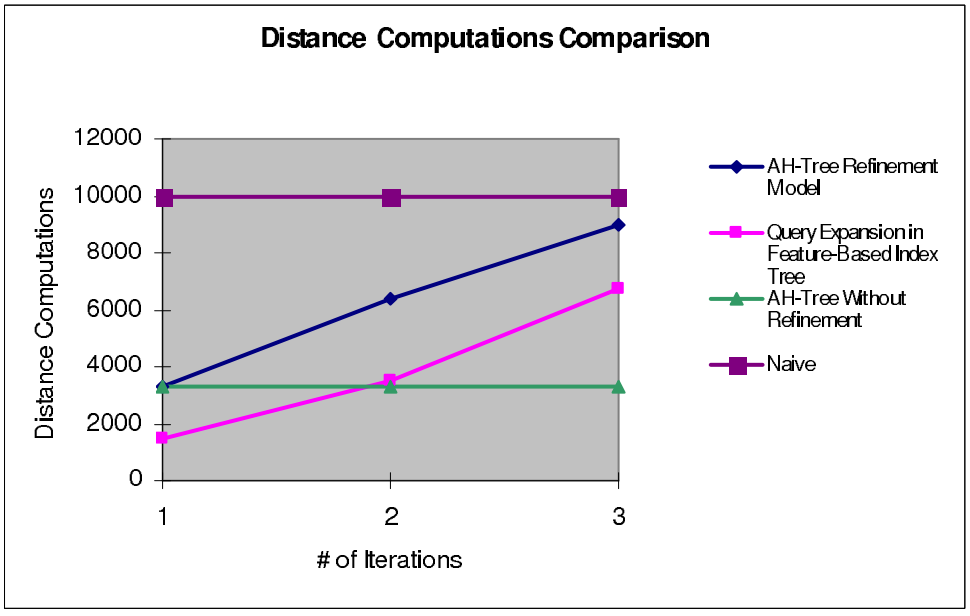
Thus to compare different retrieval models (with and without index structures and refinement strategies) and justifying the need of an efficient index structure as well as a good refinement model, a metric should be formulated that will compare the models in terms of both the specified factors and thus help the users choose the appropriate method depending upon his/her need. A cost metric is proposed, called the Model_Score (expressed in Equation 7.12 in terms of computation time and F1 Score), to be utilized in comparing the different systems as discussed in the next subsection.

Evaluation Metric

This metric is determined with the consideration that the best model will be the one with minimum computation time and maximum F1 Score. Thus, the model score of a particular system is devised as the product of the inverse of its deviation from the maximum F1 Score (among all the models) and minimum computation time (among all the models). Thus, $\frac{T-T_{min}}{3\times\sqrt{(\sum_{i=1}^{n}(T_i-T_{min})^2)/n}}$ determines the deviation of the computation time of a model from the best computation time (minimum) and produces a normalized error



(a)



(b)

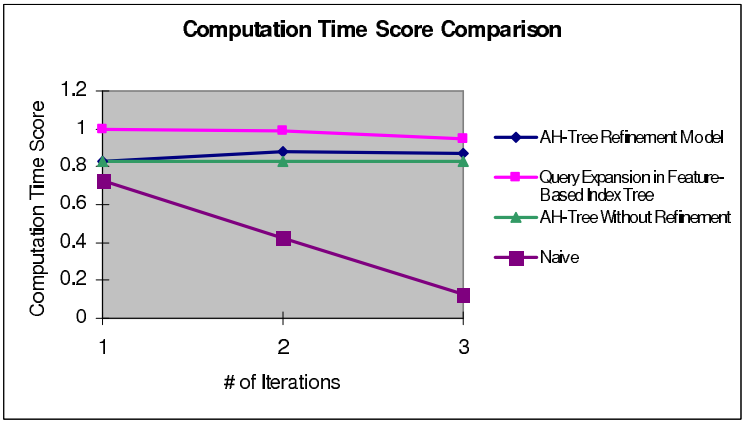
Figure 7.2: (a) F1 score compared over three iterations, (b) Number of distance computations compared over three iterations.

value. The greater the error value, farther is the computation time from the best possible computation time and lesser should be the computation time score. Thus, the value is subtracted from 1 and inversion is achieved. The same approach follows for determining the second part of the product, $\frac{F-F_{max}}{3\times\sqrt{(\sum_{i=1}^{n}(F_i-F_{max})^2)/n}}$, with the only difference being the fact that now the best possible F1 Score is the maximum of all the available F1 Scores. Hence, the normalized error of the F1 Score with the best possible value for a particular model is determined and is subtracted from 1 to get the actual score. The value 3 is used as a factor during determining the normalized error following Gaussian Normalization method, where using a factor of 3 increases the percentage of the probability of a value to lie in the range -1 and 1. Thus, greater the model score, better is its usefulness as multimedia retrieval framework. The current score is developed giving equal weights to the computation time and the F1 Score. But, depending upon users' prerogative, weights can be adjusted between them to modify the score.

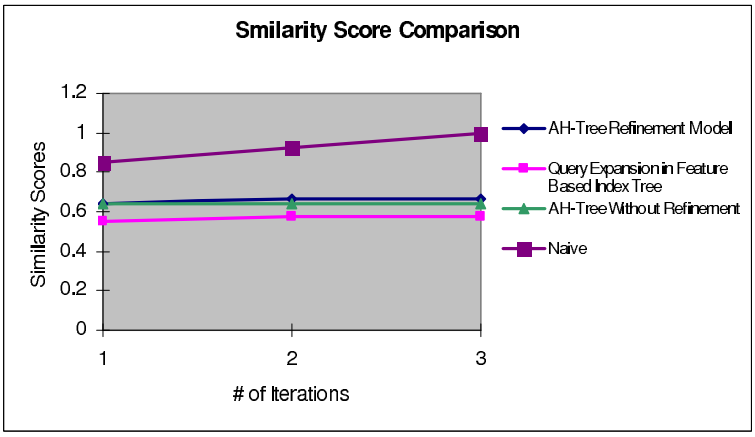
$$Model\_Score = \left(1-\frac{T-T_{min}}{3\times\sqrt{(\sum_{i=1}^{n}(T_i-T_{min})^2)/n}}\right)\times\left(1-\left|\frac{F-F_{max}}{3\times\sqrt{(\sum_{i=1}^{n}(F_i-F_{max})^2)/n}}\right|\right) \tag{7.12}$$

Where, T is the computation time of the particular model, T_{min} is the minimum computation time among all the models considered, F is the F1 Score of the particular model, F_{max} is the maximum F1 Score among all the models considered, and n is the total number of models.

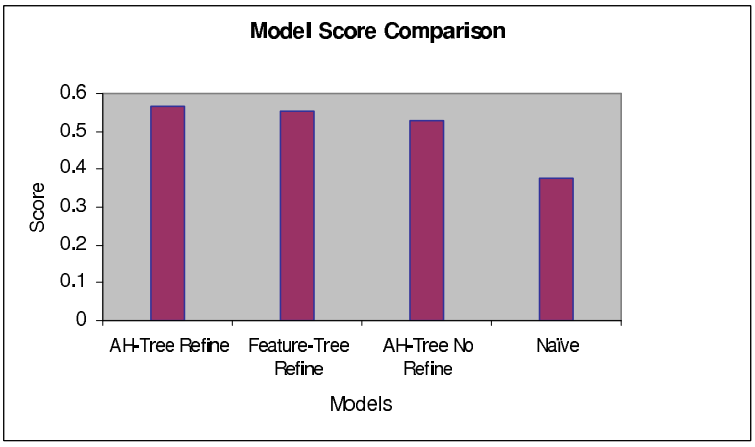
The Model Score is computed for four different frameworks as presented in the last two columns of Table 7.3 and graphically in Figure 7.3(c). From them, it can be concluded that the proposed query refinement approach on a distance based index structure has distinctively better performance than a framework without any refinement method and the sequential search framework. It has comparable and slightly better performance than the framework with query refinement approach on a feature based index structure. Thus, the proposed approach successfully achieved the two important goals of this research viz. (i) to develop a query refinement model for distance based index structure comparable



(a)



(b)



(c)

Figure 7.3: (a) Computation time score compared over three iterations, (b) Similarity score compared over three iterations, (c) Average model score.

in computation cost to the existing approaches for feature based index structures (ii) improve the relevance of query results for scenarios where the low-level feature similarity do not follow the same pattern as the high-level semantic similarity (this is demonstrated by the higher F1 score for our proposed approach as compared to the query refinement approaches like [37]). It can be concluded from the experimental data analysis that the proposed method has potential of future extension and can be utilized in other genres of multimedia retrievals like content-based video retrieval.

7.3 Conclusion and Future Work

In this chapter, a technique to embed query refinement methodology into the metric space of a multidimensional distance based index structure is proposed. The refinement technique is mainly developed to enable a distance based index structure support CBIR with user feedback efficiently and improve query results at each iteration. The refinement model utilizes the query expansion approach and proposes ways to introduce multi-point queries into a metric space. It proposes techniques to not only refine the low-level feature space but also to refine the high-level image similarity based on user feedback and use them seamlessly in the index structure during similarity search routines. Additionally, a cost metric called the Model Score, is proposed to determine the overall performance of a multimedia data retrieval framework in terms of computation time and F1 Score. As future work, it is planned to introduce this query refinement model to the generalized index structure, GeM-Tree [47] to refine queries and support content based information retrievals with relevance feedback for both images as well as videos.

Table 7.3: Experimental results

	Iterations	Accuracy (in %)	Computation Time (in sec)	F1 Score	# of Distance Computations	Model Score	Average Model Score
AH-Tree Refine	1^{st}	70	0.086	0.215	3311	0.529	0.564
	2^{nd}	88	0.064	0.25	6361	0.584	
	3^{rd}	95	0.068	0.253	8966	0.580	
Feature Tree Refine	1^{st}	30	0.011	0.092	1440	0.548	0.552
	2^{nd}	44	0.017	0.125	3475	0.565	
	3^{rd}	50	0.035	0.127	6769	0.543	
AH-Tree No Refine	1^{st}	70	0.086	0.215	3311	0.529	0.529
	2^{nd}	70	0.086	0.215	3311	0.529	
	3^{rd}	70	0.086	0.215	3311	0.529	
Naive	1^{st}	80	0.13	0.5	10000	0.617	0.377
	2^{nd}	90	0.26	0.6	10000	0.392	
	3^{rd}	99	0.39	0.7	10000	0.122	

CHAPTER 8

GENERATING SOCIAL NETWORK PREVIEWS USING GRAPH SIMILARITY

8.1 Introduction

With the increased popularity of social networking applications such as Facebook [73], Twitter [210] and Linkedin [141], there is an explosion in the number of users interacting with each other using these tools. This huge amount of dynamic information is a desirable platform for social analysis to understand the nature of social ties, the characteristics of a social network, and the behavior of its members. Such studies promise deep insight into the social behavior of the users; such as their preferences, their interaction patterns, and the types of users a particular social network attracts. Gaining such insights in turn help to better design these applications, to cater to the users' requirements. Not only does social network application designs benefit from it, but other areas such as organizational strategy making and marketing policies are also helped by such knowledge.

Additionally, in the course of this research, it was identified that multimedia data management frameworks can benefit largely from the intelligent utilization of social network representations and analysis. With the rising use of social networking tools, information retrieval can no longer be considered a solitary task. Rather, people constantly collaborate with one another while searching and retrieving information. Since, multimedia data (such as images and videos) carry more information than text-based based data; it is fast becoming a preferred medium of communication. For example, people are increasingly sharing videos from Youtube in their social networks on Facebook. The number of users is exploding and so is the use of multimedia data. Thus, there is a pressing need to manage and organize these data efficiently based on their behavior and mutual relationships while considering different aspects such as their low-level representations, their semantic interpretation as well as the social network relationships

determined from their sharing and usage patterns. Such organization approach will in turn help in improving the quality of the retrieval results. For example, currently if a user needs to search a video of interest from Youtube, the search is mainly performed based on the keywords that have been attached to the videos. Thus, the quality of the search result depends largely on the consistency and reliability of the keywords despite of the fact that the characteristics of the data itself or the behavior pattern of the user (to be obtained from his social network) can provide useful information to improve the quality and expedite the search and retrieval process. The evolving relationships among multimedia data in a collaborative environment can be modeled using a social network graph representation where the multimedia data themselves behave as actors forming their own social networks depending upon the behavior pattern of the users. Analyzing these social networks formed by the data, or more specifically the *Data Networks*, provide valuable insights into the data characteristics. This in turn would aid in designing the management and retrieval frameworks of these data. But, the major challenge to analyze these information is its sheer size. An important aspect of any social network analysis is visualizing the information and analyzing the relations from the structures. An overall structural view of a social network provides immense while easily derived knowledge, that an user process with his/her cognitive intuition. As pointed out by Burt in [29], the holistic structural analysis of a social network is better than an atomistic analysis as the former explicitly considers the social context within which actors make evaluations. It is well accepted that individual behavior and opinions are rooted in the structures to which people belong [104]. But, if a social network structure is too big, its overall structural characteristics is no longer easily discernible. The cognitive load imposed on the users analyzing it, is also increased. Thus, a much desirable option is to obtain a snapshot or preview of the original social network structure which would contain a fewer number of involved actors and ties but would carry and preserve the overall characteristics of the original structure.

As pointed out in [129], reducing the number of visual elements improves the clarity as well as the performance of the layout and rendering of a graph structure. The approach used most frequently to reduce the number of network elements is clustering [183] and using clustered graph. A clustered graph $C = (G, T)$ consists of an undirected graph $G = (V, E)$ and a rooted tree T such that the leaves of T are exactly the vertices of G [74]. As per the traditional definition, Clustering is the process of discovering groupings or classes in data based on chosen semantics [104]. For graph structures, clustering is concerned with grouping structurally similar components together, which is also called natural clustering [183]. Using semantic data associated with the graph members (actors of the social networks) for grouping purposes lead to content-based clustering [158]. Once disjoint clusters are identified, the number of elements to be displayed are reduced by representing a group of elements with a single element (specifically, the cluster center). This approach provides an overview of the entire dataset while retaining a contextual similarity. Most algorithms, such as [7][66] look for a balance between the total number of representative clusters and the number of nodes within each cluster. A definition of how edges between the clustered nodes should be induced is presented in [111]. Once cluster nodes are connected via edges, the clustered graphs are visualized as: (i) ghosting, (ii) hiding and (iii) grouping [129].

There are several challenges associated with representing a large social network structure with clustered graphs. Firstly, as with any clustering approach, determining the optimum number of clusters to represent a dataset perfectly, is a challenge. Secondly, picking up the cluster centers is a difficult task. An ample amount of domain knowledge is necessary to isolate the nodes, to be used as initial cluster seeds, from a large data-pool. Finally, there is no clearly described technique by which the natural structure of the original graph is preserved in the representative clustered graph. Thus, to represent an information-rich social network with a clustered graph, one needs to analyze the domain and the dataset closely to find out the specific characteristics of the dataset.

Once the typical properties of the original graph is identified, the different steps involved in generation of clustered-graphs (such as determining the node metrics, specifying the node connectivity criteria, picking up the cluster centers) should be optimized based on those information. Hence clearly, it is difficult to have a generic and precise approach of developing clustered graphs representing large social network structures with fewer nodes and edges while preserving the overall network characteristics of the original graph.

In this chapter, we propose a social network preview technique utilizing graph similarity. The proposed technique represents a large social network graph with fewer nodes while preserving the structural characteristics of the original graph. It employs a coupled edge-node score along with a semantic score to determine the similarity between a representative node and the nodes of the original graph. An assignment algorithm (Extended Munkres Algorithm) is then utilized to assign each of the m candidate nodes to one of n original nodes. The assignment ensures to maximize the total similarity score between the n nodes of the original social network graph with the m nodes of the representative graph, where $m \ll n$. An edge connectivity criteria is proposed for the representative graph to preserve the edge connectivity pattern of the original graph. The proposed social network preview technique is different from the existing clustered graph approach in the following ways:

1. The main step of the clustering method is to group similar actors (or actors with similar behavior) together. Defining a precise similarity criteria for a particular social network is often challenging without detailed pre-analysis of the dataset. The definition of similarity between the actors depends on the characteristics of the particular social network. For example, in case of a social network developed on e-mail communications, similarity between the actors might be based on the frequency of communication among members with *similar* rank or might be defined based on the *inter-rank* communication pattern. Clustering can be performed based on either of these criteria but the resulting clustered graphs will have very different

structures and characteristics. Thus the precise similarity criteria need to be determined from the original graph, before performing clustering, to retain the original characteristics in the representatives. The proposed approach does not need such pre-meditated information as it uses a generic similarity score that only considers the overall structure of the given social network.

2. Structural similarity maximization between the original and the representative graph is an important step of the proposed approach. Thus, an overall structural resemblance between the original and the representative graphs are guaranteed to some extent. No such guarantee can be expected from the clustered approach unless a definite, pre-meditated clustering and edge connectivity approach is defined based on domain-knowledge of the dataset.

Since Centrality [88][89][90] has been regarded as an useful measure of the overall structure of a social network graph [223][70], we use an evaluation score based on it to find how close the representative graph is to the original graph. From an extensive experimental analysis, it has been seen that the proposed approach generates representative graph structures closer to the original graphs than the clustered approach.

The rest of the chapter is organized as follows: Section 8.2 presents the proposed social network preview technique with a detailed description of each step. This is followed by Section 8.3, which presents the Evaluation Score utilized to analyze the generated representative structures. Section 8.4 describes the Data Set used in the experiments for verifying the proposed methods. An extensive experimental results and analysis is presented in Section 8.5. Section 8.6 discusses briefly how Multimedia Data Networks can be generated and analyzed followed by a brief conclusion and discussions of future research direction in Section 8.7.

8.2 Generating Social Network Preview

A social network is popularly represented with a graph structure where each actor corresponds to a node and each connection/relation between its actors corresponds to an edge in the graph. For the rest of the chapter, the term *graph* and *social network* is used interchangeably. To generate a social network preview, four main steps are followed. The process starts with an initial node sampling technique whereby m nodes are selected to represent the original social network having n nodes ($m \ll n$). The sampling process can be random or pre-meditated. The second step is to have a similarity score between the nodes of the original social network graph structure and the selected representative nodes. The similarity score determination further involves two steps. First, structural metrics are assigned to each node based on the position of the particular node in the entire network and its relationship with its neighbors. Second, semantic similarities between the nodes of the original graph structure and the selected nodes are calculated. We use an aggregate of the coupled edge-node similarity calculation technique [228] (for determining structural similarity) along with an Euclidean Distance measurement (for determining semantic similarity) to obtain the final score between a pair of nodes. The similarity is represented as a nXm matrix. The third step is to optimally assign each of the the m picked nodes to one of the n original nodes so as to maximize the overall similarity between the nodes of the original and the representative graphs. We use an extended version [30] of a combinatorial optimization algorithm called Hungarian or Munkres algorithm [137][138][159] to solve the assignment problem. The final step is to generate the representative preview graph from the representative nodes. In this step we need to determine whether two nodes should be connected via an edge depending upon the relation of the corresponding nodes in the original graph. We decide whether to connect two nodes based on a shortest path length approach. The different steps of the algorithm to generate a social network preview, as discussed above, is summarized in Table 8.1. The input to the algorithm is the original social network structure with n nodes and the

output is the representative social network structure with m nodes. The two methods used in the algorithm, namely *Node_Assignment*() and *Preview_Graph_Generation*() is detailed in Table 8.2 and 8.3 respectively.

Table 8.1: Generating social network preview

Social_Network_Preview_Generation(SN**)** {
//SN is the original social network graph with n nodes.
Select m nodes from SN based on centrality distribution;
for k iterations {
For each node pair (n, m) {
Set $Struct\_Sim[n, m]$ = edge-node coupled similarity between n and m;
//$Struct\_Sim[n, m]$ *stores the similarity between nodes n and m computed*
//iteratively by considering behavior of neighboring nodes and edges.
}}
For each node pair (n, m) {
Set $Sem\_Sim[n, m]$ = semantic similarity between n and m;
//$Sem\_Sim[n, m]$ *stores the semantic similarity between nodes n and m computed*
//utilizing Euclidean Distance on the semantic vectors.
}
Set $C = W_i\ Struct\_Sim + W_j Sem\_Sim$;
//C is the combined similarity, W_i *is the weight attached to Struct_Sim*
//and W_j *is the weight attached to Sem_Sim.*
Set A = Node_Assignment(C);
Set G_{disp} = Preview_Graph_Generation(A);
//G_{disp} *is the representative preview graph.*
output: G_{disp};
}

8.2.1 Selecting Nodes

There are several approaches towards sampling Social Networks. All of them have a basic goal: to preserve the characteristics of the original social network structure in the sampled network. There are different sampling schemes, namely simple random, stratified, probability proportional to size, systematic, cluster and multistage [87]. Broadly, sampling methodologies can be categorized as (1) Random and (2) Pre-Meditated. Ran-

dom sampling is equivalent to selection approaches that are dictated by probability laws. As pointed out in [86], random samples consist of observations from a probability distribution belonging to some specific parametric family of distributions. Pre-Meditated approach, as the name suggests, picks up specific samples from a given population and there is no randomness or probability associated with it. It should be pointed out here that the social network sampling discussed here is different from information/population sampling methods [92] [93] utilized while *collecting* social network data. In the former, characteristics of the overall social network is already identified and attempts are made to retain those traits in the sampled population but for the later, the characteristics is identified progressively during the sampling process as new samples and relationships are encountered. In the proposed approach, the sampling step is designed as a plug-in and any sampling method can be defined by the user without any loss of generality.

Random sampling of social networks can be extended to random graphs [108] based on graph theory and statistical probability distributions on sets of graphs. When a finite graph (a graph with a fixed population) is investigated by sampling and observing only a part of it, a typical class of graph sampling called subgraph sampling can be implemented. For social network preview, such subgraph sampling could also be utilized. The major subgraph sampling methods are [21][31][75]. Inclusion probabilities are utilized in these sampling approaches. If S be a subset of V (where, V is the original population with n members and S is the sample population with m members) selected with a random sampling having inclusion probabilities $P(i \epsilon S) = \pi_i$ and $P(i \epsilon S, j \epsilon S) = \pi_{ij}$, for *simple random sampling*, $\pi_i = \frac{m}{n}$ and $\pi_{ij} = \frac{m(m-1)}{n(n-1)}$, $i \neq j$. The subgraph sampling can be further classified as *induced subgraph sampling* [84], *star sampling* [32], etc.

In this chapter, two sampling techniques are utilized: a simple random sampling and a centrality-based pre-meditated strategy. First, the value of m should be determined. The choice of m depends on the data set, the kind of analysis to be performed on the social network structure and the performance of the visualization method utilized. A

simple random sampling is chosen because the subsequent steps of the preview generation algorithm attempts to maximize the similarity between the sampled nodes with the original nodes and assign them accordingly without depending upon the initial samples. The main objective of this chapter is to demonstrate the effectiveness of using the Graph Similarity, the Hungarian Assignment and a typical node connectivity approach together. Thus, to make the initial samples unbiased, a simple random sampling strategy is adopted. Definitely, as more computationally extensive and complex sampling techniques are used, the end results would benefit accordingly. In the random sampling approach used, m of the original n nodes are randomly chosen without replacements, as representatives. In statistics, a sample space is defined as the set of of all possible outcomes. In this case the sample space (S) is represented as $S = \{0, 1\}$, which denotes if a particular node is chosen (1) or not (0). Once a particular node is selected, it is eliminated from the pool and n is decremented by 1.

For demonstrating the effect of pre-meditated approach, a *stratified* sampling based on centrality scores of the nodes is utilized. The main goal is to preserve the distribution of the original social network as closely as possible in the samples so that even if a social network has a typical trait visible in only a certain population type, the chances of capturing it in the sampled pool is increased while using *stratified* sampling. All the original nodes are first sorted based on their degree centrality and are divided into strata. Next equal number of nodes are selected from each strata. The stratification of the original dataset is done based on the centrality values because centrality has been successfully utilized in the past to study the holistic behavior of social networks in several complex social systems [56][16]. However, since a great deal of subjective interpretation is involved in the definition of the *special trait* of a particular social network and is by itself imprecise, no sampling method can be considered to be perfect or universal.

8.2.2 Determining Similarity

Graph Similarity has been a research topic for a long time. There are several approaches to compute Graph Similarity. Isomorphism between a pair of graph is perhaps the basic interpretation of Graph Similarity. Two graphs are *isomorphic* if there exists a bijective function between the node sets of each graph such that two nodes are connected by an edge in one graph iff their images under the bijection are connected too. However, determining if two graphs are isomorphic is a NP-complete problem. There are several other techniques where a global approach is undertaken to determine the similarity between pair of graphs such as error correcting graph matching [26], maximum common and minimum common subgraph matching utilizing edit distance technique [27], etc. A separate genre of graph similarity approach is based on considering the local characteristics of the nodes and the edges. Here, the similarity between a pair of nodes or edges is determined based on the behavior of the neighboring nodes and edges respectively. Such local iterative approaches has been found useful in the hub and authority scoring employed in web searching [130] [22]. Other algorithms based on similar idea are Similarity Flooding [156], where two different database structures are matched by representing each database as graph with labels on the nodes (database elements) and edges (relationships between elements); SimRank [118], where a graph's self-similarity (that is the similarity between pairs of nodes within a single graph) is calculated; determining the evolutionary relationships between species [105] using enzyme graphs; etc. However, all of these approaches consider only the nodes and its neighbors in order to compute the similarity between a pair of graphs iteratively. They do not consider scoring the similarity based on the behavior of the edges. For social network graphs, the nature of relationships between the actors (graph elements), represented as edges of a graph, are as crucial as the characteristics of the actors themselves. Thus, if the edge similarity is not considered while computing the similarity scores between the graph structures, major information might be lost. Thus, in the proposed approach, a coupled node-edge similarity score [228] is

adopted. The main idea behind this method is to update edge scores based on node scores and node scores based on the edge scores. The concept of edge similarity is introduced by defining a relationship between the similarity among the nodes and the edges as: *two edges belonging to two graphs are considered similar if their source and terminal nodes are similar as well.* Let, G_A and G_B be two graphs with m nodes and p edges and n nodes and q edges respectively. Let, $s_A(i)$, $s_B(j)$, $t_A(i)$ and $t_B(j)$ denote the source node and terminal node of G_A and G_B for edges i and j respectively. Let, x_{ij} denote the similarity between nodes i and j of G_A and G_B respectively and y_{ij} is the similarity score between edges i and j belonging to G_A and G_B respectively. The equation computing the edge and node scores for the k^{th} iteration can be represented as:

$$y_{ij}(k) \leftarrow x_{s(i)s(j)}(k-1) + x_{t(i)t(j)}(k-1) \tag{8.1}$$

$$x_{ij}(k) \leftarrow \sum_{t(u)=i,t(v)=j} y_{uv}(k-1) + \sum_{s(u)=i,s(v)=j} y_{uv}(k-1) \tag{8.2}$$

Here, Equation 8.1 computes the similarity score between edges i and j of G_A and G_B at the k^{th} iteration by adding the node similarity scores between the sources ($s(i)$ and $s(j)$) and terminals ($t(i)$ and $t(j)$) of the edges i and j, computed at the $(k-1)^{th}$ iteration. Similarly, Equation 8.2 computes the similarity score between nodes i and j at the k^{th} iteration by summing up edge similarity scores, obtained at the $(k-1)^{th}$ iteration, for all the edges where nodes i and j has been the source and the terminal respectively. A Frobenius norm is applied at each iteration for normalization. Equation 8.1 and 8.2 can be represented in matrix form as:

$$Y_k \leftarrow B_s^T X_{k-1} A_s + B_t^T X_{k-1} A_t \tag{8.3}$$

$$X_k \leftarrow B_s Y_{k-1} A_s^T + B_t Y_{k-1} A_t^T \tag{8.4}$$

Where, A_s and A_t are the edge-source and edge-terminus matrices of Graph A, B_s and B_t are the edge-source and edge-terminus matrices of Graph B and T represents the transpose of the corresponding matrix. Edge-source and edge-terminus matrices can be

easily derived from adjacency matrices, which are normally utilized to represent a graph structure. For $k = 1$, that is for the initial condition, both X and Y are chosen to be all-ones. The intuition behind this choice is to provide an unbiased initial condition where all the nodes and edges are considered equally *similar* to one another. The above equations are designed for *simultaneous* update of the node and the edge pairs. But, for practical implementation purpose, a *sequential* update is utilized which can be defined as:

$$y_k = Gx_{k-1} \tag{8.5}$$

$$x_k = G^T y_k \tag{8.6}$$

or as:

$$x_k = G^T y_{k-1} \tag{8.7}$$

$$y_k = Gx_k \tag{8.8}$$

Where, $G = A_s^T \otimes B_s^T + A_t^T \otimes B_t^T$, x and y are equal to $vec(Y)$ and $vec(X)$ respectively. However, for the purpose of determining the similarity between the original social network structure and its representative preview structure, only the node similarity, that is x is utilized. This is because for all the dataset we used, the nodes have identifiers but have no edge labels. Thus, for final preview, it only matters whether a particular node is used as a representative and if it is connected to other nodes by *any* edge. It is not important to find out whether a *particular* edge is utilized in the preview graph or not. Since, edge representative does not contribute towards the final preview representation, we omit using the edge similarity matrix. Nevertheless, the node similarity matrix is generated while considering the edge similarities as well.

Additionally, for each node of the original social network graph, a vector of semantic scores is defined. The length of the vector and the types of the semantic scores depend on the dataset and the application. The overall role of an individual in a social environment has often been described as an aggregation of sets of relations of various types

linking this node/actor as *ego* to sets of others [96]. In a social network structure, two elements a and b can be considered structurally equivalent if a relates to every object x of C (C is the population category to which both a and b belong) in exactly the same ways as b does. This concept of similarity between two nodes/actors of a social network can be extended in determining a *semantic similarity matrix*. For most social network structures, Centrality has been found to be an important characteristics that captures the influence and contribution of each node (member) on the overall network. Centrality is computed based on the structural property of an individual with respect to other members belonging to the same network. But it also provides information about semantic properties such as the power/importance/contribution of the individual in its social environment. Additionally, for labeled or weighted graph structures, where weights are assigned to the edges and nodes are labeled, they can be utilized as semantic score metrics as well. In this chapter, the three different centrality values (namely, degree centrality, betweenness centrality and closeness centrality) are utilized. However, other specific information, if available for the dataset to be analyzed, can be utilized as well which will enrich the semantic similarity score vector. For example, for a social network developed based on the email communications among members of an organization [131], the *rank* of the individual in the hierarchy might be utilized as a semantic metric. For instance, the *rank* might be categorized as *manager*, *director*, or *senior engineer*. The communication between individuals, analyzed based on their ranks, might present interesting social network activities. Thus, while computing the similarity between a pair of nodes or edges, considering the ranks assigned to the nodes (actors) might provide a more inclusive similarity measure.

For the *numerical* attributes (such as centrality values), popular Euclidean Distance function can be utilized to determine the (dis)similarity measure between a pair of nodes. However, for *nominal* attributes (such as rank information), definitions of (dis)similarity becomes non-trivial [55]. A commonly used approach is *overlap metric* [202], where

Table 8.2: Node assignment algorithm maximizing similarity

```
Node_Assignment(C) {
 //C is the combined similarity matrix with n rows and m columns.
  Set k = min(n, m);
  Set c_max = maximum similarity value from the matrix C;
  Replace each c_ij of the C matrix with c_max − c_ij;
  From each row subtract the row minimum;
  From each column subtract the column minimum;
  For each element of the matrix C {
   Find a 'zero';
   Set i = row containing the 'zero';
   Set j = column containing the 'zero';
   if (i^th row and j^th column do not have a marked 'zero')
    Mark the zero at the (i, j) cell as 'starred';
  }
   Set k_covered = Number of columns containing a 'starred zero';
   if (k_covered ≠ k) {
    Mark a 'zero' as 'prime' in a column having no 'starred zero';
    if (row having 'prime zero' do not have any 'starred zero') {
     while (a column has a 'prime zero' with no 'starred zero') {
      Remove marking from each 'starred zero';
      Mark each 'prime zero' as 'starred';
      Remove all the covered lines;
     }
    }
     else {
      while !(rows having 'zero' uncovered) {
       Mark this row as 'covered';
       Remove 'covered' marking from the column
       containing the 'starred zero';
      }
      Set uncovered_min = smallest uncovered value;
     }
     For each element of each row marked 'covered'
      Add uncovered_min to the corresponding element;
     For each element of each column not marked 'covered'
      Subtract uncovered_min to the corresponding element;
    }
    else {
     solution_set = cells having 'starred zero';
     //row and column values of the starred zero is the assignment pair.
    }
}
```

for two possible values v_x and v_y, the distance is assigned as *zero* when v_x and v_y are identical and *one* if they are different. The main drawback of this metric is that it is unable to differentiate between different degrees of similarity and considers that all attribute values are of equal distance from each other. For example, a *director* can be considered more similar to a *manager* than to a *senior engineer*, but this differentiation cannot be translated by the overlap matrix. Thus, real-valued distance metric is often more desirable. A frequently used real-valued metric is *value difference metric (vdm)* [202] which is defined as follows:

$$d(v_x, v_y) = w(v_x) \sum_{c \in C} (P(c|v_x) - P(c|v_y))^2 \tag{8.9}$$

Where, C is the set of all class labels, $P(c|v)$ is the conditional probability of class c given v, and $w(v_x) = \sqrt{\sum_{c \in C} P(c|v_x)^2}$ is a weighing factor attaching higher weights to an attribute value that discriminates a class better. However, *vdm* has three major limitations: first, due to the asymmetric nature of the weighing factor (w) it is not a metric, second, it implicitly assumes attribute independence and third, the class conditional probabilities need to be determined separately from the training data which might not generate satisfactory results especially when the training data set is limited. Decision tree classifiers [173] are another approach to handle nominal data but their main limitations are their inability to handle attribute correlations efficiently and poor performance for continuous attributes. An alternative approach is to use *Adaptive Dissimilarity Matrix* [55] by learning the dissimilarity between the nominal attribute values. Let us consider a training set $\lambda = \{(x_1, y_1), ..., (x_n, y_n)\}$, with input $x_i = (x_{i1}, ..., x_{im})$ having m attributes and $y_i \in -1, +1$ be the class label. If an attribute X can take values in $V_X = \{v_{x1}, v_{x2}, ..., v_{xn_x}\}$, the dissimilarity measure (d) between each of these values is a real-valued function on VXV such that:

$$0 = d(v_x, v_x) \leq d(v_x, v_y) = d(v_y, v_x) \leq \infty, \forall v_x, v_y \in V \tag{8.10}$$

The (dis)similarity (d) for a pair x_i, x_j is defined as:

$$d(x_i, x_j) = \sqrt{\sum_{X=1}^{m} d_X^2(x_i, x_j)} \tag{8.11}$$

Where,

$$d_X^2(x_i, x_j) = M_X(x_{iX}, x_{jX}) = F_X^2(x_{iX}, x_{jX}) \tag{8.12}$$

Where, M_X is the non-negative matrix representing the (dis)similarity between two values in V_X, $M_X = F_X \odot F_X$ and F_X is learned based on the training set λ by minimizing the classifier error.

Thus for a social network graph having n nodes and to be represented with m representative nodes (selected as explained in Section 8.2.1), two nXm similarity matrices are generated. The first matrix, called the *structural similarity matrix*, is developed by applying the coupled node-edge scoring technique and the second matrix, called the *semantic similarity matrix*, is developed by applying appropriate distance functions on the semantic vectors (either numerical or nominal) of a pair of nodes. Thus, each cell (aXb) of each matrix has the structural or semantic similarity between the nodes a and b respectively. The two matrices are then combined by following a weighted matrix cell-by-cell addition as represented in equation 8.13 to form the *combined similarity matrix*. Weights (W) are attached by the users depending on the data available and the characteristics of the data to be analyzed and represented. The weight varies from $0.1 - 1$. Lower the weight, lesser is the contribution of the particular similarity matrix towards the overall similarity. Here, since the main motivation is to demonstrate the performance and generic characteristics of the proposed graph preview technique without depending on pre-acquired knowledge of the particular dataset, both the matrices are given equal

importance and the weights are set to 1.

$$\begin{bmatrix} c_{0,0} & c_{0,1} & . & . & c_{0,m} \\ c_{1,0} & c_{1,1} & . & . & c_{1,m} \\ . & . & . & . & . \\ . & . & . & . & . \\ c_{n,0} & c_{n,1} & . & . & c_{n,m} \end{bmatrix} = W_a \begin{bmatrix} a_{0,0} & a_{0,1} & . & . & a_{0,m} \\ a_{1,0} & a_{1,1} & . & . & a_{1,m} \\ . & . & . & . & . \\ . & . & . & . & . \\ a_{n,0} & a_{n,1} & . & . & a_{n,m} \end{bmatrix} + W_b \begin{bmatrix} b_{0,0} & b_{0,1} & . & . & b_{0,m} \\ b_{1,0} & b_{1,1} & . & . & b_{1,m} \\ . & . & . & . & . \\ . & . & . & . & . \\ b_{n,0} & b_{n,1} & . & . & b_{n,m} \end{bmatrix} \tag{8.13}$$

Where, the left-hand-side of the equation represents the combined similarity matrix, the right-hand-side represents the structural and semantic similarity matrices respectively, W_a is the weight attached to the structural similarity matrix and W_b is the weight attached to the semantic similarity matrix.

8.2.3 Node Assignment

Once the similarity between the m representative nodes with the n original nodes is determined, each of these m nodes needs to be assigned to an original node in such a way so as to *maximize* the total similarity score. An extension [30] of Munkres Algorithm [159] (a combinatorial optimization algorithm) is used to solve the assignment problem. The assignment problem derives its name from the practice of optimally assigning n workers to n jobs (each worker to one job only) where there is a fixed cost associated to each worker performing a particular job. The optimum assignment should result in a minimum total cost while assigning all the n workers to one of the n jobs. The original Munkres Algorithm is capable of handling only square matrices. The assignment problem for square matrices can be formally stated as: given a nXn square matrix A with each element represented as a_{ij}, one needs to find a permutation p (p_i; $i = \{1, 2, ..., n\}$) that *minimizes* $\sum_{i=1}^{n} a_{i_{p_i}}$. However, the assignment problem can be extended for rectangular matrices and is defined in that case as: for a nXm matrix, a set of k independent elements(where $k = \min(n, m)$) should be determined for which the

sum of these elements is minimum. Though, for cost-related assignment scenarios, the optimum outcome is a total *minimum cost*, for assignment scenarios based on similarity measure, the optimum outcome is a total *maximum similarity*. So, an initial step is added to the original algorithm, where values of each cell in each row of the *combined similarity matrix* is subtracted from the maximum value among all the entries. The version of the Munkres Algorithm, as used in this chapter, is presented in Table 8.2. The nXm *combined_similarity_matrix* is created where n is the total number of nodes in the original social network graph and m is the number of selected nodes to be utilized as representatives. If necessary, the matrix should be rotated to ensure that $m \leq n$. Since, for our purpose, m is always much less than n, the rotation is not necessary. As the *combined similarity matrix* is a rectangular matrix, the k (as discussed above for extended munkres algorithm) need to be computed. Next, the minimum value from each row is subtracted from each corresponding row value. The same is then repeated for column values as well where the minimum value from each column is subtracted from each column entry. Each element of the resulting matrix is then searched for a *'zero'* value. If there is no other *'zero'* element marked *'starred'* in that particular row or column, the corresponding entry is marked *'starred'*. Next, each column containing at least one *'starred zero'* is marked as *'covered'*. If k columns has already being marked *'covered'*, the *'starred zero'*s describe a complete set of unique assignments and the process is complete. However, if k columns are not *'covered'* so far, a *'noncovered zero'* is found and marked as *'prime'*. If the row containing this marked *'zero'* do not have a *'starred zero'*, alternating *'zero'*s are marked *'starred'* or *'primed'* respectively. On the other hand, if the row containing the aforementioned *'primed zero'* has a *'starred zero'*, the particular row containing the *'primed zero'* is marked *'covered'* and the *'covered'* marking is removed from the particular row containing the *'starred zero'*. The smallest *'uncovered'* value is then sequentially added to each element of every *'covered'* row and subtracted from each element of every *'uncovered'* column. The process is repeated for each *'noncovered*

zero' until k columns are covered. The time complexity of the implemented version of the algorithm is $O(n^3)$. However, the complexity can be decreased to $O(n^2)$ by adopting the dynamic version of the algorithm as proposed in [209]. After the completion of the assignment process, each representative node has been assigned to an unique original node and these m assigned original nodes are used for the remaining steps of the the social network preview generation technique.

8.2.4 Representative Graph Generation

In order to generate the final representative preview of a social network graph, the representative nodes, assigned to the original nodes, need to be connected via edges to form a graph structure. The technique by which edges are added between a node pair determines the structure of the resulting graph. Since preserving the structure of the original graph, as closely as possible, in the representatives is the primary goal of the proposed approach, the edge-connectivity criteria is very crucial. Table 8.3 presents the algorithm which generates the final representative graph to preview a large social network graph structure. The input is an array A containing the indices of the nodes of the original graph that the representative nodes are assigned to. The edge connectivity (whether to connect a node pair directly by an edge) is decided based on the shortest path length between the node pair being considered. The process starts by initializing the representative graph (G_{disp}) with m nodes and an empty set of edges. The m nodes are selected from the indices of the original graph stored in the A matrix. The shortest path between each node pair is computed from the original graph. Though this is a computationally expensive task, but it can be performed offline and only once for each graph. For each node i in the final display graph, the maximum length of the shortest path between i and all other nodes in the original graph is determined. For each node-pair (i, j) in A,

the *allowed_path_length_ratio* is determined. It is defined as:

$$allowed\_path\_length\_ratio[i,j] = \frac{shortest\_path[i,i]}{Max\_SP[i]} \tag{8.14}$$

Where, shortest_path[i,j] is the length of the shortest path between the nodes i and j and Max_SP[i] is the maximum length among the shortest paths between node i with all other nodes of the original graph. For each pair of nodes in A, if the *allowed_path_length_ratio* is greater than or equal to a *cut off value*, the node-pair (i,j) is connected via an edge and the edge is added to the edge set of the representative graph G_{disp}. The denominator of *allowed_path_length_ratio* is set as $Max\_SP[i]$ because it provides information about the overall edge-connectivity characteristics of the particular node in the original graph. A higher value for $Max\_SP[i]$ suggests that this node has a high *reachability* in the Graph and connects to distant neighbors. A higher value for $Max\_SP[i]$ lowers the *allowed_path_length_ratio*. Thus, for the same *cut off value*, a node with a higher $Max\_SP[i]$ has a greater chance of being connected via an edge to another node than a node with a lower $Max\_SP[i]$ (suggesting its lower *reachability* in the Graph). The *cut off value* should range between 0 and 1. Lower values make the edge connectivity criteria strict whereas higher values loosen this requirement condition. If this value is kept too low, very few edges will be connected and would lead to a disconnected final graph. On the other hand if it is kept too high, almost all the nodes will be connected to each other. There cannot be a fixed threshold value, applicable to all graphs. The *cut off value* needs to be determined dynamically for individual graphs after determining its overall connectivity characteristics. One approach is to find the *allowed_path_length_ratio* for every node in the original graph during the pre-analysis phase. The average of the *allowed_path_length_ratio* is then utilized as the *cut off value*. Other approaches include running the algorithm through several iterations with different *cut off* values in different ranges and picking up the appropriate one. For example, for a largely disconnected graph, different low *cut off* values, such as between $0.1 - 0.2$ are tried. Whereas, for a

densely connected graph, higher values in the range $0.5 - 0.75$ are utilized in different iterations.

Table 8.3: Representative graph generation

Preview_Graph_Generation(A**)** {
//A is the array containing ids of the original nodes to which the representative
//nodes are assigned.
Initialize output graph G_{disp} with m nodes and empty vertex set;
for each node i in original graph {
for each node j in original graph {
Set shortest_path[i,j] = shortest path length between
node i and j in the original graph;
}}
for each node k in A {
Set Max_SP[k] = Max(shortest_path [k]);
//Max_SP stores the maximum of the shortest path length
//between node k and all other nodes in graph.
}
for each pair of nodes (i and j) in A {
Set $allowed\_path\_length\_ratio[i,j] = \frac{shortest\_path[i,j]}{Max\_SP[i]}$;
if ($allowed\_path\_length\_ratio[i,j] \leq cut\_off\_ratio$) {
Add node i and j by $edge_{ij}$;
Add $edge_{ij}$ to G_{disp};
}}
output: G_{disp}
}

8.3 Evaluation Score

Evaluating the generated representative preview graphs, in terms of similarity to the original graphs, is largely based on perspective of the individual and can be imprecise in nature. However, since preserving the overall structure of the original graph in the representative graph is the main motivation of this research, an overall structural comparison of the graphs is utilized for the evaluation purpose. The structure of a social network graph can be determined by measuring the degree of similarity between connected nodes (dyads). [161] proposes a measure to quantify this similarity using *assortative mixing* by

Table 8.4: Data set characteristics

DataSet	**# of Nodes**	**# of Edges**	**Average E_C**
adjnoun	112	850	0.565733
celegansneural	297	2359	0.720667
*enrongraph*_500	215	400	0.730867
*enrongraph*_5000	516	5000	0.7972
*enrongraph*_10000	628	10000	0.7377
karate	34	156	0.433467
lesmis	77	508	0.5339
netscience	1461	5484	0.945133

determining the correlation between centrality measures of every node-pair. An alternative approach is using Euclidean distances to measure the similarity between dyads, which has been found to be more precise in determining the overall structural characteristics of a social-network graph [122]. As *centrality* has been found to capture the overall characteristics of a social network structure based on individual actor behaviors, the Euclidean distance computation utilizing centrality of *all* the dyads of a social network graph provides an overall structural measurement metric. The structure of a social network graph is determined by the ties or the relations among the nodes. The two extreme structures resulting from such ties are star configuration (where every node is connected to a single node) and circle configuration (where every node is connected to every other node). Thus E_C, defined in Equation 8.15, provides information about the structural characteristics of the graph (if it is having a star configuration or a circle configuration or a configuration in-between). The Euclidean distance based on equi-centrality is defined as:

$$E_C = 1 - \frac{\sum_{k=1}^{M} \sqrt{(c_{ik} - c_{jk})^2}}{max(\sum_{k=1}^{M} \sqrt{(c_{ik} - c_{jk})^2})} \tag{8.15}$$

Where, c_{ik} and c_{jk} are the normalized centrality values of actors i and j connected by the edge k, $i, j = 1, 2, ..., N$, $k = 1, 2, ..., M$, N is the number of actors in the network and M is the number of dyads.

The denominator is the maximum possible value of the numerator (presented in Equation 8.16) and is used as a normalizing factor. When the graph has a *star* configuration,

the numerator becomes maximum since every actor is connected to a single central actor, the centrality of the node at one end of the edge (c_{jk}) is always zero (illustrated in Figure 8.1(a)). Hence, the numerator becomes:

$$numerator = \sum_{k=1}^{M} c_{ik} \tag{8.16}$$

E_C has a minimum value of 0 when the graph has a star configuration and the numerator and the denominator of the second term of Equation 8.15 becomes equal. It attains a maximum value of 1 when the graph has a circle configuration (Figure 8.1(b)) where all the actors have exactly the same centrality value and the numerator of the second term of Equation 8.15 becomes 0. Thus, when E_C of a particular graph is high, it indicates that the actors in it tends to associate with others who have similar central position in the network. On the other hand, when E_C of a particular graph is low, it suggests that actors having more central locations interact with actors with less central positions in the network. As there are different types of centrality calculations, such as degree centrality, betweenness centrality and closeness centrality, there can be different E_C measures depending upon the type of centrality used. Thus, from the score of E_C, one can derive information about the overall structure of the social network graph structure, whether it is tending towards a *star* or a *circle* configuration.

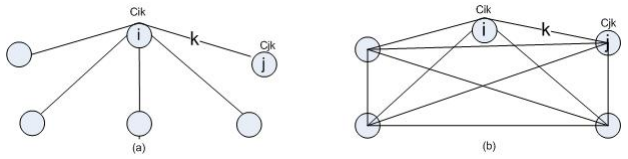


Figure 8.1: A graph with (a) Star configuration (b) Circle configuration

To evaluate how close the generated representative graph is to the original social network, their structural metrics are compared. For a pair of original graph and its generated preview graph, the E_C values are calculated and the percentage error ($Error_{O,R}$)

is determined as:

$$Error_{O,R} = |\frac{(E_{C_O} - E_{C_R})}{E_{C_O}}|X100 \tag{8.17}$$

Where, E_{C_O} and E_{C_R} are the overall Centrality-based Euclidean Scores for the original and the representative graphs respectively. Lower the $Error_{O,R}$ value, closer is the structure of the representative graph to the original graph.

Table 8.5: Generated pre-determined representative graph characteristics

DataSet	**# of Nodes**	**# of Edges**	**Average E_C**
adjnoun	59	102	0.722567
celegansneural	153	374	0.822433
enrongraph_500	122	174	0.596967
enrongraph_5000	294	1118	0.725767
enrongraph_10000	346	1901	0.7765
karate	17	19	0.596067
lesmis	35	55	0.8504
netscience	275	2637	0.8972

8.4 Data Set

In this chapter, eight social network data-sets with different characteristics and sizes have been used for demonstrating the effect of the proposed algorithm. The dataset *adjnoun* [162] is the adjacency network of common adjectives and nouns in the novel David Copperfield. The dataset *celegansneural* [217] is a network representing the neural network of C. Elegans. The datasets *enrongraph_500*, *enrongraph_5000* and *enrongraph_10000* are social networks developed from the e-mail communications among employees in an organization [131]. In order to demonstrate the effect of size, the same dataset with different number of data elements is used (number of data elements in *enrongraph_500* $<$ *enrongraph_5000* $<$ *enrongraph_10000*). The dataset *karate* [230] is a social network of friendships between 34 members of a karate club at a US university in the 1970s. The dataset *lesmis* [132] is the co-appearance network of characters in the novel Les Miserable. The dataset *netscience* [162] is the co-authorship network of scientists working

on network theory and experiments. The input data is in the form of GML (Graph Modeling Language) [107]. Each dataset has been pre-analyzed to derive an idea about the characteristics of these social network data. These measures can be compared with that of the generated preview graphs for evaluation. Since, centrality is an important property of a social network structure, three different centrality values, namely degree centrality, betweenness centrality and closeness centrality are determined and presented in Figure 8.2.

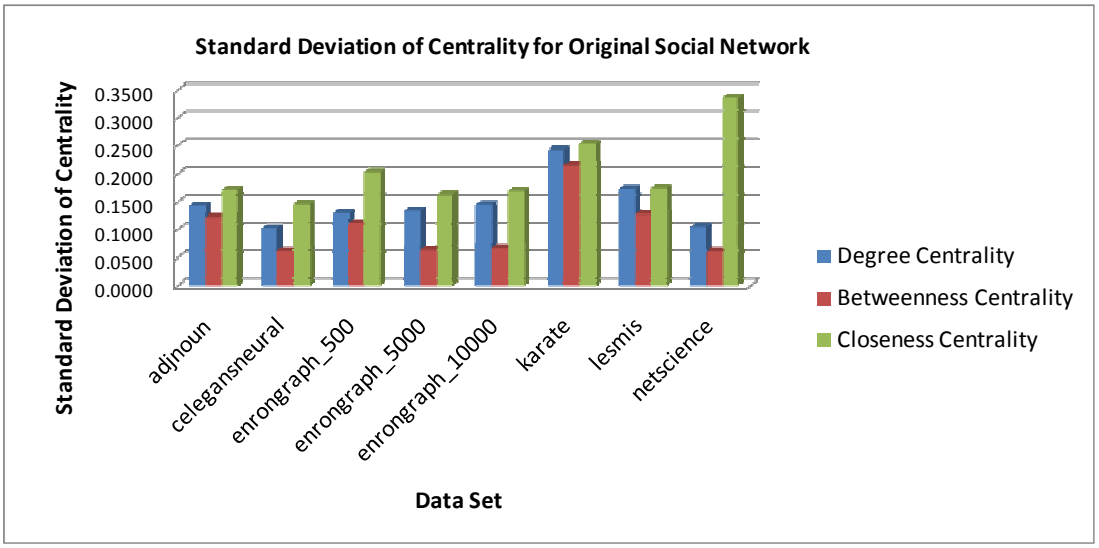


Figure 8.2: Characteristics of the original social network graphs

Additionally, different statistics of the individual dataset is computed, such as number of nodes in the social network graph (actors), number of edges (relations between the actors) and the average E_C (as explained in Section 8.3). Also, the standard deviation of the centrality values is calculated to obtain the centrality distribution. Table 8.4 presents the characteristics of the input data. As E_C provides information about the general structural characteristics of the social network graph (0 for star configuration and 1 for circle configuration), from the value of average E_C, it can be deduced that *adjnoun* and *lesmis* has neither a complete star configuration nor a complete circle configuration. Thus, in it the actors uniformly associate with one another and do not have a preference of either as-

sociating with others similar in position (power) to itself or with those who are different. On the other hand, *celegansneural*, *enrongraph_500*, *enrongraph_5000*, *enrongraph_10000* and *netscience*, all have high E_C thereby suggesting that in these datasets, actors tend to network with other actors who are similar to it (or having similar positions/power in the data set). Thus, intermingling among different categories is less. Conversely for dataset *karate*, a low E_C suggest that intermingling is large (that is actors having more central positions associate with ones having less central positions).

8.5 Empirical Analysis

In this section, a detailed experimental analysis of the proposed social network preview technique is presented with different types of social network data varying both in source, size as well as in network structure. For each data-set, four representative social network graphs are generated, namely *Pre-determined Representative Graph* (Representative I), *Random Representative Graph* (Representative II), *Clustered Graph* (Clustered I) and *Random Clustered Graph* (Clustered II). Representative I and Representative II are generated by utilizing the proposed preview generation algorithm utilizing graph similarity. The only difference between them is in the initial sampling process. While Representative I uses a centrality-based stratified sampling technique, Representative II adopts a random sampling without replacement method. For both Representative I and II, the structural similarity matrix is computed using an iterative edge-node coupled similarity computation method. For all the datasets used, the number of iterations (k) necessary for the solutions to converge range from $3 - 5$. Since the most popular existing method of representing social network graphs with fewer nodes/edges is via *Clustered Graphs*, the approach proposed in this chapter is compared with them. There can be several techniques of generating clustered graphs as discussed in Section 8.1. The two important steps of generating clustered graphs are (i) constructing the clusters and (ii) connecting

the clusters forming graphs. For unbiased comparison purposes, we formed the clusters based on centrality criteria as Representative I and II use centrality for the initial sampling. The average of three different centrality measures (namely degree, betweenness and closeness) are calculated for each node of a social network graph and are sorted. 20% of the nodes are used as initial cluster seeds or cluster centers. These 20% of the nodes should have preferabely unique centrality scores. Each node (the cluster center) is checked for adjacent nodes, not already in another cluster. If any such node is found, it is inserted into the corresponding cluster. Thus, clusters are formed based upon interactions among the neighboring nodes. This approach is adopted to introduce the notion of locality in the clustered-graphs and to make them comparable to the preview representatives, generated with the the proposed technique, where neighboring nodes influence the similarity scores between the nodes. Once the clusters are formed, a graph structure is obtained by connecting the clusters with edges (relations). Two clusters are connected via an edge if there is at least an edge between two nodes belonging to each cluster. For example, let us assume two clusters A and B with i and j nodes respectively. Let, $\{a_1, a_2, ..., a_i\}$ and $\{b_1, b_2, ..., b_j\}$ be the node-sets belonging to clusters A and B respectively. Clusters A and B are connected via an edge in the clustered graph $G_{clustered}$, iff $\forall$ A,B; $\exists$ $edge_{rs}$ between some a_r,b_s. Clustered II is generated following the same algorithm with the only difference in the way the initial cluster seeds are selected. For Clustered II, 20% of nodes from the original social network graph are selected as cluster seeds randomly. The main reason for selecting 20% of the original nodes as representative cluster seeds is due to the fact that for the given datasets, most did not have more than 20% nodes with unique centrality values. The reason for preferring unique cluster seeds is to make the clusters as disjoint as possible (which is the baseline for good cluster generation).

The statistics of the representative graphs, generated with different techniques, is presented in Tables 8.5, 8.6, 8.7 and 8.8. Table 8.5 presents the characteristics of the representative preview graph generated utilizing the proposed method with pre-meditated

sampling. Comparing the Average E_C score between Representative I and the Original Graph, it can be concluded that the generated representative preserves the original structural characteristics *better* when the data set is large. This can be seen from the E_C scores for the dataset *enrongraph_10000*, *enrongraph_5000* and *netscience*, having the three highest number of nodes among the given dataset in Table 8.4 (628, 516 and 1461 respectively). This is mainly because larger datasets have more sample points and more well-identified distributions which help in capturing the characteristics better.

Table 8.6: Generated random representative graph characteristics

DataSet	**# of Nodes**	**# of Edges**	**Average E_C**
adjnoun	54	82	0.742867
celegansneural	112	327	0.873733
*enrongraph*_500	114	173	0.552133
*enrongraph*_5000	211	513	0.700767
*enrongraph*_10000	307	1435	0.760067
karate	5	3	0.532867
lesmis	29	37	0.830433
netscience	134	620	0.887633

Table 8.6 presents the characteristics of the representative preview graphs generated utilizing the proposed method with random sampling. They behave in a similar way to Representative I though with a slightly higher E_C error. Though for a few instances, Representative II performs better than Representative I, this behavior can be contributed to the randomness of the selection process and is not a trend. If the random samples are able to capture the distribution of the original dataset closely, the final representative graph has a close structural resemblance to the original graph. Whereas, if the random samples fail to capture the distribution, the final results deteriorate. However, it should be noticed that the effect of the initial sample selection is not that pronounced in the proposed approach. The similarity computation, the node assignment and the graph generation steps together are able to produce impressive final results independent of the initial selection techniques. This further demonstrates the domain-independence of the proposed technique as the only place where any domain knowledge might be necessary is

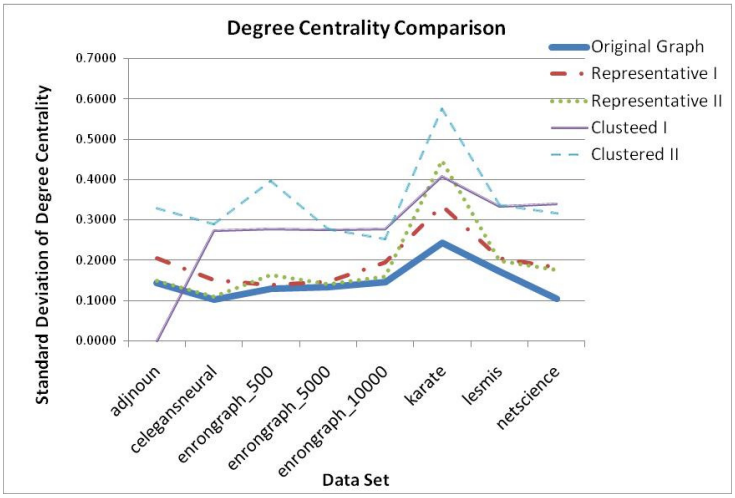
the initial node selection step. Thus, since the performance is consistent even for random selections (where no domain-information is used), the claim of the generic nature of the proposed algorithm is established.

Table 8.7: Generated clustered graph characteristics

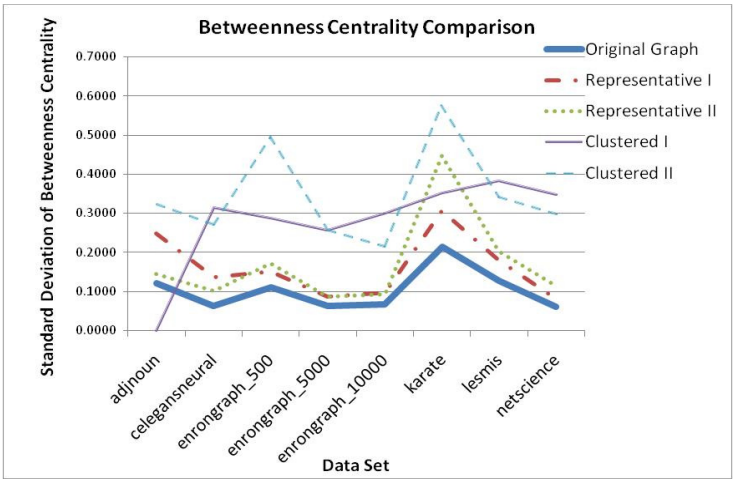
DataSet	**# of Nodes**	**# of Edges**	**Average E_C**
adjnoun	9	36	1
celegansneural	26	264	0.082067
*enrongraph*_500	14	52	0.251167
*enrongraph*_5000	52	821	0.121433
*enrongraph*_10000	61	1336	0.091967
karate	6	11	0.041467
lesmis	9	29	0.076433
netscience	18	93	0.118167

Table 8.7 presents the characteristics of the representative preview graph generated utilizing a clustered graph approach as discussed before. The E_C error is higher than the proposed approach for all the datasets. This is due to the drawback of the clustered graphs where the overall structural similarity is not preserved naturally in the representative preview graphs. Table 8.8 presents the characteristics of the representative preview graphs generated utilizing clustered graph technique with random initial cluster seed selection and performs very poorly in terms of E_C error. This is due to two reasons: first, just as clustered graphs (Clustered I), natural structure conservation is absent and second, random initial cluster seeds might fail to pick up the most suitable ones (those that are capable of forming disjoint clusters) which further deteriorate the performance.

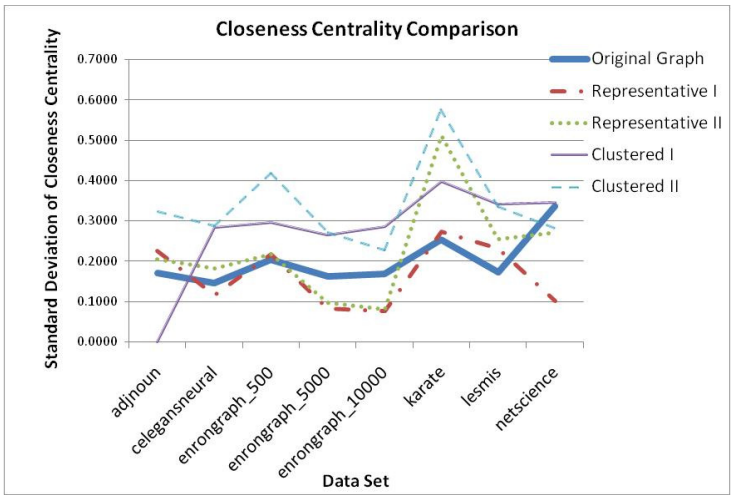
Figure 8.3 compares the data characteristics in terms of centrality values of the representative preview graphs generated using different discussed techniques with the original graphs. Figure 8.3(a) compares the degree centrality values, Figure 8.3(b) compares the betweenness centrality values and Figure 8.3(c) compares the closeness centrality values. It can be seen that both Representative I and II have values closer to the original graphs than the clustered graphs. Similar deductions can be made from Figure 8.4, where E_C of the different preview graphs generated using different techniques is compared with the



(a)



(b)



(c)

Figure 8.3: Comparison between original graph and representative graphs for (a) Degree centrality, (b) Betweenness centrality, (c) Closeness centrality

Table 8.8: Generated random clustered graph characteristics

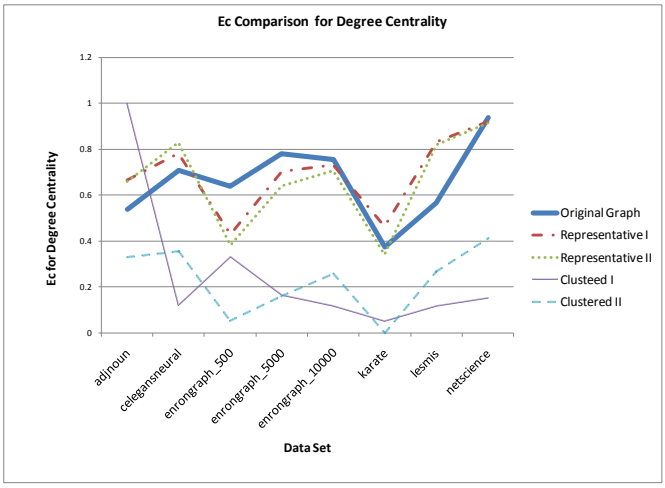
DataSet	**# of Nodes**	**# of Edges**	**Average E_C**
adjnoun	9	22	0.1825
celegansneural	30	207	0.223467
*enrongraph*_500	8	21	0.028033
*enrongraph*_5000	39	334	0.092567
*enrongraph*_10000	57	551	0.175767
karate	4	5	0
lesmis	9	18	0.2359
netscience	17	60	0.335233

original graphs. It can be seen here as well that Representative I and Representative II have closer E_C values to the original graphs than the clustered graphs.

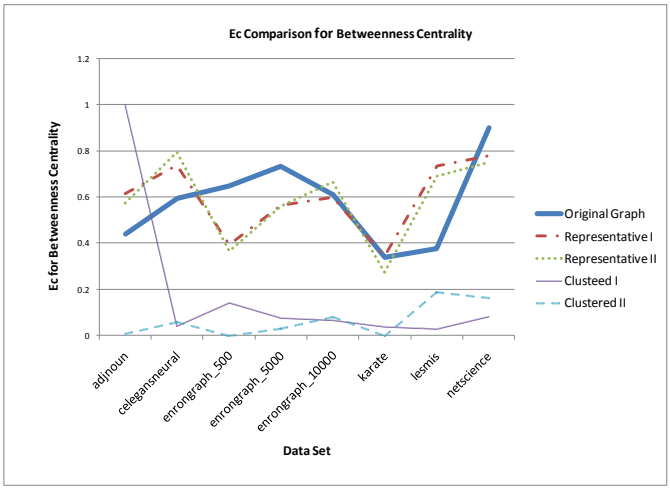
Figure 8.5 compares the performance of the proposed technique (Representative I and Representative II) with clustered graphs (Clustered I and Clustered II) in terms of the E_C error as presented in Equation 8.17. The proposed algorithm produces less error than the clustered graphs. The average percentage error for Representative I and Representative II is 24% and 25% respectively whereas for Clustered I and II, the average percentage error is as high as 84%. The representative preview graphs generated with the proposed algorithm along with the original graph structures and the clustered graphs are presented in Figures A.1 - A.28 for most of the datasets for visual comparison. Since Random Clustered Graphs have similar characteristics as Clustered Graphs, their representations are not presented in these figures. As pointed out before, evaluation of the representative graphs is imprecise and though the visual representations provide some preliminary ideas about the overall similarity between different graph structures, precise scores such as E_C is nevertheless very important for more precise conclusions.

8.6 Multimedia Data Network

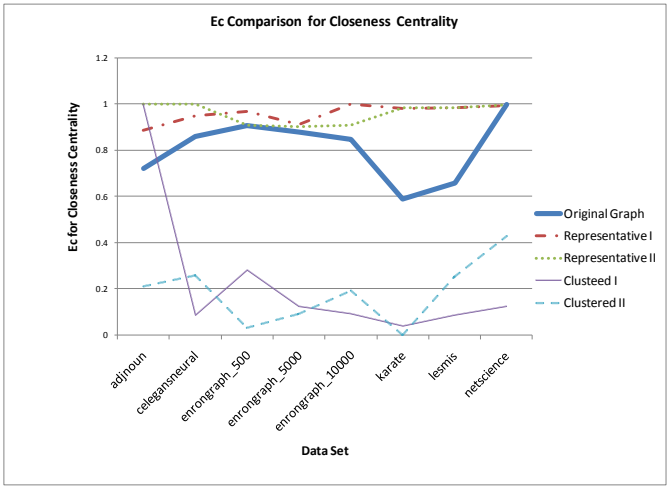
When a multimedia data corpus is shared by multiple users in a collaborative environment, the pattern of the user behavior affects the relationships among the data. This



(a)

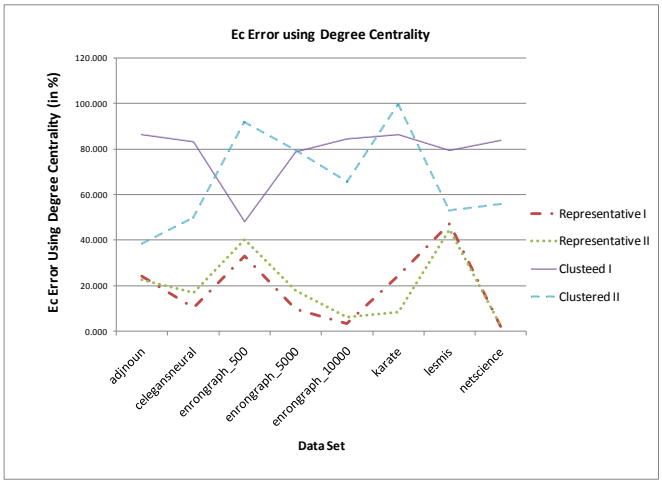


(b)

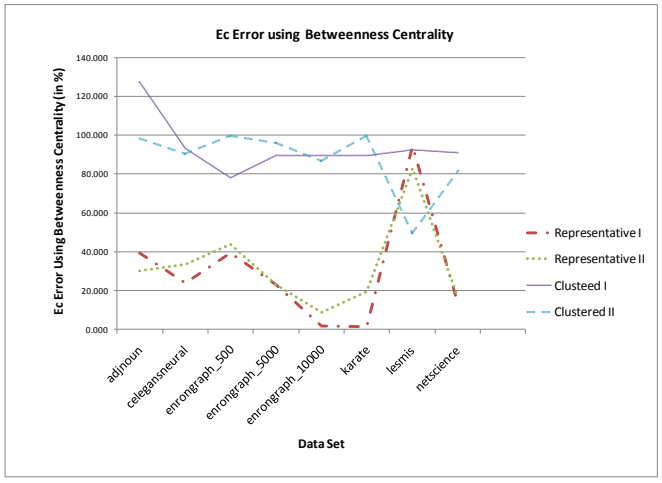


(c)

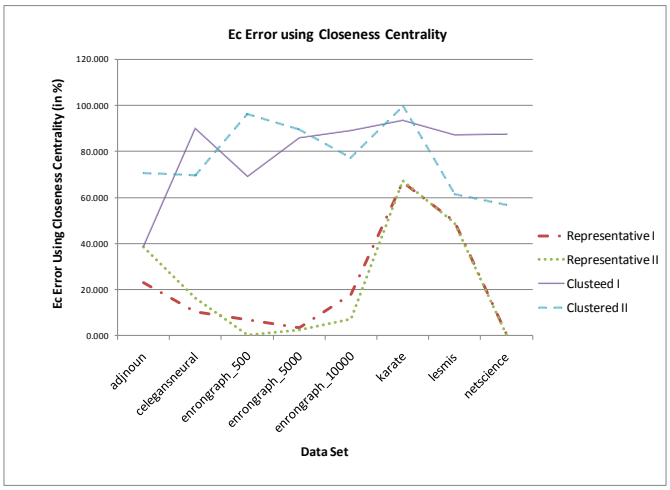
Figure 8.4: Comparison of E_C between original graph and representative graphs using (a) Degree centrality, (b) Betweenness centrality, (c) Closeness centrality



(a)



(b)



(c)

Figure 8.5: Comparison of E_C error between original graph and representative graphs using (a) Degree centrality, (b) Betweenness centrality, (c) Closeness centrality

relationship between the data might change with different user groups even if the data corpus remains the same. Thus for each user group, the relationships between multimedia data objects can be modeled as a graph, replicating the representation of a social network. Such a representation of the multimedia data corpus, modeled based on the user behavior, can be termed as a *Data Network*. Each multimedia data object is represented as a node in the graph and can be considered as an actor (to keep the resemblance with a social network). The relationship between the data objects are represented as edges. These relationships are determined from the users behavior, that is how the users have used these multimedia data objects together. The result is a model of the entire multimedia data corpus depicting their inter-relationships for a particular user group. It should be mentioned here that just like a social network, the type of relationship between the multimedia data varies and depends on the application from which the user behavior has been collected.

One of the most important and common relationships between a pair of multimedia data is similarity. Since the concept of similarity between multimedia data is an imprecise but nevertheless a very fundamental factor in guiding the management and organization policies, here, the data network is modeled based on the similarity relationships between the multimedia data. Furthermore, the main motivation of this dissertation is to develop an index structure that can accommodate the different characteristics and behavior of multimedia data. Thus, the primary interest was to ascertain that the data network developed can improve the proposed index structure and make it more flexible and efficient. Since, the index structure has to primarily handle content-based similarity queries, developing the data network modeled on similarity and utilizing it to better design the index structure is the desired outcome. Keeping all the above requirements in mind, the data network for the multimedia data corpus is built based on the similarity which is measured as the *number of times users belonging to a particular group have thought two images to be similar to each other*. To emphasize on the *degree of similarity*, a weighted

graph representation is used where the varying weights are demonstrated with variable thickness of the edges. Figure 8.6 presents a sample of the data network generated for 100 images from the COREL data corpus. The similarity information is collected from a content-base image retrieval application where users are given the option to mark images similar to a submitted query. The data has been collected from several users belonging to the student community of Florida International University, for over five years. As seen in Figure 8.6, such graph is often disconnected. Images form clusters and similar images are clustered together at different regions of the Data Network. Also, it should be noted that the edges (relationships) have a varying thickness. More is the thickness, stronger is the users' perception of similarity between that pair of image. The entire COREL data corpus has been modeled in Figure 8.7. As described in this chapter, since the size of the generated Data Network for 10000 images is large and hence visually analyzing it becomes quite difficult. The proposed graph preview technique can be as well utilized to generate representatives of the Data Network with fewer number of nodes and edges.

8.6.1 Analyzing Multimedia Data Network

In order to utilize the information gathered from the generated Multimedia Data Network to design the database components of the multimedia database management system, the Multimedia Data Network needs to be analyzed. Since, the Multimedia Data Network is modeled based on Social Network structures, characteristics related to a social network are used. It has been discussed in details in the beginning of this chapter how *centrality* plays an important role in capturing the behavior of an individual element of a social network with respect to the overall network structure and characteristics. The three types of centrality, namely Degree Centrality, Closeness Centrality and Betweenness Centrality is computed for all the elements of the entire Data Network. For brevity, 100 top centrality values along with the corresponding image ids are presented in Appendix A in Tables A1, A2 and A3 respectively. Table A1 presents the 100 highest degree centrality values,

Table A2 presents the 100 highest closeness centrality values and Table A3 presents the 100 highest betweenness centrality for the COREL database.

Improving GeM-Tree Utilizing Multimedia Data Network

It should be recalled here that though the retrieval strategies supported by the proposed index structure consider the high-level semantic relationships among the multimedia data objects, the generation of the index structure along with the corresponding operations such as *insert*, *delete*, and *update* are based on only the low-level features of the data. As presented in Lemma 4.3.1, the high-level relationship could not be included while constructing the GeM-Tree without violating the properties of the underlying metric space. Thus, the feature-level similarity computation (also called the distance) between a pair of multimedia data objects is the only factor deciding the distribution of the data objects in the metric space. This definitely has its disadvantages especially when the dataset is such that the low-level features are unable to translate the high-level semantic similarity.

Under those circumstances, the generated Data Network and the subsequent analysis can help to introduce the concept of high level similarity into the index structure. Some typical decisions related to the index structure that might be derived from the semantic network and the analysis of the various properties are as follows:

1. Insertion Policies: As recalled from Chapter 4, the insertion policies applied to a node of an index structure greatly affects the subsequent performances of the index structure. It is essential to pick up the appropriate node where an incoming data is to be inserted as it influences the similarity search results. If an appropriate node is not chosen during insertion, it may lead to false dismissals of potential query results, thus reducing the accuracy or might result in unnecessary traversal of a particular node, thus increasing the computation cost. The insertion policy used in the proposed index structure follows the rule of picking up a node as a candidate

which has no or minimum increase in the covering radius as expressed in Equation 8.18 and 8.19 respectively and discussed in details in Section 4.2.4.

$$d(O_r, O_n) \leq r \tag{8.18}$$

$\forall$ O_p, O_q, pick O_p if

$$d(O_p, O_n) - r_p \leq d(O_q, O_n) - r_q \tag{8.19}$$

It can be noted here, that the above policy is entirely determined based on the low-level feature distances. The d used in both the equations essentially measures the distance between the feature vectors of the candidate multimedia data object and the multimedia data object to be inserted. To the best of our knowledge, there is no existing insertion policy that considers the high-level semantic relationships between the multimedia data objects as well. The Centrality value can play a pivotal role in this case. More specifically the degree centrality should be utilized along with the low-level feature similarity to select the intermediate tree node where new data should be inserted. Hence, if a node with a higher degree centrality is chosen as a candidate node where a new data object is to be inserted, after it passes a certain threshold of feature-level similarity with the data object, the probability that these two data objects are semantically related becomes larger. Thus, if there are two candidate nodes with comparable feature level distances (similarity) with the data object to be inserted, its a prudent choice to pick up the one that has a greater degree centrality as the particular node is more important and influential in the semantic network and has been voted, by users, as similar to more data objects in the database than the other one. For example, during insertion of a particular image object, say it has almost equal low-level similarity with images #198 and #7226, each having a degree centrality value of 0.97 and 0.74 respectively, with the low-level similarity slightly higher with image #7226. But, since the degree centrality of image #198 is much higher that that of the other, image #198 can

be chosen to be the node where the incoming data is to be inserted. Elementary experimental results demonstrated the effectiveness of the above concept. Such approach can be also utilized when a tie is reached during the splitting of a full node during insertion. The approach introduces the concept of high-level semantic relationships between multimedia data objects in the tree structure itself without violating the metric properties.

2. Deletion Policies: At present, whenever a delete request is received from an user, it is executed, without any consideration of the effect it might have on the existing semantic relationships among the data objects. But, frequently such decision might lead to removal of important semantic links, useful in the retrieval techniques and thus affecting the performance of subsequent query results. The property of betweenness centrality can be utilized to determine the cascading effect that a multimedia data object might have on the rest of the network. If there are several strong semantically related pairs linked via a particular data object, it is certainly not a good idea to remove it from the database. Thus, while a deletion request is received, before acting upon it, it is efficient to analyze its importance in the semantic network. If it is a crucial data object, the user should be notified of the effects that its deletion will have and possibly avoid such action.

Another important usefulness of this data modeling is to understand the collaborative behavior of the multimedia data and utilize it to customize the index structure so as to be able to serve applications where multimedia data need to be managed in a collaborative environment. It is worth mentioning here that not only the index structure but other components of the database management framework such as the query processor, query optimizer, etc. might be designed differently with the behavioral information of the multimedia data gathered from such Data Network.

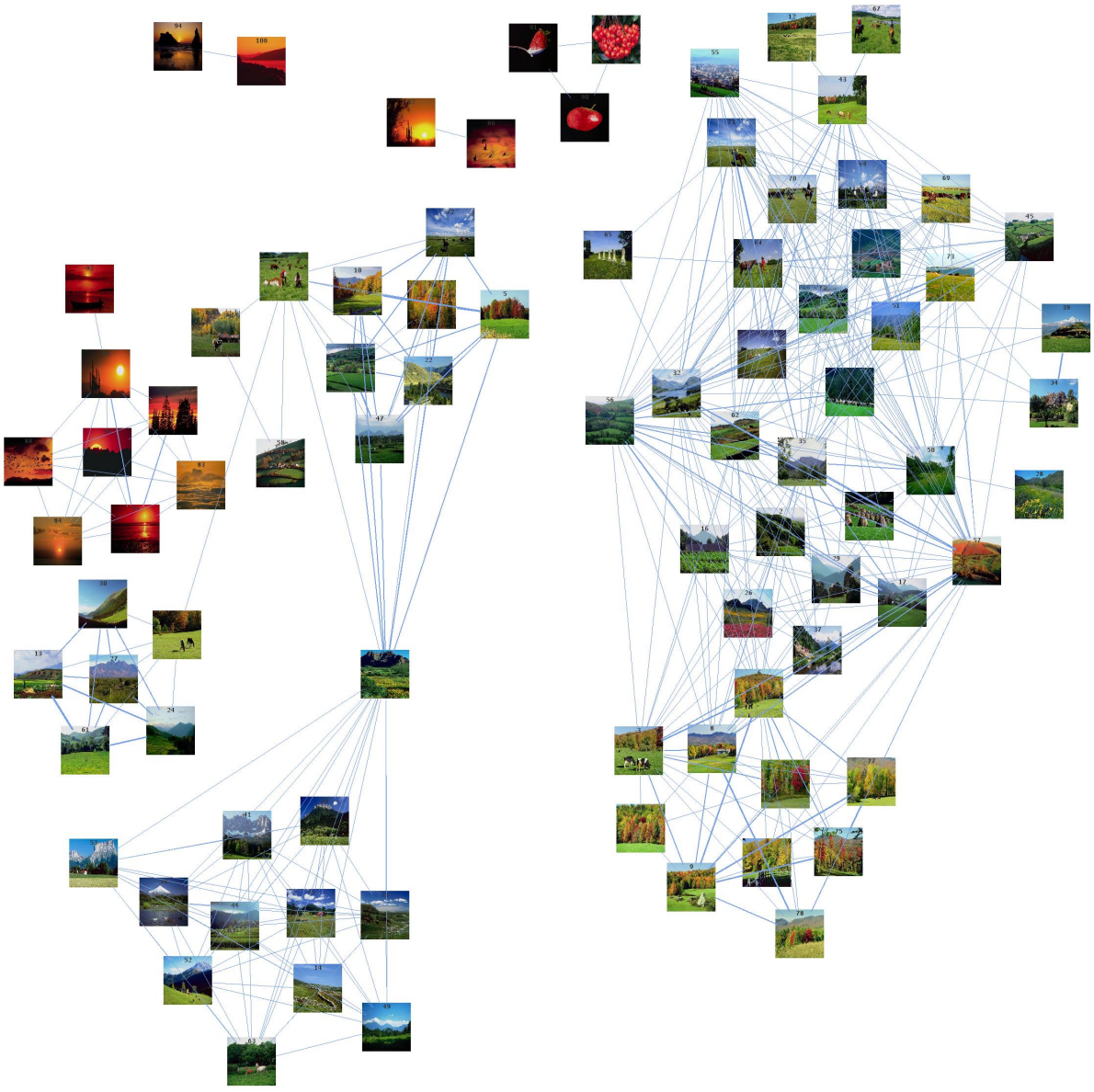


Figure 8.6: Sample data network for 100 images from COREL

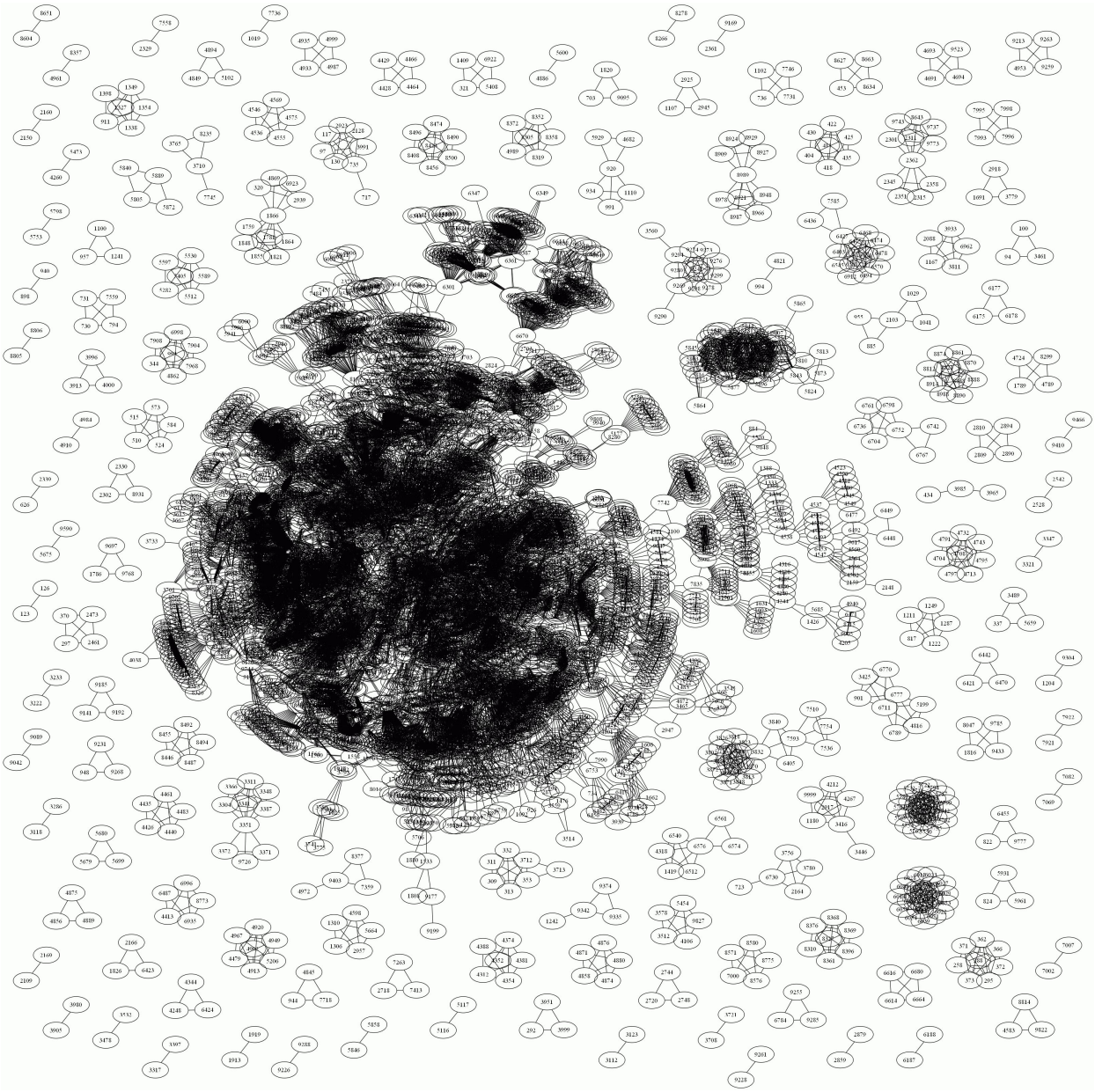


Figure 8.7: Complete data network for 10000 images from COREL

8.7 Conclusion

In this chapter a social network preview technique utilizing graph similarity is proposed which represents a large social network graph with fewer nodes and edges. Thus, visualization and analysis of such large social networks become convenient. The proposed approach largely preserves the original social network structure in the representative graphs. It uses an edge-node coupled graph similarity method to compute the structural similarity between the nodes of the original and the representative graphs and an euclidean distance based metric to compute the semantic similarity between them. A node assignment approach is adopted utilizing an extended Munkres Algorithm to assign each representative node to a node of the original graph so as to maximize the total similarity between the two graphs. A final preview graph is generated by appropriately connecting the representative nodes following a novel graph generation method based on shortest path lengths. The generated graph structures are evaluated and compared with the original graphs using a metric based on centrality scores which can provide important information about the overall structure of a graph. Extensive experiments are performed with different social network data belonging to different genres and having different characteristics and sizes. The proposed technique is compared with a popular graph preview approach, the clustered graph method. Experimental results demonstrate that the proposed algorithm is capable of generating representative social network previews with low percentage error and closely preserves the overall characteristics of the original social networks. The proposed technique also performed better than the clustered graph methods. Thus, it can be concluded that the proposed technique is a promising method and has potential for future improvements. The concept of social network representation is extended to model multimedia data relationships for a given data corpus to form Multimedia Data Networks. The generated Data Networks are analyzed to collect important information about individual data object with respect to the entire network. These properties can be utilized to introduce the concept of high-level semantic information into the

proposed index structure to improve the quality of query results without violating the properties of the underlying search space.

CHAPTER 9

A DISTRIBUTED MULTIMEDIA DATA MANAGEMENT OVER THE GRID

9.1 Introduction

Grid computing can be described as a form of distributed computing which combines the power of several computing nodes of varied computing resources to execute one or more tasks collaboratively in a seamless and transparent manner without any central control [80][81][82]. In the recent years, the popularity of Grid Computing has enabled experts from different scientific backgrounds to use its high computing power to execute computation intensive applications. Often these applications are data intensive like in protein folding, semiconductor manufacturing, and DNA sequence analysis. Such applications need a well defined data management within the distributed Grid environment.

There are basically two different approaches of designing a Data Grid: namely management of static data and supporting dynamic data sets. The first approach is also called Level 0 Data Grid [40]. It does not address data management issues as updates, transactions, integrations, etc., which are typical to data that changes with time. Basically, it addresses two fundamental issues: data access and meta data access. The data access provides managing, accessing and transferring data that are stored in the storage (typically as file systems). It essentially implements a storage system abstraction so that the applications need not be aware of the specific low-level policies utilized in the data management. The meta-data service provides a mechanism for presenting and using the information about the data (stored in the files). Different categories of meta data can be used: such as content information of the file, data creation environment, and application-specific information related to the data. Apart from the two basic functionalities, Level 0 Data Grid provides some added services such as authorization and authentication, resource allocation, and performance evaluation. Level 1 [203] data Grid

is for dynamic data sets and accommodates methods such as access, management, transaction and synchronization of data. To develop data Grids comparable in performance and robustness to the traditional data management techniques, features including indexing, querying, and transaction management should be provided effectively as well. These features should be incorporated seamlessly along with features which are typical to Grid environment such as data regionalization, data synchronization, and load balancing.

As it has been already discussed, multimedia data is more complicated than traditional text-based data both in representations as well as in access mechanisms involved during their retrievals. Thus, any data management framework for multimedia data should be able to accommodate the atypical characteristics of multimedia data: namely the multidimensional representation and the semantic information. Though multimedia data is more complex than traditional text-based data; they are a popular media of communication due to their expressiveness. Thus, their presence and requirements in today's popular applications cannot be avoided. Hence, to enhance the usability of Grid environment, the data Grid should be able to manage multimedia data effectively as well. But, since multimedia data is quite different from traditional text-based data, their management frameworks should also be different. For example, the index structure for multimedia data should be multidimensional as opposed to the popular single dimensional index structures of text-based data. Additionally, since their information needs are different, the retrieval methodologies that the database system, managing multimedia data, should support are different too. All these calls for a dedicated multimedia data management framework over the distributed environment of a Grid architecture.

The Internet can be considered as a distributed environment and can be simulated with a Grid architecture. Several popular applications such as social networks, collaborative tools, and search use multimedia data heavily. Thus a multimedia data management architecture for Grid environment can be considered as a prototype for investigating multimedia data management within the Web environment. One specific application which

can benefit immediately from such prototype is *multimedia search*. Currently, the multimedia search is based on *keywords* or *annotations*. Such search paradigm degrades the relevance of the search results manifold. Firstly, a single multimedia object (an image or a video) can have multiple high-level semantic meaning attached to it as the semantics vary with the perspective of the user who labels it. Thus, one keyword will be unable to capture the different aspects of different users. Secondly, the multimedia data is represented and stored as multidimensional feature vectors. Thus, for a keyword-based search, during retrieving them from the underlying storage, a relationship need to be established between the low-level features and the high-level semantics (keywords with which they are labelled). This relationship is often fuzzy and there exists a gap between them, called the semantic gap. This affects the relevance of the query results and degrades the quality of the search results. The best approach is to introduce a content-based search paradigm for multimedia data which will be distributed over the Internet. Thus, a successful layout of a multimedia data management and content-based retrieval system over the Grid will be a potential solution for solving the problem of managing multimedia data in the Web environment.

In this chapter, we lay down the framework for distributed multimedia data management over a Grid computing environment. It comprises of two types of components: firstly, components related to the multimedia data management such as index structure, and query manager; and secondly, components related to the Grid architecture like automatic load balancing techniques, and replica management policies. These two sets of components should seamlessly communicate with one another so that the overall goal of achieving multimedia data management over a distributed Grid environment is achieved. A database management system is primarily composed of two major blocks: a robust storage and efficient well-rounded retrieval mechanisms. An index structure is the backbone of both and is the useful connection between them. Traditional single dimensional index structures such as B-Tree [14] cannot handle the multidimensional feature vectors that

are used to represent the multimedia data. Though there are numerous multidimensional index structures such as those in [60][35] that can handle the multidimensional aspect of the multimedia data but they lack the capability to handle the high-level semantic relationships efficiently. In this dissertation, we proposed a generalized multidimensional index structure, GeM-Tree [47] designed for efficient management of multimedia data comprising of images and videos. In this chapter, we extend the usability of such multimedia index structures in a distributed Grid environment. We propose a distributed query management technique which embeds a content-based similarity search into a k-NN based algorithm in a distributed environment. We also introduce Grid specific components including a load balancing manager and semantic relationships manager between Grid nodes to enable the proposed multimedia database framework to be used successfully in a Grid environment. Extensive experiments with varied data loads and number of computation nodes are performed. The promising results demonstrate the usability of the proposed architecture and its potential extensibility.

The rest of the chapter is organized as follows. Section 9.2 presents a discussion on the related works in the field of distributed data management techniques and Data Grids. Section 9.3 lays down the overall framework of the proposed system and discusses the different components in details. Section 9.4 provides a detailed empirical study of the proposed system. Section 9.5 presents a brief conclusion and future direction of this research.

9.2 Related Work

In this section, we study the existing works on three important aspects: Distributed Multimedia Database Management Framework, Distributed Index Structures and Data Grids. Developing a successful distributed multimedia database management framework over the Grid environment is basically a seamless combination of all these different as-

pects. Thus, understanding the characteristics of each aspect would help to clearly define the capabilities that should be incorporated into the proposed architecture. Also, it would help to identify the limitations of each individual aspect when handling multimedia data in a distributed Grid environment. Thus, this survey of related works would enable to appreciate the necessity of an effective multimedia data management framework to be incorporated into a Grid architecture.

Though there are some proposed architectures for distributed multimedia database systems such as [20][116], none of them discusses the intrinsic database components; for example, index structures, and query manager in a distributed environment. [116] proposes an object-oriented database with an object request broker (a brokering server). It uses specialized repository servers for storing different multimedia data types. Using specialized servers enable some query functionalities such as content-based retrievals, and optimized access to be allocated at the repository level rather than at the database level. Thus clearly, here the data storage is separated from the main database functionalities. Hence, different database tuning and optimization techniques, those depending on both the data stored as well as on the user accessibility, such as query optimization, query cost determination, and index structures cannot be used seamlessly across the entire framework (since storage and database functionalities are separated). [20] also treats each multimedia data as an object and does not represent it as feature vectors. Hence, there is no well-defined index structure to facilitate efficient storage and retrieval based on the contents. The retrieval is done with object graphs where two levels of object graphs are used: namely local and central. Thus, the logical relationships among the multimedia data objects are captured but their relationships in terms of their content as well as storage strategies are not handled. Moreover, it doesn't propose any index structure, as robust as ones used in relational database systems, to be deployed in a distributed environment.

A replicated index structure for distributed data is proposed in [149]. It uses a dPi-tree based on the Pi-tree [72]. The index structure is replicated in each location of the distributed environment without message passing schemes. Though the proposed index structure can be utilized in a distributed environment, it is not tailored to suit the requirements of complex multimedia data. Firstly, although theoretically it is supposed to be able to handle multidimensional data, complex containment issues can arise. Secondly, it is a space-based index structure and hence does not support similarity search (based on distance calculation) naturally as distance-based index structures do. Finally, content-based retrievals, typical for multimedia data, are not embedded in the search methodologies. Although [134] proposes a distributed search tree in a dynamic distributed environment, it uses an extended binary leaf search tree. This limits the usability of such approach for multidimensional data representation. Also, [119] proposes a lazy update method for B+ tree in a distributed environment. However, B+ tree is not a suitable candidate to handle multimedia data as it cannot handle multidimensional data effectively.

[40] discusses the various approaches to design a Data Grid. It defines the requirements that a data Grid must satisfy and APIs necessary for its implementation. The design of the early data Grids was based on four major principles: mechanism neutrality, policy neutrality, compatibility with Grid infrastructure and uniformity of information infrastructure. The architecture is typically a two-layered structure, where the lower layer provides the data Grid specific services such as those related to the storage system and to the meta-data repository. The upper layer consists of the high-level components such as the replica selection service, and replica management service. The storage system utilized in the proposed architecture is basically a file structure and uses GridFTP for data transfers. There are no database components such as index structures or query managers associated with the storage and meta-data repositories. [82] defines a virtual Data Grid that is capable of encompassing the expertise of a large distributed diverse

multidisciplinary communication. It proposes general abstractions for representing data and computation. Further, it lays down a virtual data schema and an architecture that develops techniques for representing and maintaining data on an Open Grid Service Architecture (OGSA). These architectures are specifically for static data and do not address issues such as data synchronizations, and transaction data policies. To enable these frameworks to support dynamic data, services such as data regionalization, data synchronization, transactional management, data locality, event notification, and data load functions need to be introduced [203]. Additionally, data grids should have specific data distribution and data replication policies. For example, distribution approaches such as round-robin, Gaussian, random and Poisson can be used [203]. Data replication policy [139] is an important characteristics of a data Grid. The combination of data distribution and data replication policy defines the ability of a data region to support an application with minimum amount of data movement.

9.3 Overall Framework

Figure 9.1 presents the overall framework of the Distributed Multimedia Architecture over Grid. Each data node of the Grid is connected to all other nodes and has a multimedia database management system embedded in it. Each data node has a GridFTP server that takes care of the physical transfer of multimedia data objects from one node to another. The data is basically stored in a data server. The multimedia database framework is divided into four major components: a multimedia interface, a core DBMS engine, a content-retrieval engine and a high-level relationship manager. These four components interact with one another to achieve the major functionalities including query, and update. The multimedia interface handles the users' requests and access the other three components to provide the information requested by the user. The core DBMS engine manages the functions related to the database that store the multimedia data. It

is comprised of sub-components that are useful to designing a successful database system in a distributed Grid environment.

While components such as a transaction manager, and a query optimizer are the general components needed for a complete database engine design, components specific to a distributed environment such as an automatic load balancing system are also present in the proposed architecture. The content-retrieval engine houses the index structure and the access manager. The index structure along with the access manager handle the content-based retrieval queries. The index structure is a replicated multidimensional index structure which logically spans across the data nodes over the Grid. Thus, the index structure can be considered as a single unit organizing all the data that the entire Grid is comprised of. The high-level relationship manager maintains the semantic relationship among the multimedia data objects. It has three major sub-components: an affinity relationship metric, a local affinity update unit and a global affinity synchronization unit. The affinity relation metric basically captures and stores the high-level relationship between the multimedia data objects, based on the user access and feedback, while utilizing the Markov Model Mediator construct (discussed in details in Section 9.3.2). The local affinity update unit collects the user feedback and access patterns and updates the affinity relationship metric after specific time intervals. The global affinity synchronization unit updates the global affinity metrics based on the update of the local affinity metrics. The maintenance and use of the global affinity synchronization enables the users to issue queries transparently to the Grid without concerning themselves about the location and relationships of the multimedia data. Additionally the data Grid may contain other components specific to the Grid: namely a replica manager designed specially to cater the typical needs of multimedia data and applications; a failure management component designed to detect the non-functioning of a particular node and how to share the load among the remaining functioning nodes; etc.

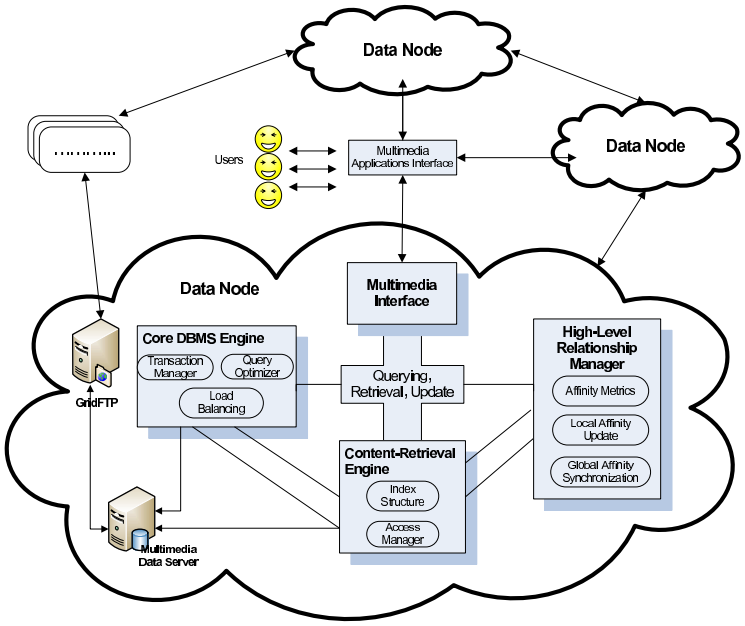


Figure 9.1: Overview of the proposed framework.

9.3.1 Replicated Multidimensional Index Structure

As mentioned in Section 9.1, an index structure is the backbone of an efficient database management system and is the link between the data storage and the retrieval engines. For the proposed framework, the index structure should be designed to satisfy two basic requirements. First, it should be able to handle multimedia data efficiently; and second, it should be possible to be deployed over a Grid environment. To satisfy the first requirement, the index structure should be a multidimensional index structure, so that it can handle the multidimensional representation of the data objects. Also, the similarity search methods supported by the index structure should be able to handle the semantic relationship among the multimedia data objects along with the content-level closeness while answering the queries. To satisfy the second requirement, the index structure should be able to span across several distributed data locations and consider characteristics of each while dealing with user requests. Basically, the k-NN algorithm, which is the standard similarity search algorithm for tree-based indexes, is customized to

support the content-based retrievals while considering the high-level semantic relationships among the data objects. In this chapter, we use GeM-Tree to organize only images and incorporate it into the proposed multimedia database management framework in a Grid environment. It should be pointed out here that videos can be used as well in the proposed framework without any loss of generality. Since images are simpler in representation and have simpler retrieval strategies, the proposed distributed architecture is tested with images as a proof of concept. It will be extended to include videos in the future.

We use a replicated indexing approach, similar in philosophy to the one proposed in [149]. The multimedia data is distributed across multiple data nodes of a Grid and the index structure is replicated across multiple sites as well. Each data node with an index replica has complete access to the local data. There is a logical link among the local index structures at each node. Thus while searching, the search results generated pick up the closest match to the submitted query among all the multimedia data present in the entire Grid repository utilizing this logical link. Each multimedia data is represented with a data signature that enables the system to uniquely identify a multimedia data object. The data signature F of a multimedia object is represented with two components, F_A and F_B.

$$F_A = \{x_1, x_2, ..., x_i\} \tag{9.1}$$

$$F_B = \{object_{id}, node_{id}, replica_{flag}\} \tag{9.2}$$

The feature vector representing the distribution of each multimedia data object is a union of the two parts represented as:

$$F = \{F_A \cup F_B\} \tag{9.3}$$

Where F_A represents the low-level feature vector of the multimedia data object and F_B is the unique identifier of the data object. The $object_{id}$ is the identifier of the multimedia data object, $node_{id}$ is the identification of the data node in the Grid where it belongs and $replica_{flag}$ is set to 1 or 0 depending on whether the particular data object has a

duplicate entry in any of the other data nodes. If 1, the replica manager of the node under consideration should be consulted whenever this particular data is accessed or modified. The benefit of using the data signature is that it makes the proposed framework transparent to the type of multimedia data object used. Any multimedia data can be represented with F. F might need slight extension to capture the characteristics of the particular multimedia data used.

Node Structures

The different node types and their structures are discussed in details in 4.2.3. In addition, a virtual link exists between the local index tree structures of the Data Grid, having a large number of semantically related data objects. The *High-level Relationship Manager* along with the *Global Affinity Synchronization* component determines which data nodes (locations) of the Grid have large amount of semantically related data objects. The index structures, belonging to those data nodes, are logically linked to represent a virtual single multidimensional index structure.

Insertion and Deletion

To insert a node in the index structure, the tree is recursively traversed until a candidate leaf node is identified. A particular sub-tree leading to the leaf node is chosen by selecting an intermediate node for which there is no or minimum increase in the covering radius. Essentially, a new object is inserted at the leaf node, and if it is full, a split is required followed by a rearrangement of the tree. Whenever a new data object is inserted into the index structure, an entry for its high-level semantic relationship with other multimedia objects is created in the *Local Affinity Update* component and the *Affinity Metrics*. As subsequent queries are issued, user feedback on the generated results generated are collected over time. They are used to populate/update values at *Affinity Metrics* and *Local Affinity Update* respectively. To delete a node in the index structure, the tree is

first traversed to locate the node. If it is an intermediate node, the pointer to the sub-tree it points to is set to zero, the memory is released and the links are rearranged. If it is a leaf-node, the actual data object at the repository pointed by it, is removed. As with any update, the *Local Affinity Update* component and the *Affinity Metrics* are modified to reflect the change.

9.3.2 Distributed Query Processing

The query processing component implements the most popular form of multimedia similarity search: namely, content-based retrieval. The Distributed Query Processing method is comprised of two major components. The first component is called the *Multimedia Application Interface* (as in Figure 9.2). It is a global query processing interface that takes in queries from the users and sends them across the data nodes of the Grid. At each data node, the queries are received by the local *Content-Retrieval Engine*, and is the second component of the Distributed Query Processor. The queries, once received by the individual local query processor are processed with the k-NN based similarity search algorithm of the multidimensional index structure. The k-NN algorithm searches the underlying data repository based on both the low-level contents of the multimedia data and their high-level semantic relationships. The search results, comprising of the k closest data objects to the query, are returned from each data node of the Grid, back to the *Multimedia Application Interface*. The search results, returned by each data node of the Grid have two pieces of information. First, the address of the multimedia data object at the local repository of the particular Grid node; and second, its *distance* from the query object. The result sets from each data node of the Grid are merged together and sorted based on the *distance*. The top k objects from the sorted list are retrieved from their corresponding local repositories to form the final query result. Figure 9.2 demonstrates the distributed query process.

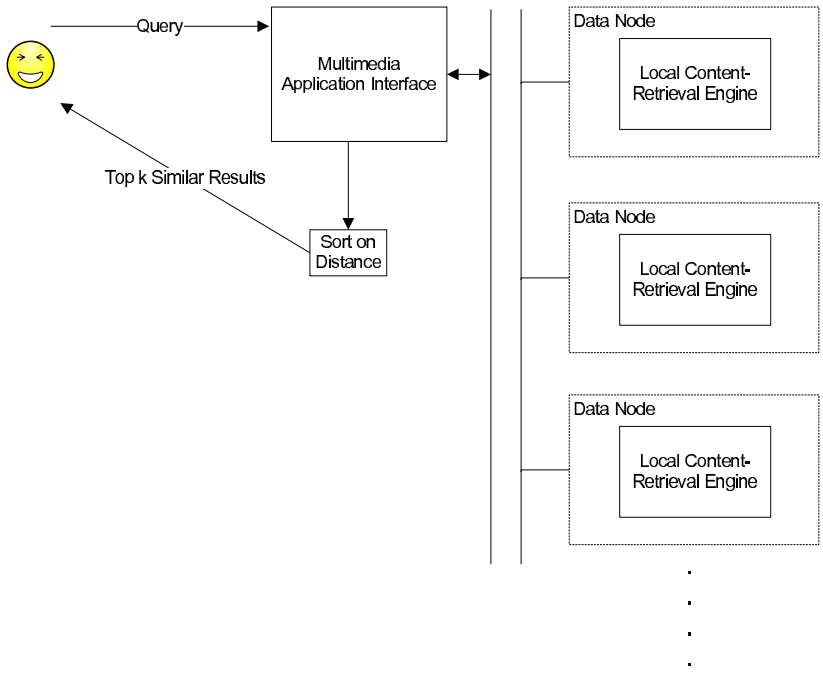


Figure 9.2: Distributed query processing.

High Level Relationship

A major attribute for the successful processing of the issued queries is the efficient maintenance and use of the high-level semantic relationship among the multimedia data objects. There are three major components of the *High-Level Relationship Manager*: namely the *Affinity Metrics*, the *Local Affinity Update* and the *Global Affinity Synchronization*. The *Affinity Metrics* stores the affinity relationships (as discussed in Section 4.3.1) of the multimedia data objects present in the local repository of the Grid node. The *Local Affinity Update* maintains the update information of the affinity values. The update process takes place whenever a new query is issued and the user feedback on the query results is obtained. The *Global Affinity Synchronization* helps in maintaining information necessary to synchronize the affinity relationships among the different nodes of the Data Grid. For example, let Image # 101 and # 369 be marked similar by the user in a particular query instance. Also, say Image # 101 belong to Node # 6 of the Grid and # 369 belong to Node # 2. A NXN matrix (N is the number of nodes in the Grid) is updated at two

locations (with same values): namely at the 6^{th} row and 2^{nd} column and 2^{nd} row and 6^{th} column where the affinity is increased between that particular pair of nodes. If the number of nodes of a Grid are huge, it is not practical to store the semantic closeness among all the nodes of the Grid. Instead, semantic relationships between the Grid nodes belonging to logical regions are maintained. As mentioned earlier, in this chapter we use image as the testbed for the prototype framework. Thus, in the rest of the chapter, we discuss the different functionalities that handle images.

The high-level image relationship utilized is captured using a stochastic construct called the Markov Model Mediator (MMM) [191], that maps the low level features and high level concepts in CBIR by capturing the image relationship as perceived by the user. It should be noted that any high-level image relationship capturing mechanism, similar to the affinity relationship, can be used in the proposed index structure without loss of generality.

Distributed Content-Based k-NN Similarity Search

Table 9.1 presents the k-NN similarity search algorithm in a distributed environment that supports Content-Based Image Retrievals (CBIR) over Grid. It follows a branch and bound technique as in [98]. The algorithm presented in Table 9.1 is for the metric region. Before ensuing the search on the metric region, a filtering stage is undertaken where the space-based index structures in each node of the Grid is searched to get the k closest feature-spaces. They are merged together and the metric search is executed on them. Although every index structure can have two basic similarity search paradigms: namely Range Search and k-NN Search, for CBIR based retrievals, k-NN approach models the information requirements of the users more naturally. Hence, we concentrate exclusively on the k-NN based search in this chapter. To issue content-based retrieval queries, an user must submit the query to the *Multimedia Application Interface*.

The low-level features are extracted from the submitted query image to represent the query in the same feature space as that of the indexed data. For example, if the images stored in the Grid are represented as color and texture features, when a query image is submitted, it should be also represented as a feature vector comprising of color and texture. The query, in the form of the feature vector, is submitted to the local multimedia interface at each Grid node. The affinity value is promoted in the multidimensional index

Table 9.1: Implementation of distributed content-based k-NN similarity search

```
Distributed_Similarity_Search(Q, N_child, r(Q), aff) { //CBIR over Grid.
  Get User Query;
  Extract the low-level feature values from the query;
  Submit the query across the Grid;
  For each Node of the Grid do: {
  Affinity_Promotion( ); //promotion of affinity value.
  ∀ O_r in N_child do: {
    if (O_r is an intermediate index node) {
    if (| d(O_M, Q) - d(O_r, O_M)| ≤ r(Q)+r(O_r)) {
      Compute d(O_r, Q) and aff(O_r, Q);
      if ((d(O_r, Q) ≤ r(Q)+r(O_r)) && (aff(O_r, Q) ≥ aff)) {
        Distributed_Similarity_Search(ptr(T(O_r)), Q, aff);
            //T(O_r): pointer to the subtree.
      }}}
    elseif (O_r is a leaf object){
      If the object qualifies the distance function and the affinity,
      add to the result set along with the distance d;
    }}}
    Merge result set from each Grid node;
    Sort result set on distance (similarity) with the query Q;
    Pick the k closest multimedia objects from the sorted result set;
}
```

structure as explained in Chapter 4. The index structure is traversed from the root to the leaf level. For each intermediate node of the index structure, the similarity between the indexed multimedia object and the query is determined in terms of both the low-level feature similarity and high-level semantic closeness. If the indexed multimedia object under consideration is more similar than the current k^{th} candidate in the priority

list, it is replaced with the indexed multimedia object, just considered. The priority queue is updated and the search continues recursively on the next closest candidate. Typically, the sub-tree contained in the candidate index entry is searched recursively. If the examined node is a leaf and satisfies the similarity conditions of distance and affinity, the corresponding data object is pushed into the result set. The result set itself is another priority queue, where the results are prioritized according to the distance and affinity score with the query object. Once, each Grid node has a result-set ready, the result-sets are sent back to the *Multimedia Application Interface*. Here, the result sets get merged and sorted. The top k objects are returned to the user as the query result. The user feedback is collected on each presented query result and components in the High-Level Relationship Manager are updated accordingly.

When the number of Grid nodes is large, it is not practical to involve all the nodes for every query. Generally, under those circumstances, initially, the query is submitted to a *reasonable* number of Grid nodes (eg. in the range between $100-200$). After receiving the query results for the first iteration, the *Global Affinity Synchronization* of the Grid nodes, which have data objects marked similar to the query object by the user, is consulted. The Grid nodes that are most similar to the Grid node under consideration (i.e., those that contain similar multimedia data objects) are selected. In the next iteration, the refined query is submitted to these selected Grid nodes and the process continues. The merged and sorted result set produced at the end of each query iteration is stored. After a few iterations (the number of iterations depends upon the Grid layout), all the previously stored result sets are merged and sorted again to get the top k results corresponding to the issued query across the entire Grid.

It should be pointed out here that to reduce the number of distance computations and use as many pre-computed distances as possible, a technique similar to [60] is introduced. In this method, in order to avoid unnecessary computation of distances between every indexed entry with the query, the covering radius of a parent node, its distance with

the child, along with its distance with the query object, is tested before a particular sub-tree is considered. It uses the classic metric space property of *triangular inequality* to formulate the checking condition. To reach a child node, its parent must have been traversed and thus there has to be a distance computation between the parent with the query. This distance computation is saved and reused for the next iteration. For example, one needs to start by computing the distance between the root with the query object. It then checks if any child of the root satisfies the qualification condition. If so, the corresponding child, along with its sub-tree, is considered.

9.3.3 Automatic Load Balancing

Any application in a Grid environment is incomplete without a proper *load balancing* functionality. Additionally, the domain that is dealt in this research (i.e. Multimedia Data) has an undeniable necessity for an effective load balancing component. This is because, multimedia data is much bulkier than ordinary text-based alpha-numeric data and the *quality of service* expected from multimedia applications is much higher than traditional text-based retrieval methods. Thus, whenever a particular Grid node is over-loaded, the load should be distributed among the less-utilized Grid nodes to attain a balanced computation cost. Moreover, as discussed in Section 9.3.2, when a query is issued to a Grid, it is simultaneously issued to all the Grid nodes. Query results from all the nodes of the Grid are collected and compiled to present the user with a single result set. Thus, if one/more nodes of the Grid is overloaded, it affects the performance of the entire Grid framework as the *Multimedia Application Interface* needs to wait till it receives responses from all the Grid nodes. We note that in some applications, although load balancing may result in a more balanced utilization of resources, it may however worsen the overall performance. For typical Multimedia Data Application, this is not the case though.

Table 9.2: Load balancing in the distributed multimedia database management framework

Load Balancing(n, i) { *//Load Balancing over Grid.*
For each iteration i {
Set min_time = minimum computation time among n Grids in iteration $i-1$;
Set max_time = maximum computation time among n Grids in iteration $i-1$;
Set n_{min} = node with minimum computation time;
Set n_{max} = node with maximum computation time;
if (number of data objects in $n_{max} \geq$ number of data objects in n_{min})
Set num_data_moved = (number of data objects in n_{max} − number of data objects is n_{min})/2;
else
Set num_data_moved = x; *//x is a pre-determined value.*
Move num_data_moved from n_{max} to n_{min};
}
}

We propose a load balancing algorithm as presented in Table 9.2. The basic heuristics used behind the proposed algorithm is computation_time ∞ number of indexed data points. Since, for developing the index structure and for subsequent queries, distances between pairs of multimedia objects need to be calculated. Thus the number of necessary distance computations increase with the increase of the number of data objects involved. Now, the total number of distances computed determine the overall computation time. So, to balance the computation time over the Grid, the number of multimedia data objects in each Grid node repository is to be balanced. The load balancing is typically not achieved in a single iteration but requires quite a few iterations. The number of iterations required depends on the data set involved. For each iteration, the maximum and minimum computation time for processing the submitted query is determined. Additionally, the Grid nodes having the maximum and minimum loads, are identified. Normally, the number of data points in the Grid node taking the maximum time should be more than that taking the minimum time. If the condition is not as it is predicted, it can be concluded that the imbalance is not related to the query but due to some other applications on the Grid. Under such circumstances a pre-defined number of data points are

moved from the most loaded node to the least loaded one. The pre-defined number (x) is determined based on the initial load in each Grid node. If the condition is satisfied, data points are moved from the most loaded to the least loaded such that they both end up having the same data load. The process is repeated until a desired balanced state is reached.

It should be mentioned here that the proposed algorithm is devised with the assumption that the Grid under consideration is a dedicated multimedia data management Grid with no other computation intensive applications running simultaneously. In other scenarios, this basic load balancing algorithm should be extended to include the different real-time factors that would decide on the amount of data to be moved. Such modifications, specific to particular Grid characteristics, should be possible without any loss of generality.

Basics of Load Balancing in a Distributed Environment

There are two approaches of dynamic load distributions: load-sharing and load-balancing. Where both load-sharing and load-balancing approaches tend to maximize the rate at which distribution systems work, when required resources are available, load-balancing additionally attempts to equalize the loads on the available nodes [147]. Additionally, load distribution algorithms can be categorized as: Sender-Initiated Algorithms [68], Receiver-Initiated Algorithms [68], Symmetrically Initiated Algorithms [135] and Adaptive Algorithm [136]. As the name suggests, in Sender-Initiated Algorithms, load-distribution is initiated by an overloaded sender that tends to send a task to an under-loaded receiver. In the Receiver-Initiated Algorithms, load distribution is initiated from an under-loaded node (receiver) to a overloaded node (sender). For Symmetrically initiated algorithms, both the overloaded as well as the under-loaded nodes initiate the load-distribution and possess the advantages of both the Sender-Initiated as well as the Receiver-Initiated algorithms. The Adaptive algorithms attempt to address the issues

that arise in the above three approaches. The main issue is the indiscriminate polling by the senders' negotiation component. The adaptive algorithms maintain the state of the relationship between the sender and the receiver and adapt themselves so as to scale well in larger systems. The load balancing algorithm proposed for our framework can be considered as the mixture of the Sender-Initiated and the Adaptive approach. It considers the states of all the nodes of the distributed environment but essentially transfers load from the most loaded node to the least loaded. A mixed approach is utilized because Grid environments have an increasing potential to grow. Thus, any function developed for a Grid environment should be scalable. By keeping track of the states of the overall system, the different load balancing parameters (such as the amount of load that should be transfered, identification of the nodes whose loads should be balanced) can be adjusted.

9.4 Empirical Study

We carried out an extensive analysis of the performance of the different critical functionalities of the proposed framework with a varied data set and in a varied environment. As mentioned before, in this chapter we used images as the multimedia object type and all subsequent experiments were performed on them. We used about 9000 images from different categories, collected from the COREL dataset [42]. These 9000 images are distributed among the data repositories of the different nodes of the Grid. The simulated distributed/Grid environment has 8 Intel-based nodes. The total storage available for users is around 320GB. Each node is simulated by a Pentium 4 processor with Hyper Threading at 3GHz. The images are represented with 12 features comprising of colors and textures.

We divided the experiments into three categories. At first, we analyze the relationship of the computation cost with the number of distribution nodes while generating the index tree. The experimental results, presented in Figure 9.3, demonstrate that as the number

of distribution nodes increases, the average computation time (measured in seconds) decreases. The same data load is distributed over multiple nodes and they all run in parallel, thus decreasing the computation overhead of individual node. We performed k-NN search for about 15 queries and averaged the results. The computation time for each instance for each query is the maximum of the computation time among the distributed loads. This is because, the *Multimedia Application Interface* waits for the query results from all the nodes before providing the aggregate query result to the user. It is interesting to note that the computation overhead during the k-NN search has no direct relationship with the number of distribution nodes used. Each node handles the query individually and the time taken for completing it depends on the data set (both the data load as well as the data content) in the particular node. As demonstrated in Figure 9.4, for Data Set A distributed over 4 nodes, the computation time increases steadily with the increase in the number of nodes. However, Figure 9.5 demonstrates that for a different data set B, the computation time drops when the number of nodes is 3 before rising again when number of nodes is 4.

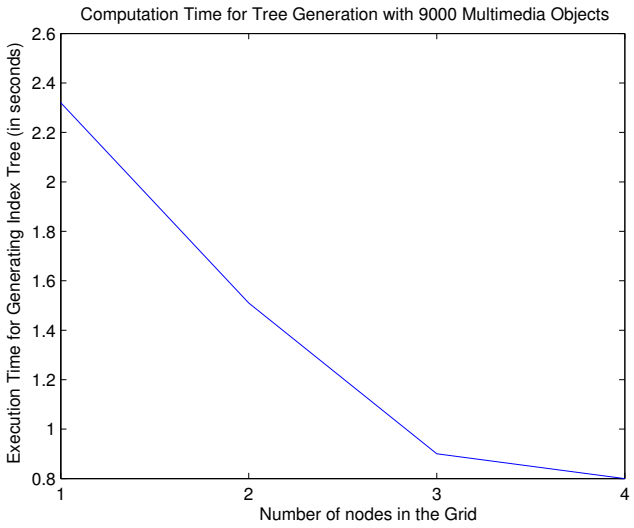


Figure 9.3: Relationship of the computation time with the number of distribution nodes during tree generation.

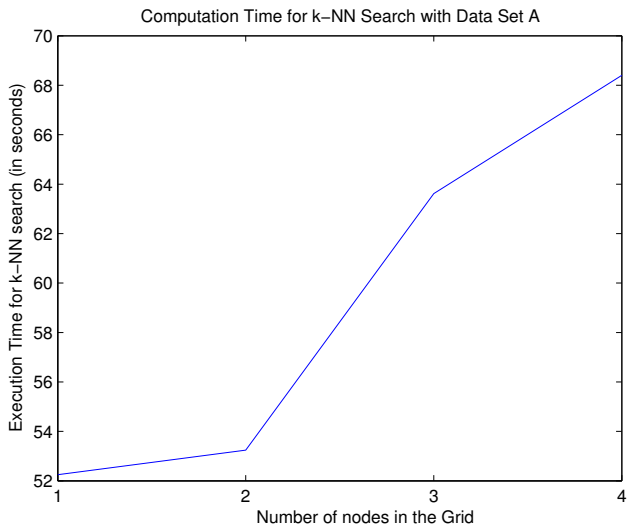


Figure 9.4: Relationship of the computation time with the number of distribution nodes during k-nn search for data set A.

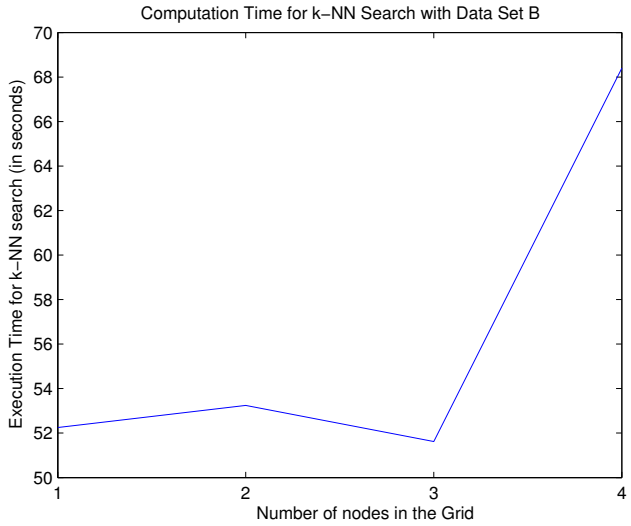


Figure 9.5: Relationship of the computation time with the number of distribution nodes during k-nn search for data set B.

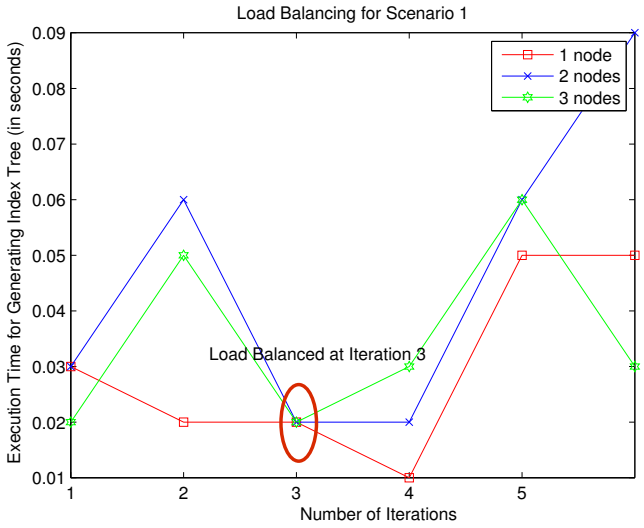


Figure 9.6: Experimental results for load balancing for data set I.

The average accuracy of query results is about $80 - 85\%$. We deployed a distance-based index structure, M-Tree, over the Grid which doesn't consider the high-level semantic relationships during the retrievals. The results obtained, although comparable in the computation overhead to the proposed framework, generated query results with very poor relevance (averaging as low as $15 - 20\%$).

The load balancing technique is demonstrated in Figure 9.6, 9.7, and 9.8, respectively. It should be noted that the load is balanced after different number of iterations for different data sets. We limited our examination for 5 iterations on an average, since in most of the cases we reached a considerable balanced load distribution within 5 iterations. Again, the variation is dependent on the characteristics of the data set used. In our experiment, we varied the number of data sets used in each case to bring a variation. Scenario I uses 500 data points, Scenario II uses 2300 data points and Scenario III uses 8500 data points, respectively.

From the detailed experimental analysis, it can be concluded that the proposed Distributed Multimedia Database Management Framework is capable of fulfilling the following requirements. First, it leverages the distributed environment of the Grid in econ-

omizing the computation overhead. Second, it is capable of supporting the popular multimedia retrieval requirements with relevant query results in a distributed environment. And finally, it successfully embeds functionalities typical to distributed environments, for example load balancing, into the multimedia environment, to make the proposed architecture adept for the Grid.

9.5 Conclusion and Future Works

In this chapter, we proposed a Distributed Multimedia Database Management Framework over a Grid. The framework introduced includes the important components necessary for storing and supporting multimedia applications over the Grid. A multidimensional replicated index structure was proposed that can support the popular multimedia retrievals based on contents. The framework introduces a stochastic construct, called the Markov Model Mediator, to capture and utilize the high-level semantic relationship among the multimedia objects. The novel inclusion of the high-level semantic relationship into the k-NN search algorithm, without violating the underlying indexed space, bridges the semantic gap and increases the relevance of query results manifold.

A load balancing approach for the multimedia data objects is also introduced, which successfully distributes the load across all the nodes of the Grid. In additions, intensive experimental analysis is performed with varied data set and different Grid configurations, which demonstrates that the proposed framework is a novel approach and a big step towards a full-fledged Multimedia Data Grid. The current framework can be extended in several directions. First, more Grid specific components such as replica managers, auto failure detection and recovery of the Multimedia Data Nodes can be added. Second, the current framework could be easily extended to support other forms of multimedia data such as videos, and documents within one seamless platform. And third, developing

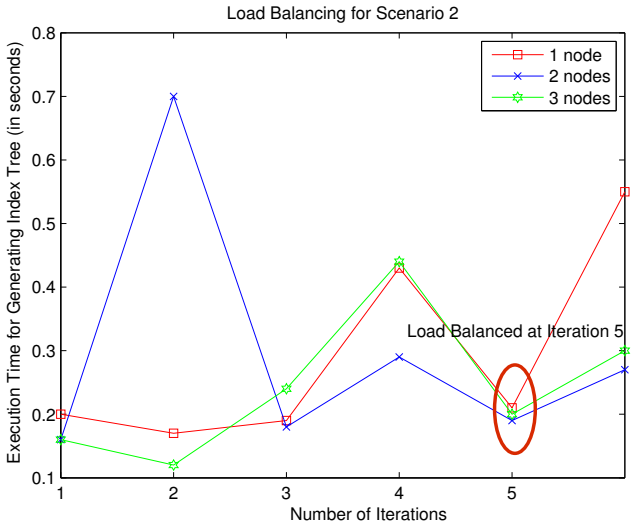


Figure 9.7: Experimental results for load balancing for data set II.

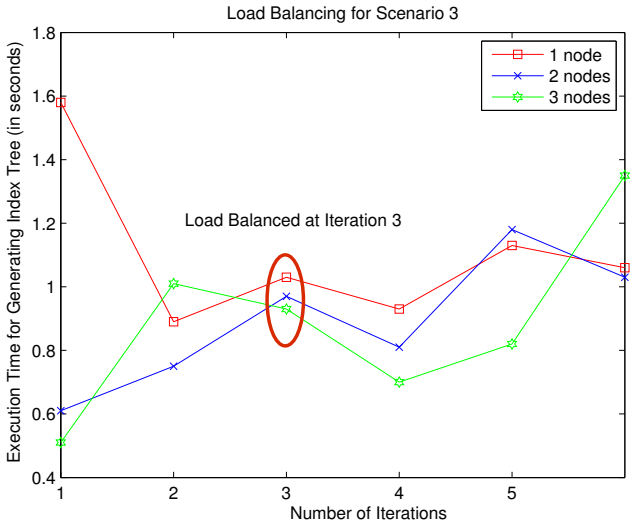


Figure 9.8: Experimental results for load balancing for data set III.

Multimedia Grid Services such as Content-Based Information Retrievals and Content-Based Multimedia search could be deployed.

CHAPTER 10

CONCLUSION AND FUTURE WORK

The main motivation behind the research presented in this dissertation is to develop a Multimedia Database Management Framework which would provide robust organization and retrieval of different genres of multimedia data from within one seamless structure. As discussed in Chapter 1, there are three major challenges in developing such a framework with a performance comparable to traditional database management systems, designed for alpha-numeric data viz. (1) the multidimensional and sometimes multi-modal feature set used to represent the multimedia data, (2) the semantic gap that exists between the high-level interpretation of a multimedia object and the low-level feature set and (3) the imprecise nature of the queries issued to multimedia data due to perception subjectivity. Thus, all the components designed for the multimedia database management framework should be able to address the above three issues efficiently for satisfactory performance. This dissertation focuses on the following three main aspects of a Multimedia Data Management Framework.

1. A generalized index structure is proposed for multimedia data which can handle the multidimensional representation, the semantic gap and the imprecise query nature quite efficiently (as corroborated by the results obtained so far). The proposed index structure, called GeM-Tree is a multidimensional distance-based index structure which uses a novel data signature technique to represent and organize both image and video data effectively in a metric space. In addition, it accommodates the hierarchical relationships that exist between the different classification units of video data. It supports the popular retrieval approaches viz. content-based image and video retrievals in its similarity search routines with the proposed k-NN based algorithms. It is also capable of supporting cross-multimedia object type similarity retrievals where users might wish to extract images or videos related to one another and to some particular concept. The novelty of the proposed index structure

is its flexibility which is demonstrated with the following capabilities: (i) it can accommodate different multimedia data representation from fixed length to variable length weighted features and (ii) it has the potential to support most of the popular multimedia data retrievals such as region-based image and video retrievals, object-based retrievals along with the basic content-based image and video retrievals as presented. Moreover, it uses a probabilistic construct called Markov Model Mediator to capture the high-level relationship between the multimedia data objects and use them in the similarity search methods to produce semantically related query image. Additionally, a distributed version of the proposed index structure is discussed which can manage multimedia data along with content-based information retrieval over a Grid architecture.

2. Since there are no known query refining strategy for distance-based index structures, a hybrid query refinement strategy is proposed for distance-based index structures which handles the high-level semantic relationship and low-level feature-wise similarity between multimedia data objects separately and thus overcomes the problems that arises while using query refinement techniques like [39] for scenarios when the low-level feature-wise similarity and the high-level semantic similarity between the multimedia data objects do not follow the same pattern.

3. A Social Network preview technique is proposed whereby large Social Networks are visualized with fewer number of nodes using graph similarity. Further, this social network visualization and analysis technique is used to model a multimedia data corpus into a Data Network where each data object acts as an actor (nodes) and their relationships, derived from the behaviors of the users who utilize them, act as the connections (edges). This modeling helps to understand the evolving semantic relationships among multimedia data in a collaborative environment. This further helps in designing the index structure so as to accommodate the storage requirements of applications designed for collaborative environments where multiple

users share and use the same data corpus. Further, this enables to introduce the high level semantic relationship while developing the index structure in operations such as insertion and deletion without violating the underlying metric space.

Though, this dissertation proposes to lay down the basic components, especially related to indexing mechanisms, required towards developing a successful multimedia database framework, its far from completion. There are several improvements that need to be made on the proposed designs and several additional components need to be developed in the future for a successful and efficient functioning of the framework.

10.1 Future Work

The proposed framework can be extended in the following directions to develop a complete Multimedia Data Management Framework that will cater the needs of different applications and user groups:

• **Intelligent Multimedia Index Structure Optimizer**

An intelligent multimedia index structure optimizer is planned to be developed by applying data mining techniques to the existing framework. As far as the literature survey goes, it can be concluded that such attempt was never done before i.e. to fine tune and optimize an index structure automatically based on the data type, access patterns and the past performance of the index structure. Historic access patterns of the users are to be stored and mined to generate decision rules which can be utilized to optimize the index structure and the retrieval algorithms. A knowledge repository is to be used which will store the past user feedbacks along with corresponding queries and query results.

An instance-based learning approach [2] will be adopted where the training sets will be stored verbatim and a distance function will be utilized to determine the closest member belonging to a training set to an unknown test instance. Once, the nearest training instance is evaluated, the class that it belongs to is predicted to be the class of the test

instance. But this nearest neighbor rule has several drawbacks. Firstly, it is very slow for large data sets which cannot be accepted in cases where the dataset is usually huge and the optimization process should be ideally dynamic and should change with the varying data set. Secondly, it performs poorly when the data set is noisy as no cross-validation is performed. Thirdly, it fails to capture the effect of different attributes of the feature set. These drawback can be largely overcome by using exemplar generalization. [189] proposes that using generalization with nested exemplars can achieve a high accuracy of classification whereas [220] disputed the claims stating that the results of [189] are domain specific. Later [153] proposed that if nesting and overlapping are altogether avoided, excellent results are achieved in most of the domains. Hence, [153] or [189] is to be used in this approach. An instance-based learning approach is chosen because the index structure developed here is also distance based and the multimedia objects are indexed based on a metric distance among themselves. Hence, it is an added benefit if the data analysis and prediction tool also follow such distance-based approaches. Also, the feature attributes used in this research are numeric, hence using a distance function to predict rules is possible. Thus, such learning approach will help to determine optimized decision rules that are derived from the historic data. For example, knowledge like different successful split policies along with the representative access patterns for each case might be fed to the system to generate a set of decision rules that will help to determine which split policy is appropriate when a particular access pattern is encountered. There can be several other decision rules that can be derived and need further investigations.

- **Additional Extensions of Existing Components**

1. Techniques such as Association Rule Mining (ARM) can be utilized to predict the semantic relationships of a new multimedia data object with the rest of the existing data objects or in other words to avoid the 'cold start'. Usually, when a new multimedia object is added to the database, it does not have any semantic relationship with any multimedia object. Thus, the similarity searches need to go

through fairly a good number of iterations, before the new object is accessed and it obtains an affinity value. This might result in poor retrieval results involving the newly added object initially, till the object gets trained and has an affinity value. Such a scenario can be avoided by applying ARM techniques, where based on the feature values and the existing affinity values (the presence of an affinity value between two multimedia objects can be considered as an association), the affinity value or at least the information if the new object is related to a particular multimedia data object can be deduced. This would help to increase the relevance of query results. Also as an extension, temporal relationship will be considered in the k-NN based similarity search techniques of the index structures for video data. At present, such information is not utilized in the index framework, but they are crucial and should be considered.

2. Document Indexing: Documents are considered as another popular multimedia data. A document can be considered to be constituted of texts, e-mails, xmls, etc. Thus, with the goal to have a single index framework to organize all types of multimedia data, introducing document indexing into the existing GeM-Tree along with images and videos should be considered. The main challenge in achieving it is to find a way to coherently represent a document as a numerical feature vector, which is currently the only known form in which a multidimensional index structure can handle any data. Mainly they are represented with key words or terms related to the context of the document. Parallel researches are being performed to determine the appropriate key terms of a document or to a context. Thus, utilizing those researches in document searching genres, a feature extraction and representation technique need to be devised to successfully index them. Also, a high-level semantic relationships among the documents need to be captured to assist in the retrieval process.

3. Supporting Traditional Alpha-Numeric Data Management: Since alpha-numeric data still forms the majority of the utilized and accessed data in the commercial scale, to ensure success of a robust DataSpace Management System, the traditional relational or object-relational database management frameworks should be integrated with the proposed multimedia database framework. Such integration should be as conflict-free as possible where the performance of one should not affect the performance of another.

4. Query Optimizer: Developing a query engine component for multimedia data is an important step. The cost evaluation rules used in a query optimizer for traditional data will not generate optimized results for multimedia data since they are way different and complicated that the traditional data. Thus, cost evaluation rules and metrics for multimedia data should be developed to improve the query performance.

• Develop Multimedia Data Management Framework for Applications in a Collaborative Environment

The multimedia database management system, capable of handling content-based retrieval, can be linked to social network applications. A sample scenario will be managing the media of a social network multimedia application such as Youtube based on user behavior observed in other related social network applications such as Facebook, where they share and post multimedia contents. The multimedia database systems can be also extended in Collaborative Search Environments, where users though explicitly may not be a part of any social network, forms an implicit relationship via their simultaneous search operations. Such concept will be very beneficial in improving multimedia web search.

Furthermore, the developed technique for managing information in collaborative environments can be applied in domains like disaster management or health-care. In disaster management applications, there is a constant influx of information from varied sources, some reliable while some not. The challenge is to manage this information (both as

texts as well as multimedia data in the form of captured images and videos) and provide an effective visualization and retrieval functionality, to help the common people as well as officials in charge of managing the situations, to evaluate and thus take crucial decisions. Collaboration is also an evolving direction for health-care management where multimedia data, in the form of medical imaging, forms a huge portion of the information. An effective representation, analysis and management of these data, to be utilized by different types of health-care professionals in different geographic locations, will be beneficial in helping the health-care systems in developing and under-developed nations. Doctors from across the globe will be able to collaborate and discuss on diagnosis based on medical images which would channel the knowledge of experts to people with lesser amenities.

It is envisioned that the multimedia database management framework proposed in this dissertation has great potential to be extended and improved so as to be a complete solution for organizing multimedia data efficiently and cater to different genres of applications, user groups and environments.

BIBLIOGRAPHY

[1] F. N. Abu-Khzam, N. F. Samatova, M. A. Rizk, and M. A. Langston, "The Maximum Common Subgraph Problem: Faster Solutions via Vertex Cover," in *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications*, pp. 367-373, 2007.

[2] D. Aha, "Tolerating Noisy, Irrelevent, and Novel Attributes in Instance-Based Learning Algorithms," *International Journal of Man-Machine Studies*, vol. 36, no. 2, pp. 267-287, 1992.

[3] H. Akaike, "A New Look at the Statistical Model Identification" *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716-723, 1974.

[4] T. Akutsu, "A Polynomial Time Algorithm for Finding a Largest Common Subgraph of Almost Trees of Bounded Degree," *IEICE Trans. Fundamentals*, vol. 76, pp. 1488-1493, 1993.

[5] A.A. Alatan, A.N. Akansu, and W. Wolf, "Multi-modal Dialogue Scene Detection Using Hidden Markov Models for Content-Based Multimedia Indexing," *Multimedia Tools and Applications*, vol. 14, no. 2, pp. 137-15, 2001.

[6] A. D. Alexandrov, W. Y. Ma, A. El Abbadi, and B. S. Manjunath, "Adaptive Filtering and Indexing for Image Databases," in *Proceedinds of SPIE Storage and Retrieval for Image and Video Databases*, pp. 12-23, 1995.

[7] C.J. Alpert and A.B. Kahng, "Recent Developments in Netlist Partitioning: A Survey," *Integration: The VLSI Journal*, vol. 19, no. 2, pp. 1-81, 1995.

[8] E. Ardizzone and L. M. Cascia, "Automatic Video Database Indexing and Retrieval," *Multimedia Tools and Applications*, vol. 4, pp. 29-56, 1997.

[9] W. G. Aref, A. Catlin, J. Fan, A. K. Elmagarmid, M. A. Hammad, I. F. Ilyas, M. S. Marzouk, and X. Zhu, " A Video Database Management System for Advancing Video Database Research," in *Proceedings of International Workshop on Multimedia Information Systems*, pp. 8-27, 2002.

[10] N. Babaguchi, Y. Kawai, and T. Kitahashi, "Event Based Indexing of Broadcasted Sports Video by Intermodal Collaboration," *IEEE Transactions on Multimedia*, vol. 4, no. 1, pp. 68-75, 2002.

[11] M.T. Barakat and P.M. Dean, "Molecular Structure Matching by Simulated Annealing. III. The Incorporation of Null Correspondences into the Matching Problem," *J Comput Aided Mol Des.*, vol. 5, pp. 107-117, 1991.

[12] J.M. Barnard, "Substructure Searching Methods: Old and New," *J. Chem. Inf. Comput.*, vol. 33, pp. 532-538, 1993.

[13] A. Basharat, Y. Zhai, and M. Shah, "Content Based Video Matching Using Spatiotemporal Volumes," *Jornal of Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 360–377, 2008.

[14] R. Bayer, "Binary B-Trees for Virtual Memory," in *Proceedings of SIGFIDET Workshop*, pp. 219-235, 1971.

[15] R. Bayer and E. M. McCreight, "Organization and Maintenance of Large Ordered Indices," *Journal of Acta Informatica*, vol. 1, no. 2, pp. 173-189, 1972.

[16] M.A. Beauchamp, "An Improved Index of Centrality," *Behavioural Science*, vol. 10, pp. 161-163, 1965.

[17] N. Beckmann, H. Kreigel, R. Schneider, and B. Seege, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," in *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pp. 322-331, 1990.

[18] S. Belongie, C. Carson, H. Greenspan and J. Malik, "Color and Texture-Based Image Segmentation Using EM and its Application to Content Based Image Retrieval," in *Proceedings of IEEE International Conference on Computer Vision*, pp. 675–682, 1998.

[19] S. Berchtold, D. A. Keim, and H. Kriegel, "The X-Tree: An Index Structure for High Dimensional Data," in *Proc. 22nd International Conf. on Very Large Database*, pp. 28–39, 1996.

[20] P. B. Berra, C. Y. R. Chen, A. Ghafoor, C. C. Lin, T. D. C. Little, and D. Shin, "Architecture for Distributed Multimedia Database Systems," *Journal of Computer Communication*, vol. 13, no. 4, pp. 217-231, 1990.

[21] A.R. Bloemena, "Sampling from a Graph," *Mathematical Centre Tracts*, pp. 85-86, 1964.

[22] V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. Van Dooren, "A Measure of Similarity Between Graph Vertices: Applications to Synonym Extraction and Web Searching," *SIAM Rev.*, vol. 46, no. 4, pp. 647-666, 2004.

[23] T. Bozkaya and M. Ozsoyoglu, "Distance-Based Indexing for High-Dimensional Metric Spaces," in *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pp. 357-368, 1997.

[24] F.J. Brandenburg, M. Himsolt, and C. Rohrer, "An Experimental Comparison of Force-Directed and Randomized Graph Drawing Algorithms," in *Proceedings of the Symposium on Graph Drawing*, pp. 76-87, 1995.

[25] R. L. Breiger, "The Analysis of Social Networks," in *Handbook of Data Analysis, edited by Melissa Hardy and Alan Bryman, Sage Publication*, pp. 505-526, 2004.

[26] H. Bunke, "Error Correcting Graph Matching: On the Influence of the underlying Cost Function," *IEEE Transactions of Pattern Analysis and Machine Intelligence*, vol. 21, no. 9, pp. 917-922, 1999.

[27] H. Bunke, X. Jiang, and A. Kandel, "On the Minimum Common Supergraph of Two Graphs," *Computing*, vol. 65, no. 1, pp. 13-25, 2000.

[28] J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121-167, 1998.

[29] R. Burt, "Toward a Structural Theory of Action: Network Models of Social Structure, Perception, and Action," *Academic Press*, ISBN: 9780121471507, 1982.

[30] F. Bourgeois and J.-C. Lassalle, "An Extension of the Munkres Algorithm for the Assignment Problem to Rectangular Matrices," *Communications of the ACM*, vol. 14, no. 12, pp. 802-804, 1971.

[31] M. Capobianco, "Statistical Inference in Finite Populations Having Structure," *Transactions of the New York Academy of Sciences*, vol. 32, pp. 401-413, 1970.

[32] M. Capobianco and O. Frank, "Comparison of Statistical Graph-Size Estimators," *Journal of Statistical Planning and Inference*, vol. 6, pp. 87-97, 1982.

[33] C. Carson, S. Belongie, H. Greenspan, and J. Malik, "Blobworld: Image Segmentation Using Expectation-Maximization and Its Application to Image Querying," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 8, pp. 1026-1038, 2002.

[34] F. Cazals and C. Karande, "An Algorithm for Reporting Maximal C-Cliques," *Theoretical Computer Sciences*, vol. 349, no. 3, pp. 484-490, 2005.

[35] K. Chakrabarti and S. Mehrotra, "The hybrid Tree: An Index Structure for High-Dimensional Feature Spaces," in *Proceedings of the IEEE International Conference on Data Engineering*, pp. 440-447, 1999.

[36] K. Chakrabarti, K. Porkaew, M. Ortega and S. Mehrotra, "Evaluating Refined Queries in Top-k Retrieval Systems," *Technical Report TR-MARS-00-04, University of California at Irvine*, 2000.

[37] K. Chakrabarti, M. Ortega, K. Porkaew and S. Mehrotra, "Query Refinement in Similarity Retrieval Systems," *IEEE Data Engineering Bulletin*, vol. 24, no. 3, pp. 3-13, 2001.

[38] K. Chakrabarti, K. Porkaew, M. Ortega, and S. Mehrotra, "Evaluating Refined Queries in Top-k Retrieval Systems," *IEEE Transaction on Knowledge and Data Engineeringn*, vol. 16, no. 2, pp. 256-270, 2004.

[39] K. Chakrabarti, K. Porkaew, and S. Mehrotra, "Efficient Query Refinement in Multimedia Databases," in *Proc. 16th IEEE International Conference on Data Engineering (ICDE)*, pp. 196-196, 2000.

[40] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," *Journal of Network and Computer Applications*, vol. 23, pp. 187–200, 2001.

[41] K. Chakrabarti, Hybrid Tree Code, *//www.ics.uci.edu/ kaushik/research/htree.html*, 2005.

[42] COREL STUDIO http://www.digitalriver.com/v2.0-img/operations/corelpps/desc/index.htm. Cited 9 March 2010.

[43] N. S. Chang and K. S. Fu, "A Relational Database System for Images," *Technical Report TR-EE 79-28*, Purdue University, 1979.

[44] N. S. Chang and K. S. Fu, "Query-By-Pictorial-Example," *IEEE Transaction on Software Engineering*, vol. 6, no. 6, pp. 519–524, 1980.

[45] K. Chatterjee and S.-C. Chen, "Affinity Hybrid Tree: An Indexing Technique for Content-Based Image Retrieval in Multimedia Databases," in *Proceedings of the IEEE International Symposium on Multimedia (ISM06)*, pp. 47-54, 2006.

[46] K. Chatterjee and S.-C. Chen, "A Novel Indexing and Access Mechanism using Affinity Hybrid Tree for Content-Based Image Retrieval in Multimedia Databases,"

International Journal of Semantic Computing (IJSC), vol. 1, no. 2, pp. 147-170, 2007.

[47] K. Chatterjee and S.-C. Chen, "GeM-Tree: Towards a Generalized Multidimensional Index Structure Supporting Image and Video Retrieval," in *Fourth IEEE International Workshop on Multimedia Information Processing and Retrieval (MIPR2008), in conjunction with IEEE International Symposium on Multimedia (ISM2008)*, pp. 631-636, 2008.

[48] K. Chatterjee and S.-C. Chen, "Hierarchical Affinity-Hybrid Tree: A Multidimensional Index Structure to Organize Videos and Support Content-Based Retrievals," in *Proceedings of 2008 IEEE International Conference on Information Reuse and Integration*, pp. 435-440, 2008.

[49] K. Chatterjee and S.-C. Chen, "Hybrid Query Refinement: A Strategy for Refining Multimedia Queries in terms of both Feature Space and Semantic Relationships in a Distance Based Index Structure," In *preparation to be submitted to ACM Transactions of Multimedia.*

[50] S.-C. Chen, M.-L. Shyu, C. Zhang, L. Luo, and M. Chen, "Detection of Soccer Goal Shots Using Multimedia Features and Classification Rules," in *Proceedings of the 4th International Workshop on Multimedia Data Mining*, pp. 36-44, 2003.

[51] S.-C. Chen, M.-L. Shyu, N. Zhao, and C. Zhang, "An Affinity-based Retrival System for Multimedia Authoring and Presentation," in *Proceedings of 11th Annual ACM International Conference on Multimedia (ACM-MM)*, pp. 446-447, 2003.

[52] S.-C. Chen, M.-L. Shyu, and C. Zhang, "Innovative Shot Boundary Detection for Video Indexing," edited by Sagarmay Deb, *Video Data Management and Information Retrieval.* Idea Group Publishing, ISBN: 1-59140546-7; pp. 217-236, 2005.

[53] S.-C. Chen, N. Zhao, and M.-L. Shyu, "Modeling Semantic Concepts and User Preferences in Content-Based Video Retrieval," *International Journal of Semantic Computing (IJSC)*, Vol. 1, no. 3, pp. 377-402, 2007.

[54] Y. Chen, and J. Z. Wang, "Image Categorization by Learning and Reasoning with Regions," *Journal of Machine Learning Research*, vol. 5, pp. 913-939, 2004.

[55] V. Cheng, C.-H Li, T.K. James, and C.-K. Li, "Dissimilarity learning for nominal data," *Pattern Recognition*, Vol. 37, no. 7, pp. 1471-1477, 2004.

[56] B.S. Cohn and M. Mariott, "Networks and Cenres of Intergration in Indian Civilization," *Journal of Machine Learning Research*, vol. 1, pp. 1-9, 1958.

[57] C. Colombo, A. Del Bimbo, and P. Pala, "Semantics in Visual Information Retrieval," *IEEE Multimedia*, vol. 7, no. 1, pp. 60-67, 2000.

[58] D. Comer, "Ubiquitous B-Tree," in *ACM Comput. Surv.*, vol. 11, no. 2, pp. 121-137, 1979.

[59] T. H. Coremen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms," *M.I.T Press*, ISBN: 9780262032933, 2001.

[60] P. Ciaccia,M. Patella, and P. Zezula, "M-tree: An Efficient Access Method for Similarity Search in Metric Spaces," in *Proc. 23rd VLDB International Conference*, pp. 426-435, 1997.

[61] M. Croitoru, B. Hu, S. Dashmapatra, P. Lewis, D. Dupplaw, and L. Xiao, "A Conceptual Graph Bsed Approach to Ontology Similarity Measure," in *Proceedings of the 15th International Conference on Conceptual Structures*, pp. 154-164, 2007.

[62] G. C. Cross and A. K. Jain, "Markov Random Field Texture Models," *IEEE Transaction of Pattern Recognition and Machine Intelligence*, vol. 5, pp. 25-39, 1983.

[63] CuVid Columbia Video Search System, *http://apollo.ee.columbia.edu/cuvidsearch/*

[64] D. Daneels, D. Campenhout, W. Niblack, W. Equitz, R. Barber, E. Bellon, and F. Fierens, "Interactive Outlining: An Improved Approach Using Contours," in *SPIE Proceedings of Storage and Retrieval for Image and Video Databases*, pp. 226-233, 1993.

[65] J. Dowe, "Content-based Retrieval in Multimedia Imaging," in *Proceedings of SPIE Storage and Retrieval for Image and Video Databases*, pp. 164–167, 1993.

[66] C.A. Duncan, M.T. Goodrich, and S.G. Kobourov, "Balanced Aspect Trees and their use for Drawing Very Large Graphs," in *Proceedings of the Symposium on Graph Drawing*, pp. 111-124, 1998.

[67] P. Eades and Q.-W. Feng, "Multilevel Visualization of Clustered Graphs," in *Proceedings of the Symposium on Graph Drawing*, pp. 101-112, 1997.

[68] D. Eager, E. D. Lazowska, and J. Zahorjan, "Adaptive Load sharing in Homogeneous Distributed Systems," *IEEE Trans. Softw. Eng.*, vol. 12, no. 5, pp. 662-675, 1986.

[69] S. Eickeler and S. Muller, "Content-based Video Indexing of TV Broadcast News using Hidden Markov Models," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 2997-3000, 1999.

[70] R.M. Emerson, "Exchange Theory Part 2: Exchange Relations and Network Structures," *Sociological Theories in Progress*, vol. 2, pp. 58–87, 1972.

[71] W. Equitz and W. Niblack, "Retrieving Images from a Database using Texture-Alogrithms from the QBIC System," *Technical Report RJ 9805, Computer Science, IBM Research Report*, 1994.

[72] G. Evangelidis, D. Lomet, and B. Salzberg, "The hB-Pi-Tree: A Modified hB-tree Supporting Concurrency, Recoverey, and Node Consolidation," in *Proceedings of Very Large Databases Conference*, pp. 551-561, 1995.

[73] Facebook at *http://en.wikipedia.org/wiki/Facebook*.

[74] Q. Feng, "Algorithms for Drawing Clustered Graphs," *PhD Thesis, Department of Computer Science and Software Engineering, The University of New Castle, Australia*, 1997.

[75] O. Frank, "Structure Inference and Stochastic Graphs," *FOA-Reports*, vol. 3, no. 2, pp. 1-8, 1969.

[76] R. Fagin, "Fuzzy Queries in Multimedia Database Systems," in *Proc. 17th ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems*, pp. 1-10, 1998.

[77] C. Faloutsos, "Searching Multimedia Databases by Content," *Kluwer Academic Publishers*, Boston, ISBN: 9780792397779, 1996.

[78] C. Faloutsos, W. Equitz, M. Flickner, W. Niblack, D. Petkovic, and R. Barber, "Efficient and Effective Querying by Image Content," *Journal of intelligent information systems*, vol. 3, no. 4, pp. 231-262, 1994.

[79] "http://en.wikipedia.org/wiki/F-score".

[80] I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," *Morgan Kaufmann Publishers*, ISBN: 978155860475, 1999.

[81] I. Foster "What is the Grid? - A Three Point Checklist," *GRIDtoday*, vol. 1, no. 6, pp. 22–25, 2002.

[82] I. Foster, "The Virtual Data Grid: A New Model and Architecture for Data-intensive Collaboration," in *Proceedings of the 15th International Conference on Scientific and Statistical Database Management*, pp. 11-22, 2003.

[83] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputer Applications*, vol. 15, no. 3, pp. 200-222, 2001.

[84] O. Frank, "Survey Sampling in Graphs," *Journal of Statistical Planning and Inference*, vol. 1, pp. 235-264, 1977.

[85] 0. Frank and D. Strauss, "Markov graphs," *Journal of the American Statistical Association*, vol. 81, pp. 832-842, 1986.

[86] O. Frank, "Random Sampling and Social Networks: A Survey of Various Approaches," *Mathematitiques et Sciences Humaines*, vol. 104, no. 26, pp. 19-33, 1988.

[87] D. Franks, R. James, J. Noble, and G. Ruxton, "Developing a Methodology for Social Network Sampling,", *Behavioral Ecology and Sociobiology*, vol. 63, no. 7, pp. 1079-1088, 2009.

[88] L. C. Freeman, "Centrality in Social Networks: Conceptual Clarification," *Social Networks*, vol. 1, no. 3, pp. 215-239, 1979.

[89] L. Freeman, S. Borgatti, and D. White, "Centrality in Valued Graph: A Measure of Betweenness Based on Network Flow," *Social Networks*, vol. 13, no. 2, pp. 141-154, 1991.

[90] N.E. Friedkin, "Theoretical Foundations of Centrality Measures," *American Journal of Sociology*, vol. 96, no. 6, pp. 1478-1504, 1991.

[91] H. Frohlich, A. Kosir, and B. Zajc, "Optimization of FPGA Configurations Using Parallel Genetic Algorithm," *Information Sciences*, vol. 133, no. 3, pp. 195-219, 2001.

[92] L.A. Goodman, "Snowball Sampling," *Annals of Mathematical Statistics*, vol. 32, pp. 148-170, 1961.

[93] M.S. Granovetter, "Network Sampling: Some First Steps," *American Journal of Sociology*, vol. 81, pp. 1287-1303, 1976.

[94] D. Greene, "An Implementation and Performance Analysis of Spatial Data Access Methods," in *Proc. 5th International Conference on Data Engineering*, pp. 606-615, 1989.

[95] H. Greenspan, J. Golberger, and A. Mayer, "Probabilistic Space-time Video Modeling via Piecewise GMM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 3, pp. 384-396, 2004.

[96] N. Gross, W.S. Mason, and W. McEachern, "Explorations in Role Analysis," in *Wiley Publishers*, ISBN: 9780471328025, 1958.

[97] B. Gunsel, A.M. Ferman, and A.M. Tekalp, "Video Indexing through Integration of Syntactic and Semantic Features," in *Proceedings of 3rd IEEE Workshop on Applications of Computer Visions*, pp. 90-95, 1996.

[98] A. Guttman, "R-trees: A Dynamic Index Structure for Spatial Searching," in *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, pp. 47-57, 1984.

[99] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Texture Features for Image Classification," *IEEE Trans. on Sys. Man. and Cyb.*, vol. 3, no. 6, pp. 610–621, 1973.

[100] M. Handcock, "Assessing Degeneracy in Statistical Models of Social Networks," *Journal of the American Statistical Association*, vo. 76, pp. 33–50, 2003.

[101] A. Hanneman and M. Riddle, "Introduction to Social Network Methods," online at *h*ttp://www.faculty.ucr.edu/hanneman/nettext/", 2005.

[102] J. Heer and D. Boyd, "Vizster: Visualizing Online Social Networks," in *Proceedings of the 2005 IEEE Symposium on Information Visualization (InfoVis05)*, pp. 33-40, 2005.

[103] J. M. Hellerstein, J. F. Naughton, and A. Pfeffer, "Generalized Search Trees for Database Systems," in *Proceedings of 21st Conference on Very Large DataBases (VLDB95)*, pp. 562-573, 1995.

[104] I. Herman, G. Melanon, and M. S. Marshall, "Graph Visualization and Navigation in Information Visualization: a Survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 24–43, 2000.

[105] M. Heymans, and A. Singh, "Deriving Phylogenetic Trees from the Similarity Analysis of Metabolic Pathways," *Bioinformatics*, vol. 19, no. 1, pp. 138-146, 2003.

[106] D. Hidovic, and M. Pelillo, "Metrics For Attributed Graphs Based On The Maximal Similarity Common Subgraph," *IJPRAI*, vol. 18, no. 3, pp. 299-313, 2004.

[107] M. Himsolt, "GraphEd: A Graphical Platform for the Implementation of Graph Algorithms," *Graph Drawing, Lecture Notes in Computer Science*, vol. 894, pp. 182-193, 1994.

[108] P.W. Holland, and S. Leinhardt, "An Exponential Family of Probability Distributions for Directed Graphs," *American Statistical Association*, vol. 76, pp. 33-65, 1981.

[109] P. Hong, Q. Tian, and T.S. Huang, "Incorporate Support Vector Machines to Content-based Image Retrieval with Relevance Feedback," in *Proceedings of IEEE international conference of Image Processing*, pp. 750-753, 2000.

[110] T.S. Huang and Y. Rui, "Image retrieval: Past, Present, and Future," in *Journal of Visual Communication and Image Representation*, vol. 10, pp. 1–23, 1997.

[111] M.L. Huang and P. Eades, "A Fully Animated Interactive System for Clustering and Navigating Huge Graphs," in *Proceedings of the Symposium on Graph Drawing*, pp. 374-383, 1998.

[112] T. S. Huang, S. Mehrotra, and K. Ramachandran, "Multimedia Analysis and Retrieval System (MARS) Project," in *Proceedings of 33rd Annual Clinic on Library Application of Data Processing-Digital Image Access and Retrieval*, 1996.

[113] C.H. Hubbell, "An Inputoutput Approach to Clique Identification," *Sociometry*, vol. 28, pp. 377-399, 1965.

[114] IBM Marvel: MPEG-7 Multimedia Search Engine, *http://www.research.ibm.com/marvel/*

[115] Y. Ishikawa, R. Subramanya, and C. Faloutsos, "MindReader: Querying Databases Through Multiple Examples," in *Proceedings of 24th International Conference on Very Large Data Bases, VLDB*, pp. 218-227, 1998.

[116] L. Ivan, M. Ricarte, and C. M. Tobar, "Towards an Architecture for Distributed Multimedia Databases," in *Proceedings of the 1996 IASTED/ISMM International Conference on Intelligent Information Management Systems*, pp. 68–71, 1996.

[117] M. Jamali and H. Abolhassani, "Different Aspects of Social Network Analysis," in *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 66-72, 2006.

[118] G. Jeh and J. Widom, "SimRank: A Measure of Structural-context Similarity," in *Proceedings of the Eights ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 538-543, 2002.

[119] T. Johnson and P. Krishna, "Lazy Updates for Distributed Search Structure," in *Proceedings of ACM SIGMOD Conference*, pp. 337-346, 1993.

[120] H. Jiang and A. Elmagarmid, "Spatial and Temporal Content-based Access to Hypervideo Databases," *VLDB Journal* vol. 7, no. 4, pp. 226-238, 1998.

[121] F. Jing, M. Li, H. J. Zhang, and B. Zhang, "An Effective Region-Based Image Retrieval Framework," in *Proceedings of ACM International Conference on Multimedia*, pp. 456-465, 2002.

[122] S.M. Kang, "A Note on Measures of Similarity Based on Centrality," *Social Networks*, vol. 29, no. 1, pp. 137-142, 2007.

[123] V. Kann, "On the Approximability of NP-Complete Optimization Problems," *Ph.D. Thesis, Department of Numerical Analysis and Computing Sciences, Royal Institute of Technology, Stockholm, Sweden*, 1992.

[124] L. M. Kaplan, et al., "Fast Texture Database Retrieval Using Extended Fractal Features," in *Proceedings of IS&T/SPIE Conference on Storage and Retrieval for Media Databases*, pp. 162-173, 1998.

[125] N. Katayama and S. Satoh, "Application of Multidimensional Indexing Methods to Massive Processing of Multimedia Information," *Systems and Computers in Japan*, vol. 31, no. 13, pp. 31-41, 2000.

[126] N. Katayama and S. Satoh, "The SR-Tree: An Index Structure for High-dimensional Nearest-Neighbor Queries," in *Proceedings of 1997 ACM SIGMOD*, pp. 369-380, 1997.

[127] A. Robles-kelly and E. R. Hancock, "Graph Matching using Adjacency Matrix Markov Chains," in *Proceedings of the British Machine Vision Conference*, pp. 384-390, 2001.

[128] A. Robles-Kelly, "A Thermodynamics Approach to Graph Similarity," in *Proceedings of the Digital Image Computing: Technqiues and Applications*, pp. 65-70, 2005.

[129] D. Kimelman, B. Leban, T. Roth and D. Zernik, "Reduction of Visual Complexity in Dynamic Graphs," in *Proceedings of the Symposium on Graph Drawing*, pp. 218-225, 1994.

[130] J.M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *Journal of the ACM*, vol. 46, no. 5, pp. 614-632, 1999.

[131] B. Klimt and Y. Yang, "The Enron Corpus: A New Dataset for Email Classification Research," in *Proceedings of the 15th European Conference on Machine Learning*, pp. 217-226, 2004.

[132] D.E. Knuth, "The Stanford GraphBase: A platform for Combinatorial Computing," *Addison-Wesley*, 1993.

[133] R. Krishnapuram, S. Medasani, J. Hwan, C. Y. Sik, and R. Balasubramaniam, "Content Based Image Retrieval Based on Fuzzy Approach," *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, vol. 16, no. 10, pp. 1185-1199, 2004.

[134] B. Kroll and P. Windmayer, "Distributing a Search Tree Among a Growing Number of Processors," in *Proceedings of ACM SIGMOD Conference*, pp. 265-276, 1994.

[135] P. Krueger and M. Livny, "The Diverse Objectives of Distributed Scheduling Policies," in *Proceedings of the IEEE Symposium on Distributed Computing Systems*, pp. 242-249, 1987.

[136] P. Krueger and R. Chawla, "The Stealth Distributed Scheduler," in *Proceedings of the 11th International Conference on Distributed Computing Systems*, pp. 336-343, 1991.

[137] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83-97, 1955.

[138] H. W. Kuhn, "Variants of the Hungarian Method for Assignment Problems," *Naval Research Logistics Quarterly*, vol. 3, pp. 253-258, 1956.

[139] H. Lamehamedi, B. Szymanski, Z. Shentu, and E. Deelman, "Data Replication Strategies in Grid Environments," in *Proceedings of the 5th International Conference on Algorithms and Architecture for Parallel Processing*, pp. 378-383, 2002.

[140] G. Levi, "A Note on the Derivation of Maximal Common Subgraphs of Two Directed or Undirected Graphs," *Calcolo*, vol. 9, pp. 341-352, 1972.

[141] Linkedin. *http://en.wikipedia.org/wiki/LinkedIn.*

[142] J. Z. Li, M. T. Ozsu, and D. Szafron "Modeling of Video Spatial Relationships in an Object Database Management System," in *Proceedings of International Workshop on Multimedia Database Management Systems*, pp. 124-132, 1996.

[143] J. Li, N. Allinson, D. Tao, and X. Li, "Multitraining Support Vector Machine for Image Retrieval," *IEEE transactions on image processings*, vol. 15, no. 11, pp. 3597-3601, 2006.

[144] Z. Liu, Y. Wang, and T. Chen, "Audio Feature Extraction and Analysis for Scene Segmentation and Classification," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 20, no. 1/2, pp. 61-80, 1998.

[145] D. Liu, K. Hua, K. Vu, and N. Yu, "Fast Query Point Movement Techniques with Relevance Feedback for Content-Based Image Retrieval," *Lecture Notes in Computer Science*, pp. 700-717, 2006.

[146] S. Liu, N. Cao, P. Moody, and T. Wang, "Trammel Map: Providing a Clear View of the Enterprise Social Network," in *IEEE Infovis 2007 (Interactive Poster)*, 2007.

[147] M. Livny and M. Melman, "Load Balancing in Homogeneous Broadcast Distributed Systems," in *Proceedings of the Computer Network Performance Symposium*, pp. 47-55, 1982.

[148] D. B. Lomet and B. Salzberg, "The hB-Tree: A Multiattribute Indexing Method with Good Guaranteed Performance," *ACM Transactions on Database Systems*, vol. 15, no. 4, pp. 625-658, 1990.

[149] D. B. Lomet, "Replicated Indexes for Distributed Data," in *Proceedings of International Conference of Parallel and Distributed Information Systems*, pp. 1-8, 1996.

[150] M. Loka, "A Method of Defining the Similarity of Images on the Basis of Color Information," *Technical Report RT-0030*, IBM Research, Tokyo, 1989.

[151] W. Y. Ma and B. S. Manjunath, "Netra: A Toolbox for Navigating Large Image Databases," in *Proceeding of IEEE International Conference on Image Processing*, pp. 568–571, 1997.

[152] W. Y. Ma and B. S. Manjunath, "Edge flow: A Framework for Boundary Detection and Image Segmentation," in *IEEE Transactions on Image Processing*, vol. 9, No. 8, pp. 1375–1388, 2000.

[153] B. Martin, "Instance-Based learning: Nearest Neighbor With Generalization," *MSc. Thesis, Dept. of Computer Science, University of Waikato*, 1995.

[154] B. S. Manjunath and W. Y. Ma, "Image Indexing Using a Texture Dictionary," in *Proceedings of SPIE Conference on Image Storage and Archiving System*, pp. 288–298, 1995.

[155] J.J. McGregor, "Backtrack Search Algorithms and the Maximal Common Subgraph Problem," *Softw., Pract. Exper.*, vol. 12, no. 1, pp. 23-34, 1982.

[156] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching," in *Proceedings. 18th International Conference on Data Engineering*, pp. 117-128, 2002.

[157] A. Motro, "Vague: A User Interface to Relational Databases that Permits Vague Queries," *ACM Transactions on Office Information Systems*, vol. 6, no. 3, pp. 187-214, 1998.

[158] S. Mukherjea, J.D. Foley, and S. Hudson, "Visualizing Complex Hypermedia Networks through Multiple Hierarchical Views," in *Proceedings of the Human Factors in Computing Systems (CHI '95)*, pp. 331-337, 1995.

[159] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32-38, 1957.

[160] A. Natsev, R. Rastogi, and K. Shim, "WALRUS: A Similarity Retrieval Algorithm for Image Databases," *IEEE Trans. on Knowledge and Data Engineering*, vol. 16, No. 3, pp. 301-316, 2004.

[161] M.E.J. Newman, "Assortative Mixing in Networks," *Phys. Rev. Lett.*, vol. 89, No. 20, pp. 208701-208704, 2002.

[162] M.E.J. Newman, "Finding Community Structure in Networks Using the Eigenvectors of Matrices," *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 74, No. 3, pp. 036104, 2006.

[163] W. Niblack, R. Barber, et. al., "The QBIC Project: Querying Images by Content Using Color, Textute and Shape," in *Proceedings of SPIE Storage and Retrieval for Image and Video Databases*, pp. 173-187, 1993.

[164] J. A. Orenstein and T.H. Merret, "A Class of Data Structures for Associate Searching," in *Proceedings of the ACM SIGMOD-PODS*, pp. 294-305, 1984.

[165] M. Ortega, Y. Rui, K. Chakrabarti, K. Porkaew, S. Mehrotra, and T. S. Huang, "Supporting Ranked Boolean Similarity Queries in MARS," *IEEE Transaction on Knowledge and Data Engineering*, vol. 10, no. 6, pp. 905-925, 1998.

[166] P. Palma, L. Petraglia, and G. Petraglia, "The Virtual Image in Streaming Video Indexing," in *Proceedings of International Conference on Dublin Core and Metadata for e-Communities*, pp. 97-103, 2002.

[167] A. Pentland, R. W. Picard, and A. Sclaroff, "Photobook: Content Based Manipulation of Image Databases," *International Journal of Computer Vision*, vol. 18, no. 3, pp. 233-254, 1996.

[168] H.H. Permuter, J. Francos and I.H. Jarmyn, "Gaussian Mixture Models of Texture and Colour for Image Database Retrieval," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 569–572, 2003.

[169] M. Petkovic and W. Jonker, "Content-based Video Retrieval: A Database Perspective," by *Kluner Academic Publishers*, ISBN: 9781402076176, 2004.

[170] K. Porkaew, K. Chakrabarti and S. Mehrotra, "Query Refinement for Multimedia Similarity Retrieval in MARS," in *Proceedings of the ACM Multimedia*, pp. 235-238, 1999.

[171] M. Patella, M-Tree Code, *http://www-db.deis.unibo.it/Mtree*, 2005.

[172] K. Porkaew, M. Ortega and S. Mehrotra, "Query Reformulation for Content Based Multimedia Retrieval in MARS," in *Proceedings of IEEE International Conference on Multimedia Computing and Systems ICMCS, Volume 2"*, pp. 47-751, 1999.

[173] J.R. Quinlan, "C4.5: Programs for Machine Learning," in *Morgan Kaufmann Publishers Inc.*, isbn 1-55860-238-0, 1993.

[174] S. T. Rachev, "The Monge-Kantorovich Mass Transference Problem and its Stochastic Applications," *Theory of Probability and its Applications*, vol. 29, no. 4, pp. 647-676, 1984.

[175] F. Ramsak, V. Markl, R. Fenk, M. Zirkel, K. Elhardt, R. Bayer, B. Forschungszentrum, and T. Mnchen, "Integrating the UB-Tree into a Database System Kernel," in *Proceedings of 26th International Conference on VLDB*, pp. 263-272, 2000.

[176] J. W. Raymond, E. J. Gardiner, and P. Willett, "RASCAL: Calculation of Graph Similarity using Maximum Common Edge Subgraphs," *Computer Journal*, vol. 45, pp. 631-644, 2002.

[177] J.W. Raymond and P. Willett, "Maximum Common Subgraph Isomorphism Algorithms for the Matching of Chemical Structured," *Journal of Computer Aided Molecular Design*, vol. 16, pp. 521-533, 2002.

[178] J. Rissanen, "Modeling by Shortest Data Description," *Automatica*, vol. 14, no. 2, pp. 465-471, 1978.

[179] J. T. Robinson, "The K-D-B-Tree: A Search Structure for Large Multidimensional Dynamic Indexes," in *Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data*, pp. 10-18, 1981.

[180] A. Robles-kelly and E.R. Hancock, "Graph Matching using Adjacency Matrix Markov Chains," in *Proceedings of the British Machine Vision Conference*, pp. 384–390, 2001.

[181] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest Neighbor Queries," in *Proc. 1995 ACM SIGMOD international conference on Management of Data*, pp. 71-79, 1995.

[182] Sellis, Timos, N. Roussopoulos, and C. Faloutsos, "The R+-Tree: A Dynamic Index for Multidimensional Objects," in *Proceedings of the 13th International Conference on Very Large Databases*, pp. 507-518, 1987.

[183] T. Roxborough and A. Sen, "Graph Clustering Using Multiway Ratio Cut," in *Proceedings of the Symposium on Graph Drawing*, pp. 291-296, 1997.

[184] B. Rubin and G. Davenport, "Structured Content Modeling for Cinematic Information," *SIGCHI Bulletin*, vol. 21, no. 2, pp. 78-79, 1989.

[185] Y. Rubner, C. Tomasi, and L. J. Guibas, "A Metric for Distributions with Applications to Image Databases," in *Proceedings of IEEE International Conference on Computer Vision*, pp. 59-66, 1998.

[186] Y, Rui, T. S. Huang, and S. Mehrotra, "Content-based Image Retrieval with Relevance Feedback in MARS," in *Proceedings of the 1997 International Conference on Image Processing*, pp. 815-818, 1997.

[187] Y, Rui, T. S. Huang, and S. Mehrotra, "Relevance Feedback: A Power Tool for Interactive Content-Based Image Retrieval," *IEEE Transactions on Circuit and Video Technology, Special Issue on Segmentation, Description, and Retrieval of Video Content*, vol. 8, no. 5, pp. 644-655, 1998.

[188] Y. Rui, T.S. Huang, S. Mehrotra, and M. Ortega, "Automatic Matching Tool Selection Using Relevance Feedback in MARS," in *Proceedings of 2nd International Conference On Visual Information Systems*, 1997.

[189] S. Salzberg, "A Nearest Hyperrectagular Learning Method," *Machine Learning*, vol. 6, no. 3, pp. 251-276, 1991.

[190] G. Schwarz, "Estimating the Dimensioan of Model," *Annals of Statistics*, vol. 6, no. 2, pp. 461-464, 1978.

[191] M.-L. Shyu, S.-C. Chen, M. Chen, C. Zhang, and C.-M. Shu, "MMM: A Stochastic Mechanism for Image Database Queries," in *Proceedings 5th International Symposium on Multimedia Software Engineering (MSE2003)*, pp. 188-195, 2003.

[192] J. R. Smith and S. F. Chang, "Transform Features for Texture Classification and Discrimination in Large Image Databases," in *Proceedings of IEEE International Conference on Image Processing*, pp. 407–411, 1994.

[193] J. R. Smith and S.-F. Chang, "Tools and Techniques for Color Image Retrieval," in *Proceedings of Storage and Retrieval for Image and Video Databases*, pp. 426–437, 1995.

[194] J. R. Smith and S. F. Chang, "Automated Image Retrieval Using Color and Texture," *Technical Report CU/CTR 408-95-14*, Columbia University, 1995.

[195] J. R. Smith and S. F. Chang, "VisualSeek: A Fully Automated Content-Based Query System," *Proceedings of ACM Multimedia*, pp. 87-98, 1996.

[196] J. R. Smith and S.-F. Chang, "Automated Binary Texture Feature Sets for Image Retrieval," in *Proceedings of ICASSP*, pp. 2239–2242, 1996.

[197] J. R. Smith and S.-F. Chang, "Visually Searching the Web for Content," *IEEE Multimedia Magazine*, vol. 4, no. 3, pp. 12-20, 1997.

[198] P. Smyth, "Statistical Modeling of Graph and Network Data," in *Proceedings of IJCAI Workshop on Learning Statistical Modelsfrom Relational Data*, 2003.

[199] T. A. Snijders, P. E. Pattison. G. L. Robins, and M. S. Handcock, "New Epecifications for Exponential Random Graph Models,", 2004.

[200] C. G. M. Snoek and M. Worring, "Goalgle: A Soccer Video Search Engine," in *Proceedings of IEEE International Conference on Multimedia and Expo*, 2003.

[201] C. Snoek and M. Worring, "Multimodal Video Indexing: A Review of the State-of-the-art," *Multimedia Tools and Applications*, vol. 25, no. 1, pp. 5-35, 2005.

[202] C. Stanfill and W. David, "Toward Memory-based Reasoning," *Communications of ACM*, vol. 29, no. 12, pp. 1213–1228, 1986.

[203] M. D. Stefano, "Distributed Data Management for Grid Computing," *Wiley*, ISBN: 9780471687191, 2005.

[204] R. O. Stehling, M. A. Nascimento, and A. X. Falcao, "On Shapes of Colors for Content-Based Image Retrieval," in *Proceedings of ACM International Workshop on Multimedia Information Retrieval*, pp. 171-174, 2000.

[205] M. Stricker and M. Orengo, "Similarity of Color Images," in *Proceedings of SPIE Storage and Retrieval for Image and Video Databases*, pp. 381–392, 1995.

[206] H. Sundaram and S.F. Chang, "Audio Scene Segmentation Using Multiple Features, Models And Time Scales," in *Proceedings of ICASSP*, pp. 2441–2444, 2000.

[207] M. Swain and D. Ballard, "Color Indexing," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11–32, 1991.

[208] H. Tamura, S. Mori, and T. Yamawaki, "Texture Features Corresponding to Visual Perception," *IEEE Transaction on Sys., Man. and Cyb.*, vol. 8, no. 6, pp. 460–473, 1978.

[209] C.A. Mills-Tettey, A. Stentz and M.B. Dias, "The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs," *Technical Report CMU-RI-TR-07-27, Robotics Institute, Carnegie Mellon University*, 2007.

[210] at *http://en.wikipedia.org/wiki/Twitter*.

[211] J.K. Uhlmann, "Satisfying General Proximity/similarity Queries with Metric Trees," *Inf. Proc. Lett.*, vol. 40 no. 4, pp. 175-179, 1991.

[212] M. Wagener and J. Gasteiger, "The Determination of Maximum Common Substructures by a Genetic Algorithm: Application in Synthesis Design and for the Structural Analysis of Biological Activity.," *Angew. Chem. Int. Ed. Engl.*, vol. 33, pp. 1189-1192, 1994.

[213] S. Wasserman and K. Faust, "Social Network Analysis: Methods and Application," *Cambridge University Press*, ISBN: 9780521387071, 1994.

[214] S. Wasserman and P. Pattison, "Logit Models and Logistic Regression for Social Networks: I. An Introduction to Markov Graphs and p*", *Psychometrika*, vol. 61, pp. 401-425, 1996.

[215] Y. Wang, Z. Liu, and J. Huang, "Multimedia Content Analysis Using Both Audio and Visual Clues," *Signal Processing Magazine*, vol. 17, pp. 12-36, 2000.

[216] J. Z. Wang, J. Li, and G. Wiederhold, "SIMPLIcity: Semantics-Sensitive Integrated Maching for Picture Libraries," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 9, pp. 947-963, 2001.

[217] D.J. Watts and S.H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440-442, 1998.

[218] R. Weiss, A. Duda, and D. K. Gifford "Content-based Access to Algebraic Video," in *Proceedings of International Conference on Multimedia Computing and Systems*, pp. 140-151, 1994.

[219] H. Wang and S.-F. Chang, "Compressed-domain Image Search and Applications," *Technical Report, Columbia Univ.*, 1995.

[220] D. Wettschereck and T. G. Dietterich, "An Experimental Comparison of the Nearest-Neighbor and Nearest-Hyperrectangular Algorithms," *Machine Learning*, vol. 19, no. 1, pp. 5-28, 1995.

[221] D. A. White and R. Jain, "Similarity Indexing with SS-tree," in *Proc. 12th International Conference on Data Engineering*, pp. 516-523, 1996.

[222] D. R. White and S. P. Borgatti, "Betweenness Centrality Measures for Directed Graphs," *Social Networks*, vol. 16, pp. 335-346, 1994.

[223] D. Willer, "Predicting Power in Exchange Networks: A Brief History and Introduction to the Issues," *Social Networks*, vol. 14, no. 3, pp. 187-211, 1992.

[224] J. K. Wu, "Content-based Indexing of Multimedia Databases," *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, vol. 9, no. 6, pp. 978-989, 1997.

[225] L. Wu, T. Bretschneider, "VP-EMD tree: An efficient Indexing Strategy for Data with Varying Dimension and Order," in *Proceedings of of International Conference on Imaging Science, Systems and Technology*, pp. 421-426, 2004.

[226] B- L. Yeo and M. M. Yeung, "Retrieving and Visualizing Video," *Communications of ACM*, vol. 40, no. 12, pp. 43-52, 1997.

[227] P. N. Yianilos, "Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces," in *Proc. 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 311-321, 1993.

[228] L. A. Zager and G. C. Verghese, "Graph Similarity Scoring and Matching," *Applied Mathematics Letters*, vol. 21, no. 1, pp. 86-94, 2007.

[229] P. Zezula, P. Ciaccia, and F. Rabitti, "M-tree: A Dynamic Index for Similarity Queries in Multimedia Databases," in *Technical Report 7, HERMES ESPRIT LTR Projects*, 1996.

[230] W.W. Zachary, "An Information Flow Model for Conflict and Fission in Small Groups," *Journal of Anthropological Research*, vol. 33, pp. 452-473, 1977.

[231] D. S. Zhang and G. Lu, "Generic Fourier Descriptors for Shape-Based Image Retrieval," in *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 425-428, August 2002.

[232] F. B. Zhan, and C. E. Noon, "Shortest Path Algorithms: An Evaluation Using Real Road Networks," *Transportation Science*, vol. 32, pp. 65–73, 1998.

[233] J. Zhong, H. Zhu, J. Li and Y. Yu, "Conceptual Graph Matching for Semantic Search," *Proceedings of the 10th International Conference on Conceptual Structures*, pp. 92-196, 2002.

[234] M. M. Zloof, "Query by Example," *AFIPS Conference Proceedings, National Computer Conference*, pp. 431-438, 1975.

APPENDICES

Table A.1: Degree centrality computation for the top 100 images of the multimedia data network for COREL datset

Image Id	**Degree Centrality**
9468	1
198	0.9714
4046	0.9429
447	0.9
4042	0.9
7240	0.9
4039	0.8714
2650	0.8571
3108	0.8571
6676	0.8571
7199	0.8571
9454	0.8571
7233	0.8429
163	0.8286
2264	0.8286
2624	0.8286
187	0.8143
7228	0.8143
7237	0.8143
7252	0.8143

Continued on Next Page...

Table A.1 – Continued

Image Id	**Degree Centrality**
7454	0.8143
7456	0.8143
7459	0.8143
7464	0.8143
7466	0.8143
7467	0.8143
7469	0.8143
7473	0.8143
7475	0.8143
7477	0.8143
7478	0.8143
7479	0.8143
7480	0.8143
7481	0.8143
7482	0.8143
7483	0.8143
7486	0.8143
7487	0.8143
7488	0.8143
7490	0.8143
7491	0.8143
7492	0.8143
7494	0.8143

Continued on Next Page...

Table A.1 – Continued

Image Id	**Degree Centrality**
7495	0.8143
9453	0.8143
9843	0.8143
2583	0.8
4028	0.8
4029	0.8
4030	0.8
4031	0.8
4035	0.8
4037	0.8
4041	0.8
4043	0.8
1510	0.7857
2630	0.7857
144	0.7714
162	0.7714
5091	0.7714
6138	0.7714
6656	0.7714
199	0.7571
1156	0.7571
1514	0.7571
1520	0.7571

Continued on Next Page...

Table A.1 – Continued

Image Id	**Degree Centrality**
6672	0.7571
6683	0.7571
9435	0.7571
165	0.7429
3071	0.7429
6109	0.7429
6139	0.7429
7204	0.7429
7205	0.7429
7206	0.7429
7207	0.7429
7210	0.7429
7219	0.7429
7220	0.7429
7224	0.7429
7226	0.7429
7227	0.7429
7229	0.7429
7232	0.7429
7241	0.7429
7244	0.7429
72534	0.7429
7255	0.7429

Continued on Next Page...

Table A.1 – Continued

Image Id	Degree Centrality
9443	0.7429
206	0.7286
2513	0.7286
2516	0.7286
2638	0.7286
2639	0.7286
2649	0.7286
2663	0.7286
3349	0.7286
4081	0.7286
5062	0.7286

Table A.2: Closeness centrality computation for the top 100 images of the multimedia data network for COREL datset

Image Id	Closeness Centrality
123	1
126	1
626	1
898	1
940	1
994	1
1019	1

Continued on Next Page...

Table A.2 – Continued

Image Id	**Closeness Centrality**
1204	1
1913	1
1919	1
2109	1
2150	1
2160	1
2169	1
2329	1
2339	1
2361	1
2528	1
2542	1
2859	1
2879	1
3112	1
3118	1
3123	1
3222	1
3233	1
3286	1
3317	1
3321	1
3347	1

Continued on Next Page...

Table A.2 – Continued

Image Id	**Closeness Centrality**
3397	1
3478	1
3532	1
3708	1
3721	1
3905	1
3980	1
4260	1
4821	1
4886	1
4910	1
4961	1
4984	1
5116	1
5117	1
5473	1
5600	1
5675	1
5753	1
5798	1
5846	1
5858	1
6187	1

Continued on Next Page...

Table A.2 – Continued

Image Id	**Closeness Centrality**
6188	1
7002	1
7007	1
7069	1
7082	1
7558	1
7736	1
7921	1
7922	1
8266	1
8278	1
8357	1
8604	1
8651	1
8805	1
8806	1
9042	1
9089	1
9169	1
9226	1
9228	1
9261	1
9288	1

Continued on Next Page...

Table A.2 – Continued

Image Id	**Closeness Centrality**
9304	1
9410	1
9466	1
9590	1
94	0.5
100	0.5
292	0.5
337	0.5
703	0.5
822	0.5
824	0.5
944	0.5
948	0.5
957	0.5
1100	0.5
1107	0.5
1241	0.5
1691	0.5
1786	0.5
1820	0.5
1826	0.5
2166	0.5
2302	0.5

Continued on Next Page...

Table A.2 – Continued

Image Id	Closeness Centrality
2330	0.5

Table A.3: Betweenness centrality computation for the top 100 images of the multimedia data network for COREL datset

Image Id	Betweenness Centrality
2151	1
1164	0.6961
165	0.6942
1119	0.6238
2094	0.6149
3885	0.5747
2782	0.5716
4081	0.5404
4641	0.5404
2519	0.524
3113	0.5101
9080	0.5022
2795	0.4978
9039	0.4929
6670	0.4869
2100	0.4654
3535	0.4547

Continued on Next Page...

Table A.3 – Continued

Image Id	**Betweenness Centrality**
1619	0.4534
2599	0.4497
2723	0.4446
3795	0.4244
3108	0.4214
1680	0.3895
7131	0.3854
2071	0.3822
7214	0.3783
4622	0.3752
2048	0.3723
9013	0.365
7240	0.3628
9363	0.3523
9440	0.3523
6020	0.3448
687	0.3322
4232	0.3312
5580	0.323
206	0.3225
7213	0.322
8782	0.3177
6015	0.3138

Continued on Next Page...

Table A.3 – Continued

Image Id	**Betweenness Centrality**
7199	0.3111
4640	0.3062
1625	0.3003
2870	0.2958
6301	0.2952
9435	0.293
2056	0.2845
9481	0.276
6153	0.275
4734	0.2726
7163	0.2715
2705	0.2581
2332	0.2566
7155	0.2528
3808	0.2464
35644	0.2446
8193	0.2433
357	0.2428
5062	0.2399
7232	0.2397
5907	0.2375
1332	0.2364
9443	0.2317

Continued on Next Page...

Table A.3 – Continued

Image Id	**Betweenness Centrality**
243	0.2315
1156	0.2291
1814	0.2278
4633	0.2273
2650	0.2252
5183	0.2248
2536	0.2117
2084	0.211
245	0.2015
7	0.2
9151	0.1993
5414	0.1973
9461	0.197
2054	0.1955
7090	0.1953
1011	0.1951
198	0.194
606	0.194
1951	0.1924
1360	0.1917
7900	0.1913
3398	0.1881
447	0.1868

Continued on Next Page...

Table A.3 – Continued

Image Id	**Betweenness Centrality**
8036	0.1868
1556	0.1833
1960	0.1802
2785	0.1773
5057	0.1765
4042	0.1725
2152	0.1702
5073	0.1702
3556	0.1666
4254	0.1661
6763	0.1656
5671	0.1655
7624	0.1651
2472	0.165

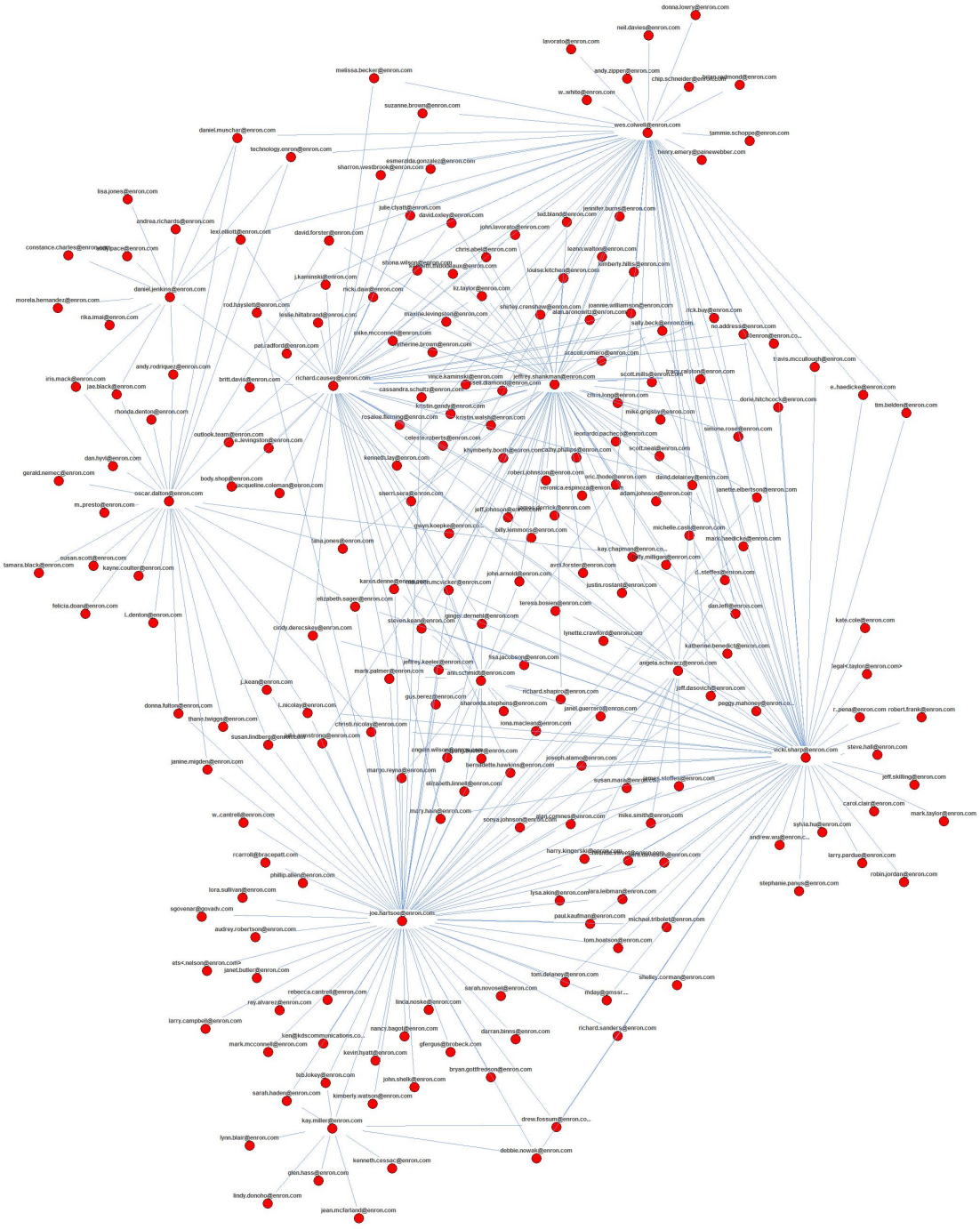


Figure A.1: Original graph for *enrongraph_500*

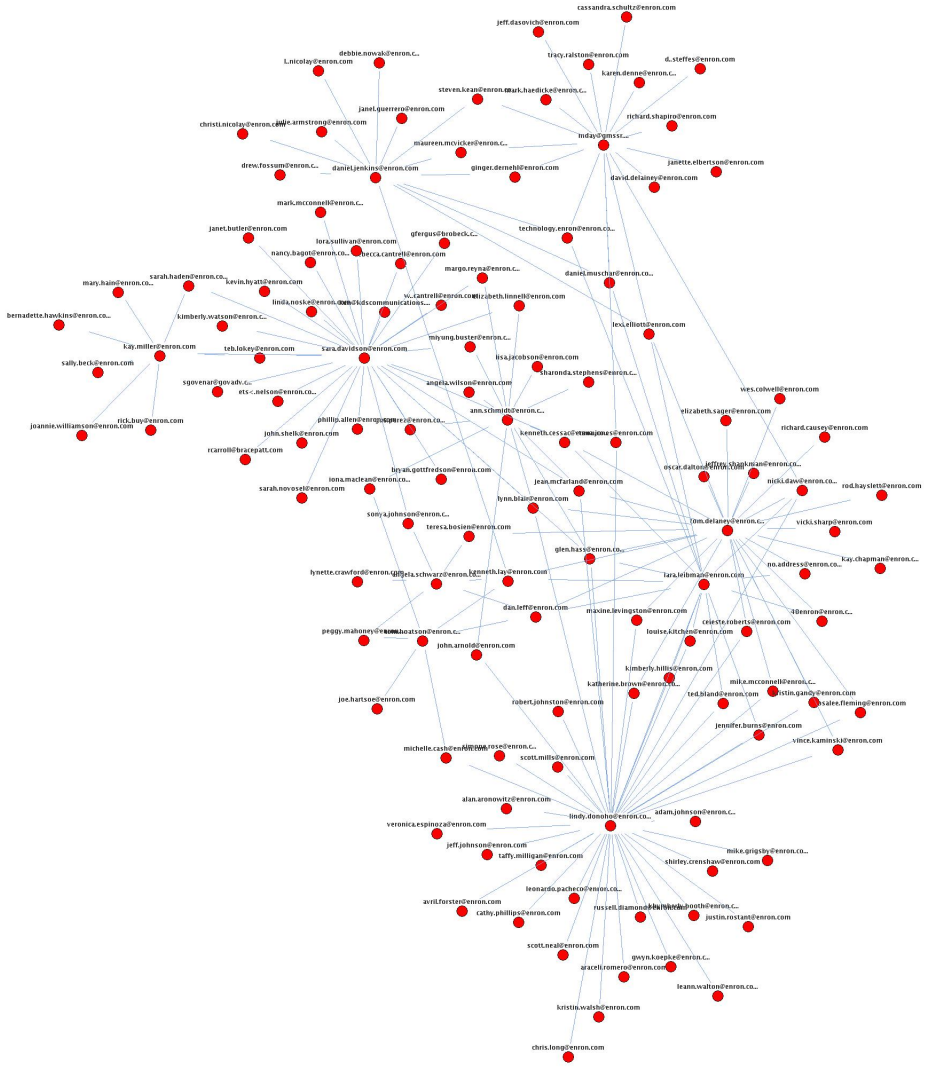


Figure A.2: Pre-determined representative graph for *enrongraph_500*

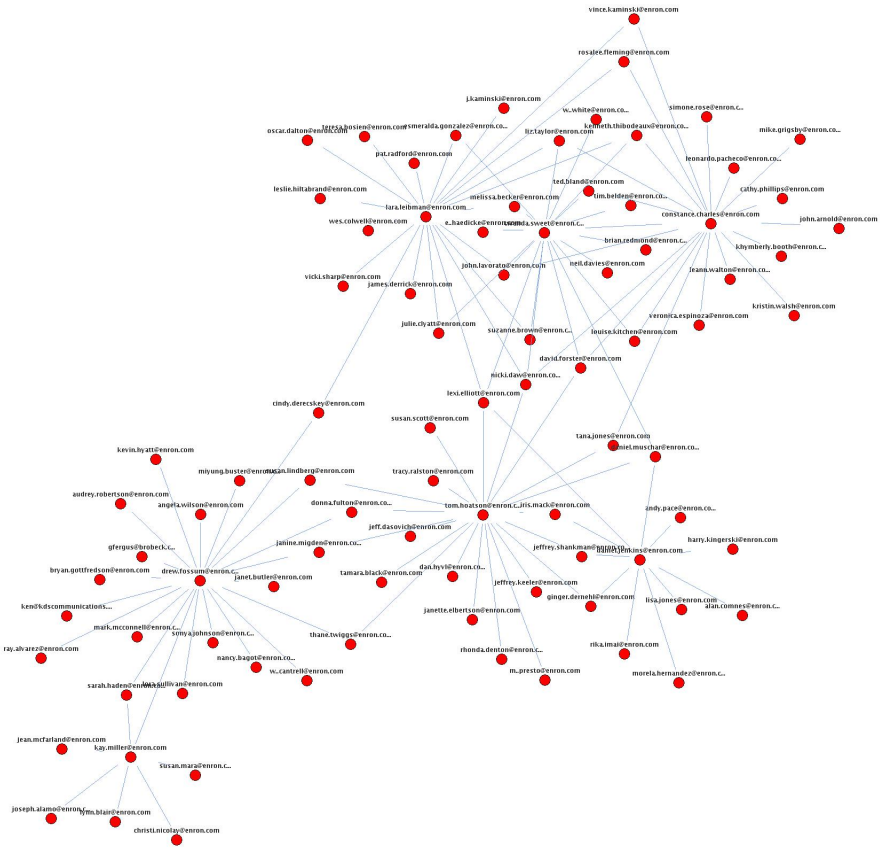


Figure A.3: Random representative graph for *enrongraph_500*

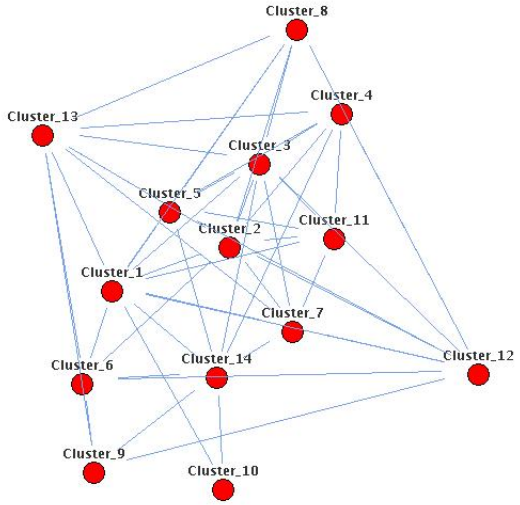


Figure A.4: Clustered graph for *enrongraph_500*

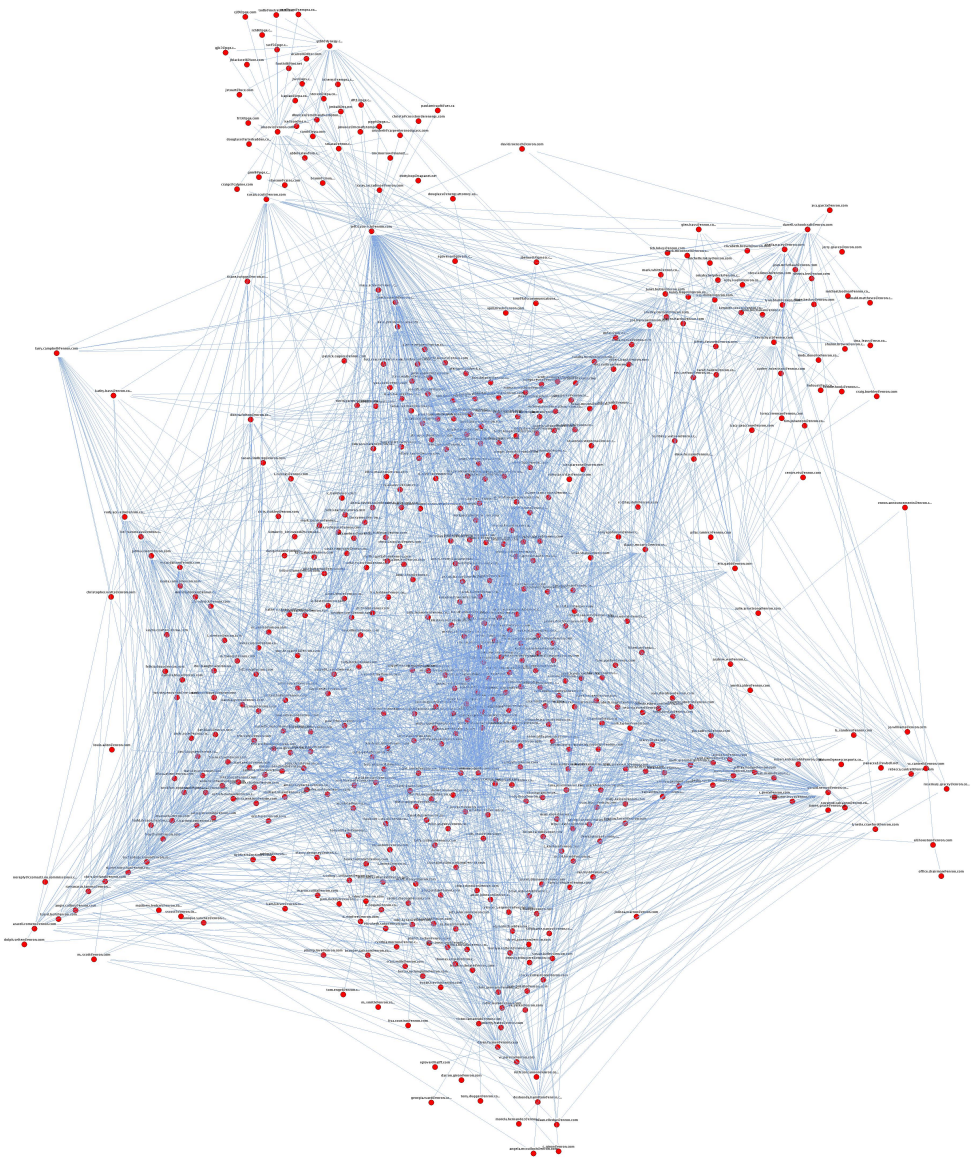


Figure A.5: Original graph for *enrongraph_5000*

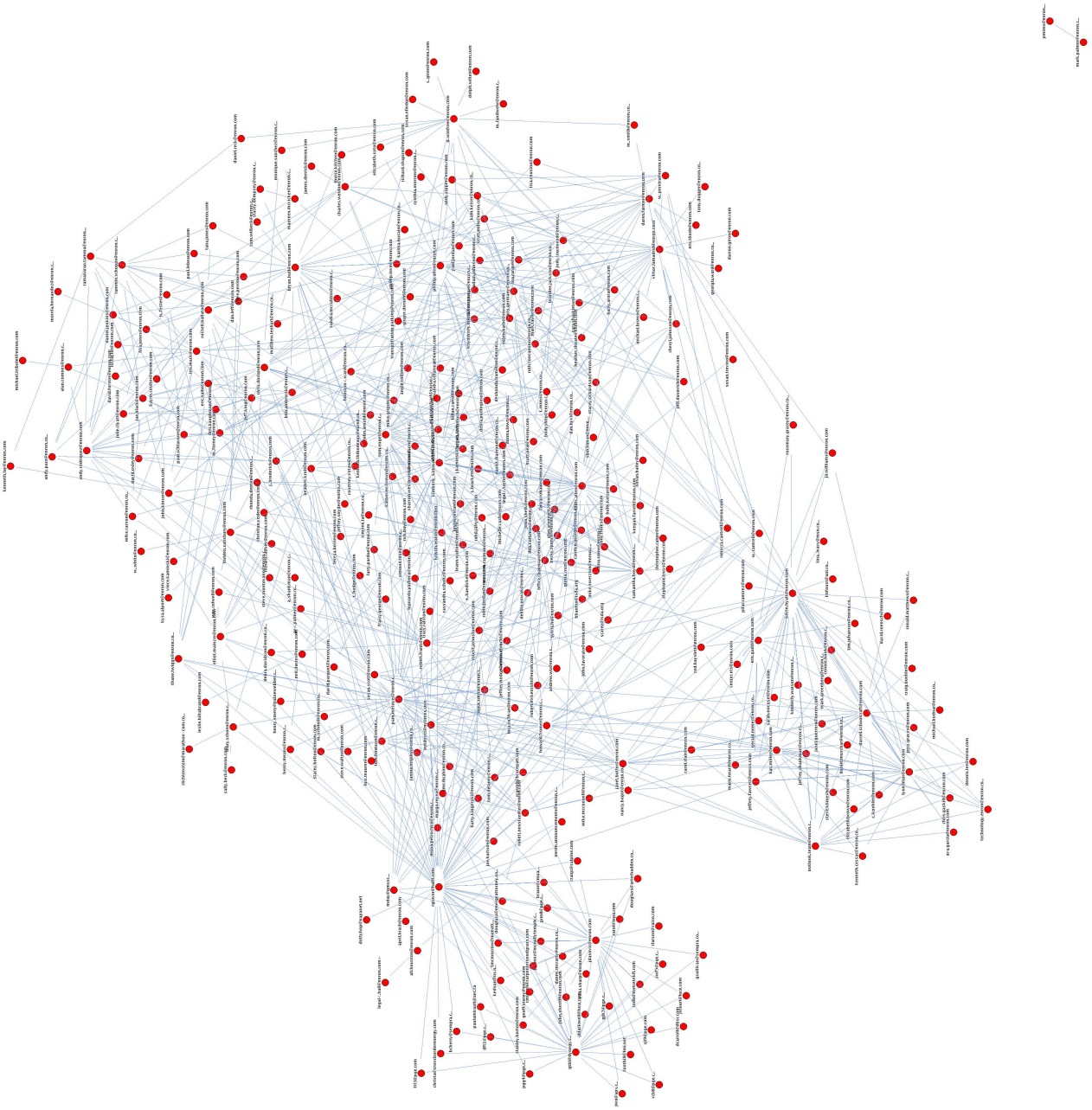


Figure A.6: Pre-determined representative graph for *enrongraph_5000*

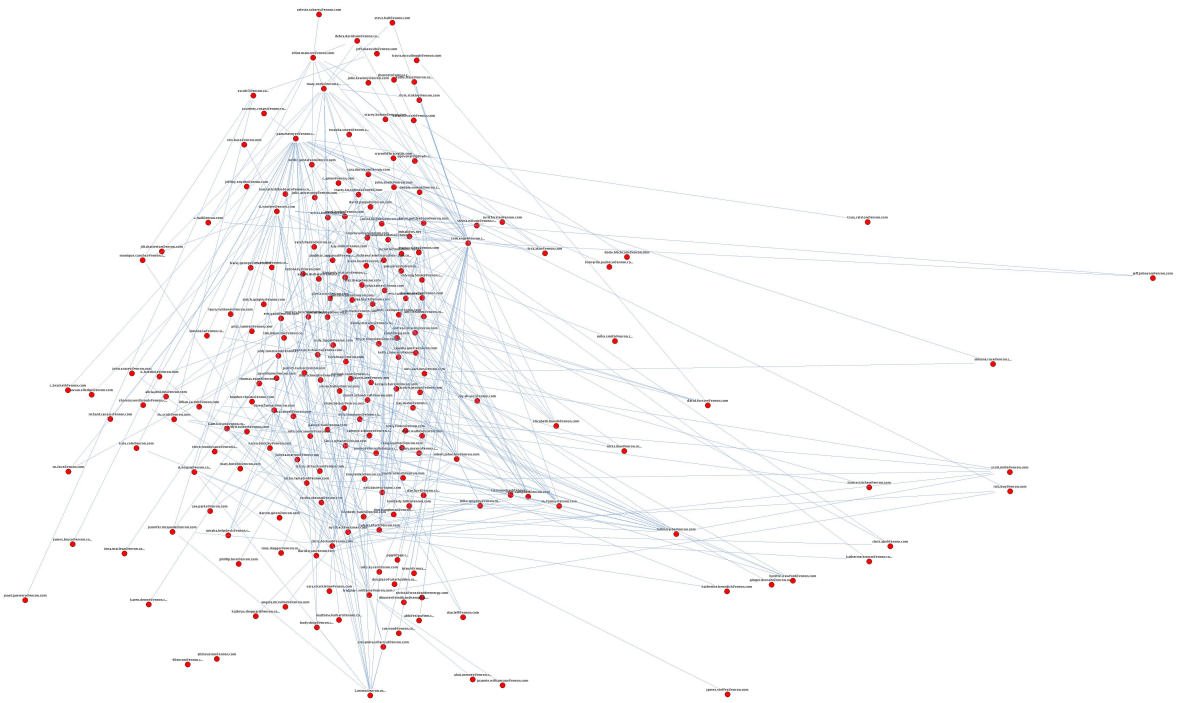


Figure A.7: Random representative graph for *enrongraph_5000*

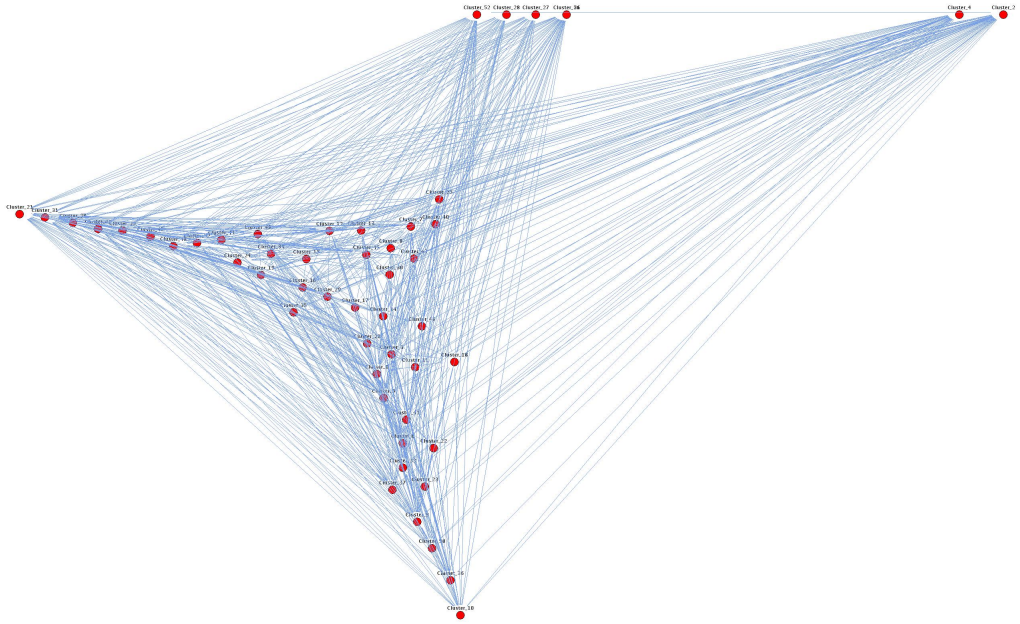


Figure A.8: Clustered graph for *enrongraph_5000*

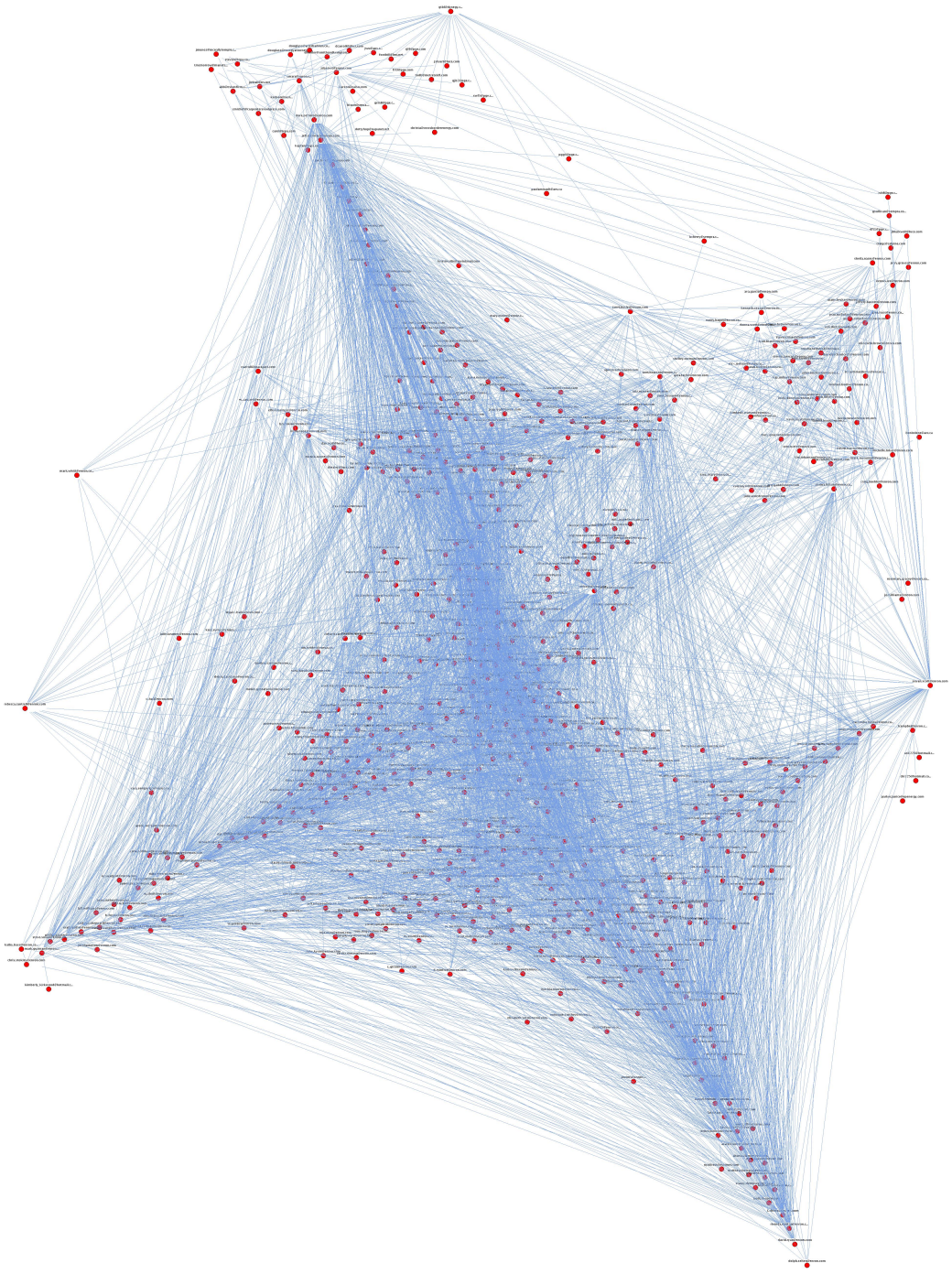


Figure A.9: Original graph for *enrongraph_10000*

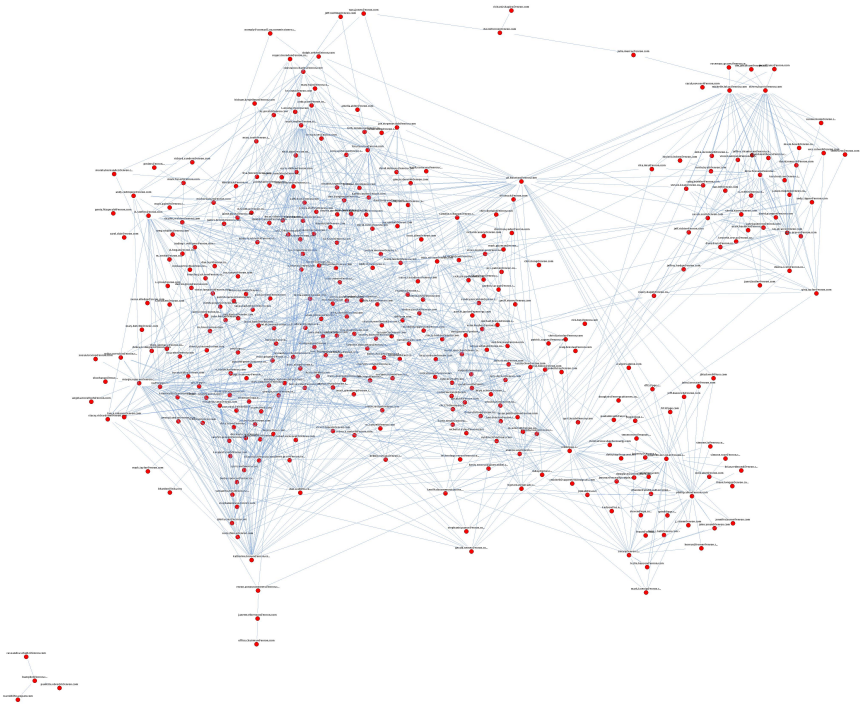


Figure A.10: Pre-determined representative graph for *enrongraph*_10000

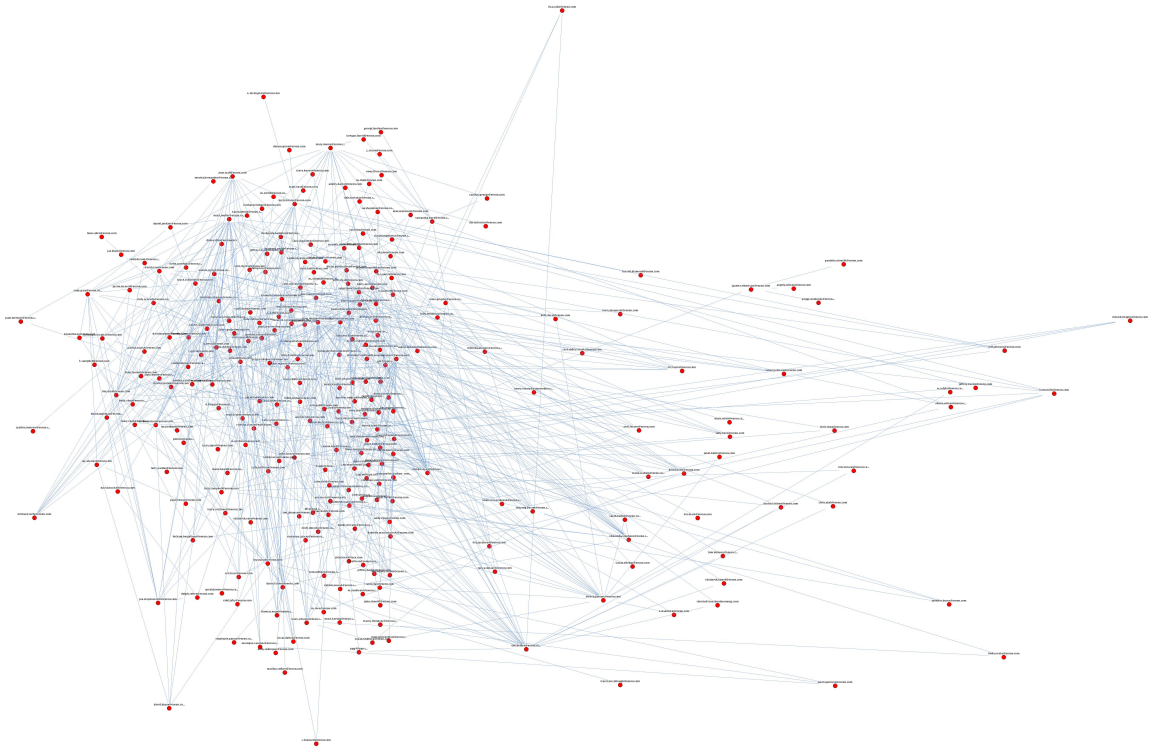


Figure A.11: Random representative graph for *enrongraph*_10000

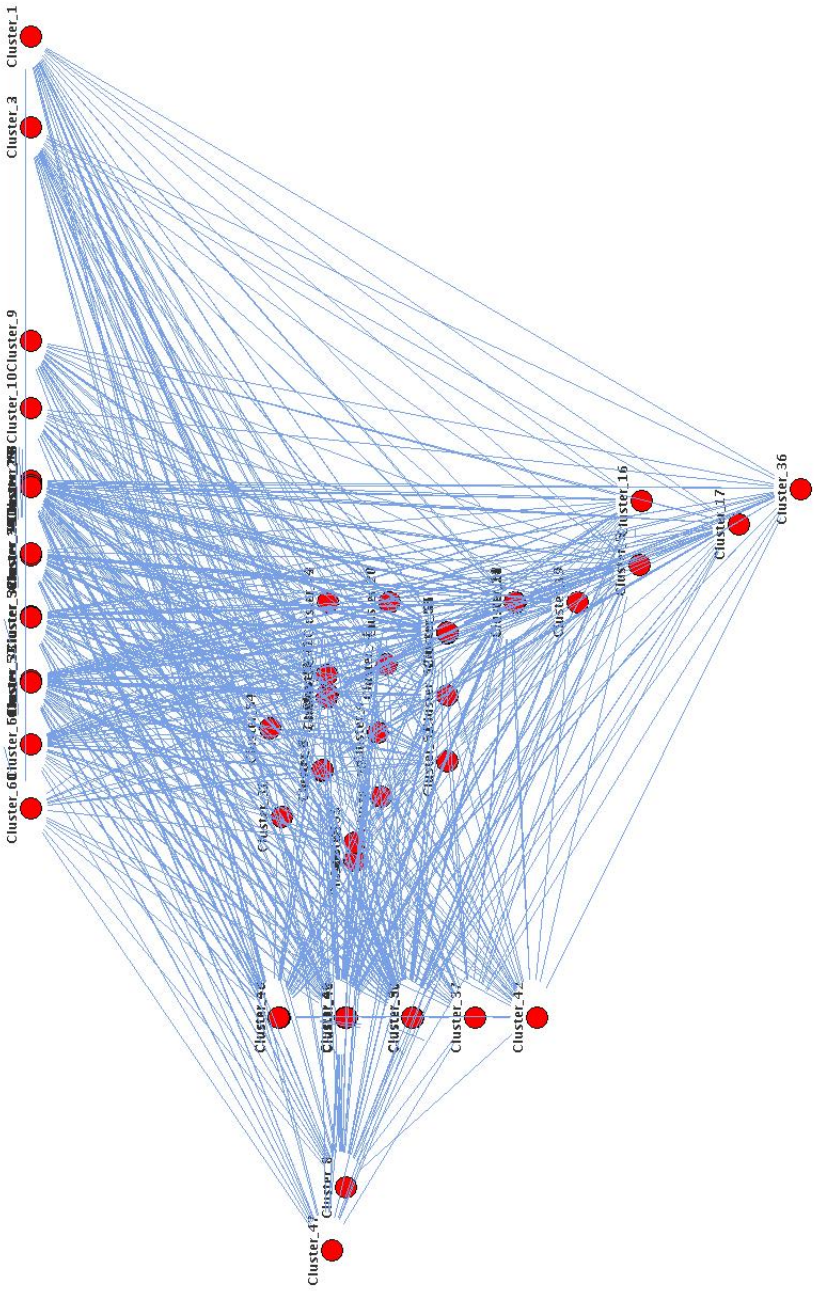


Figure A.12: Clustered graph for *enrongraph_10000*

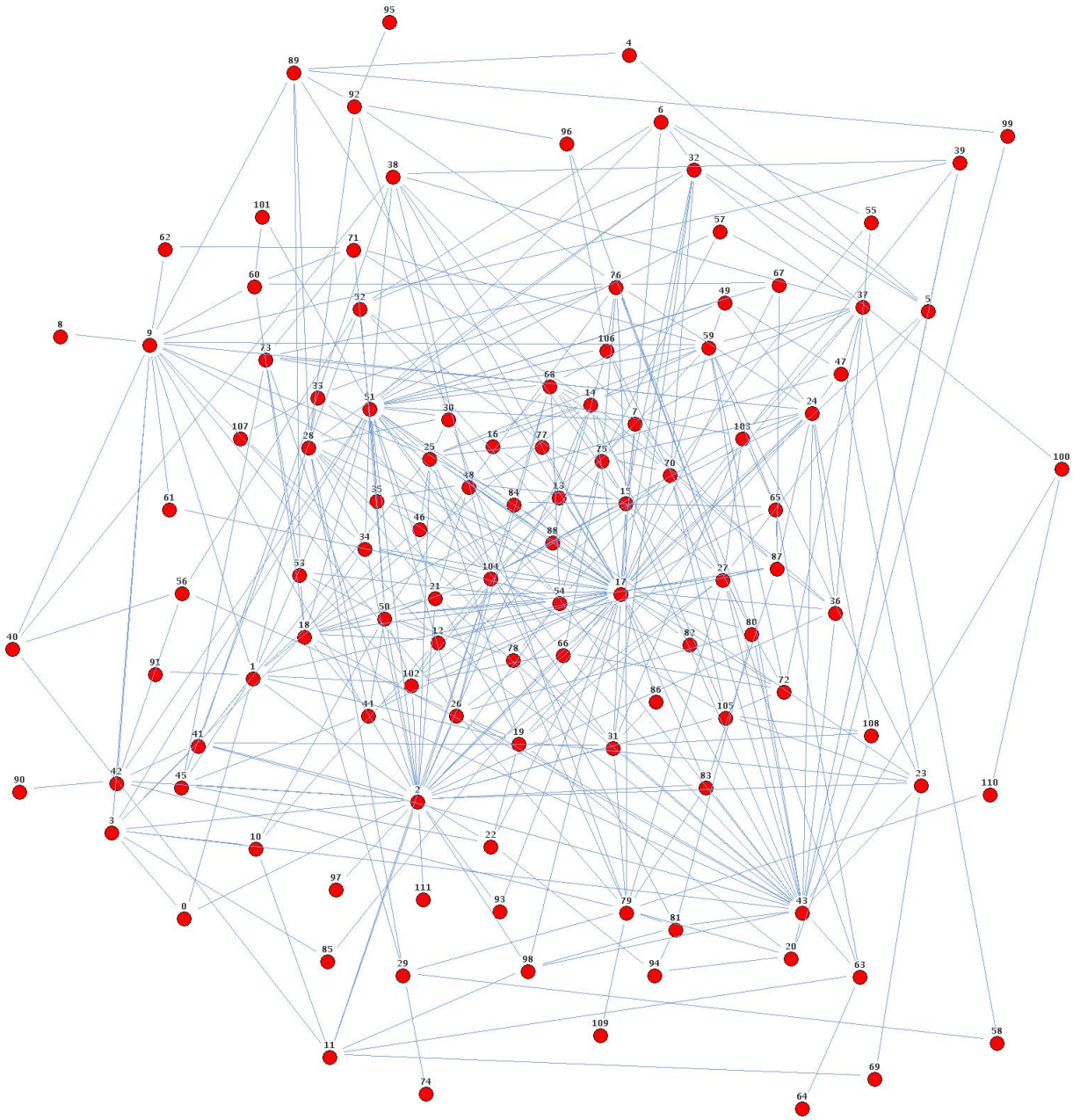


Figure A.13: Original graph for adjnoun

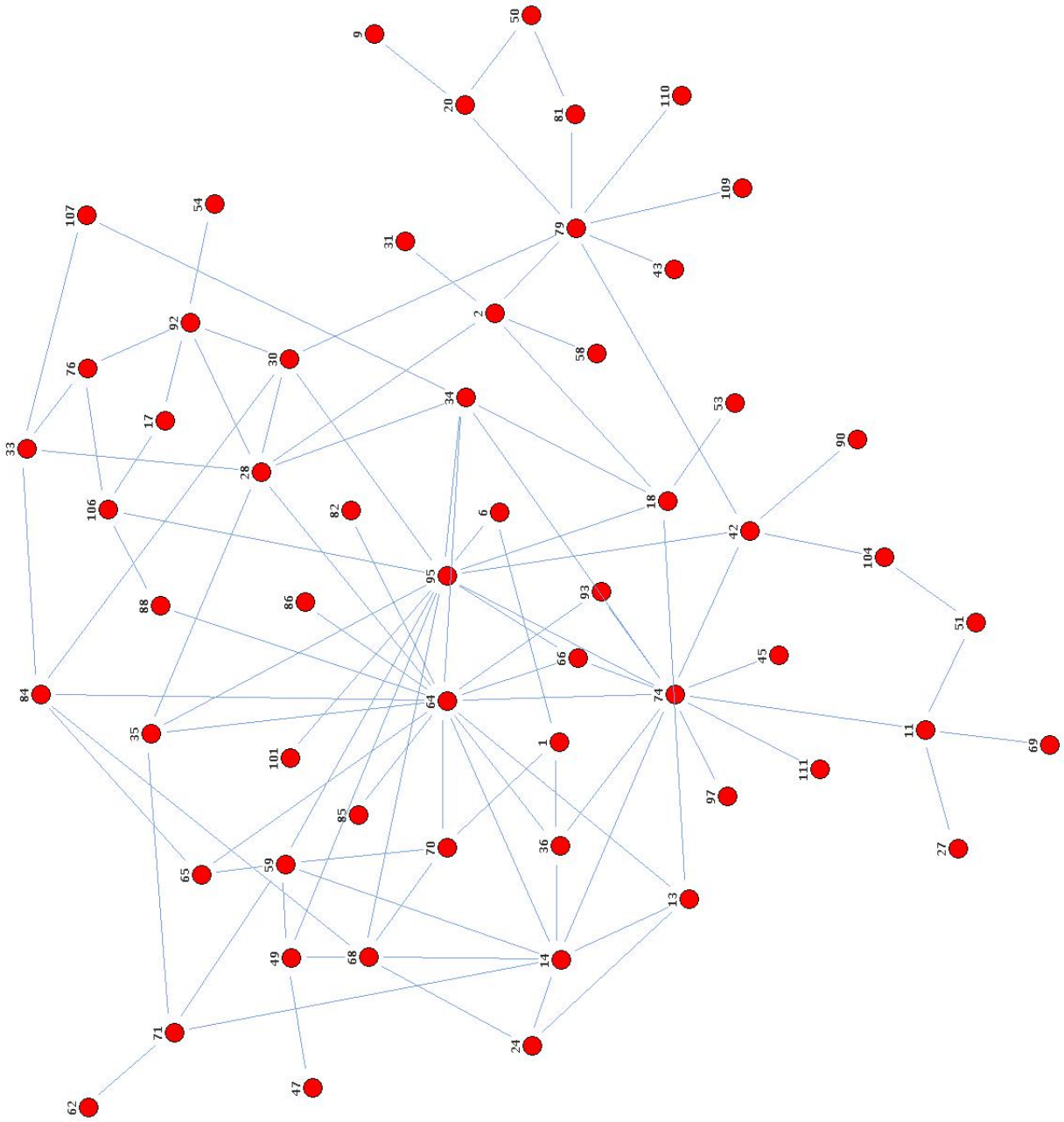


Figure A.14: Pre-determined representative graph for adjnoun

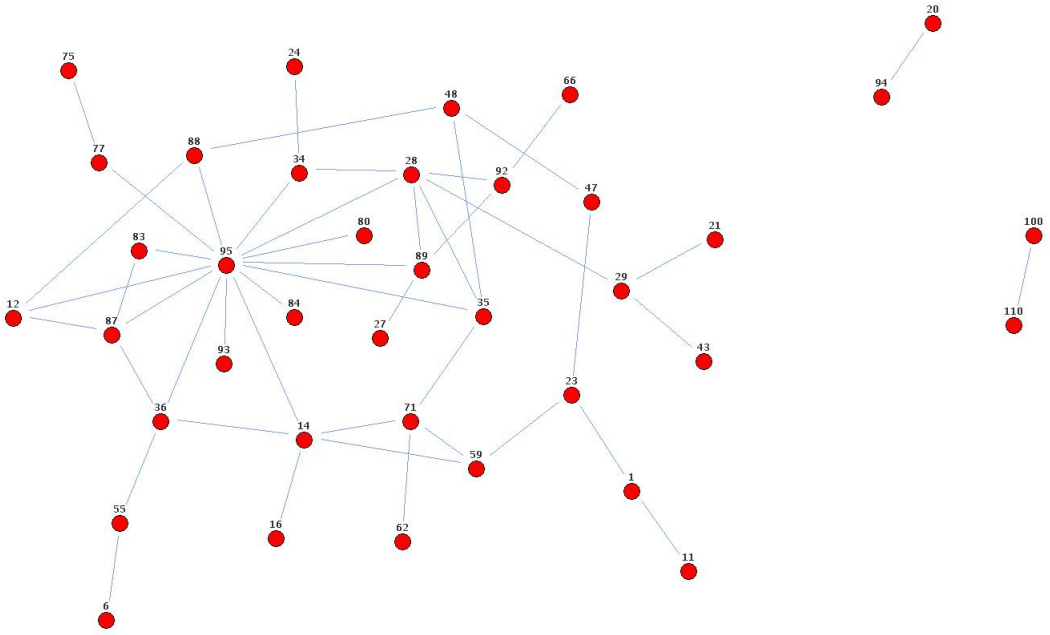


Figure A.15: Random representative graph for adjnoun

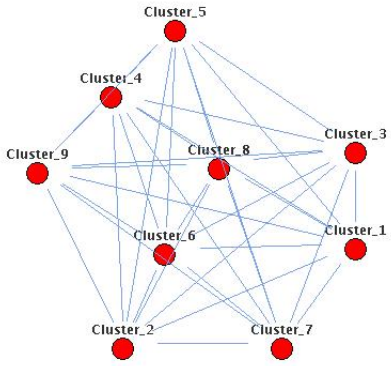


Figure A.16: Clustered graph for adjnoun

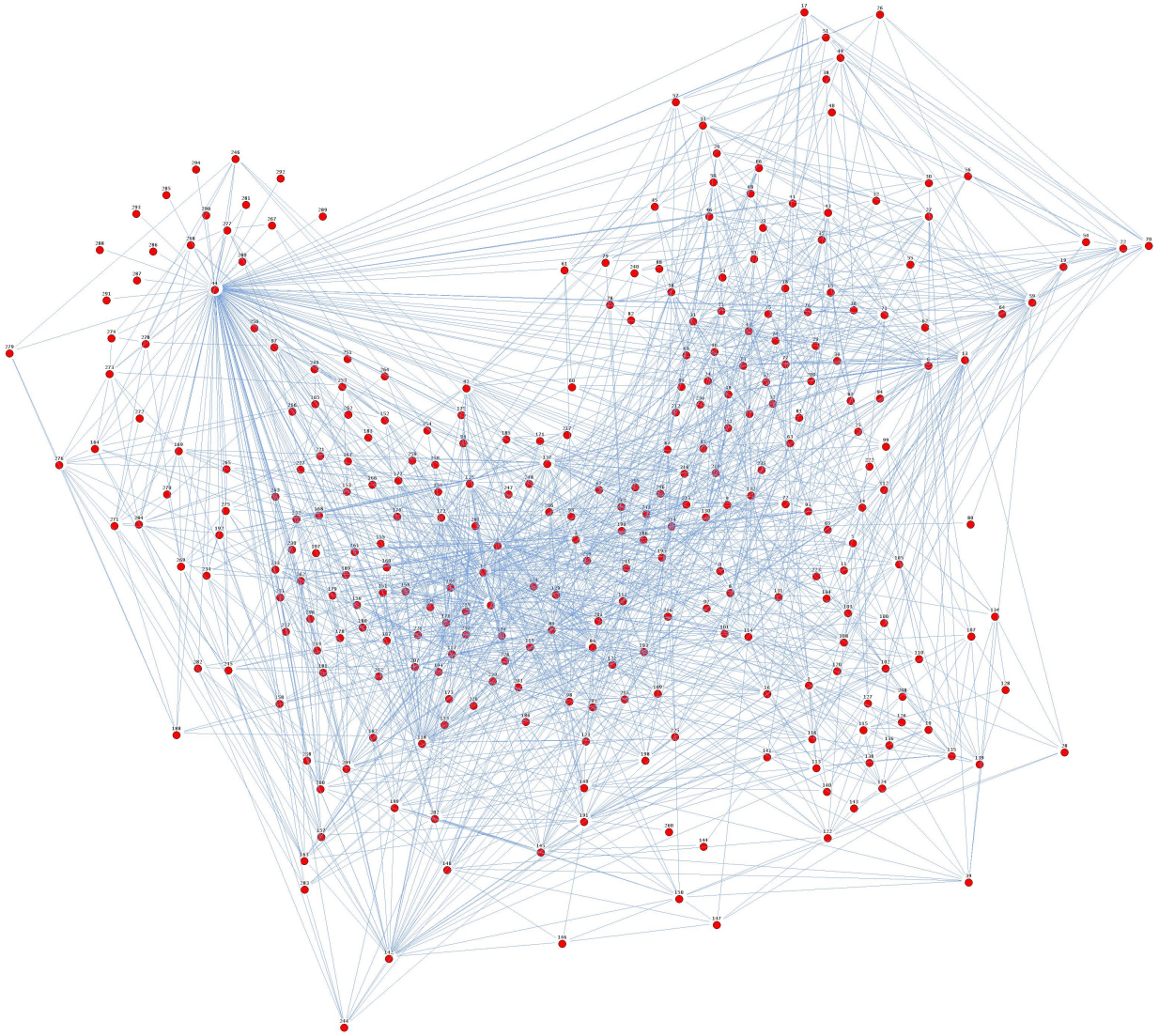


Figure A.17: Original graph for celegansneural

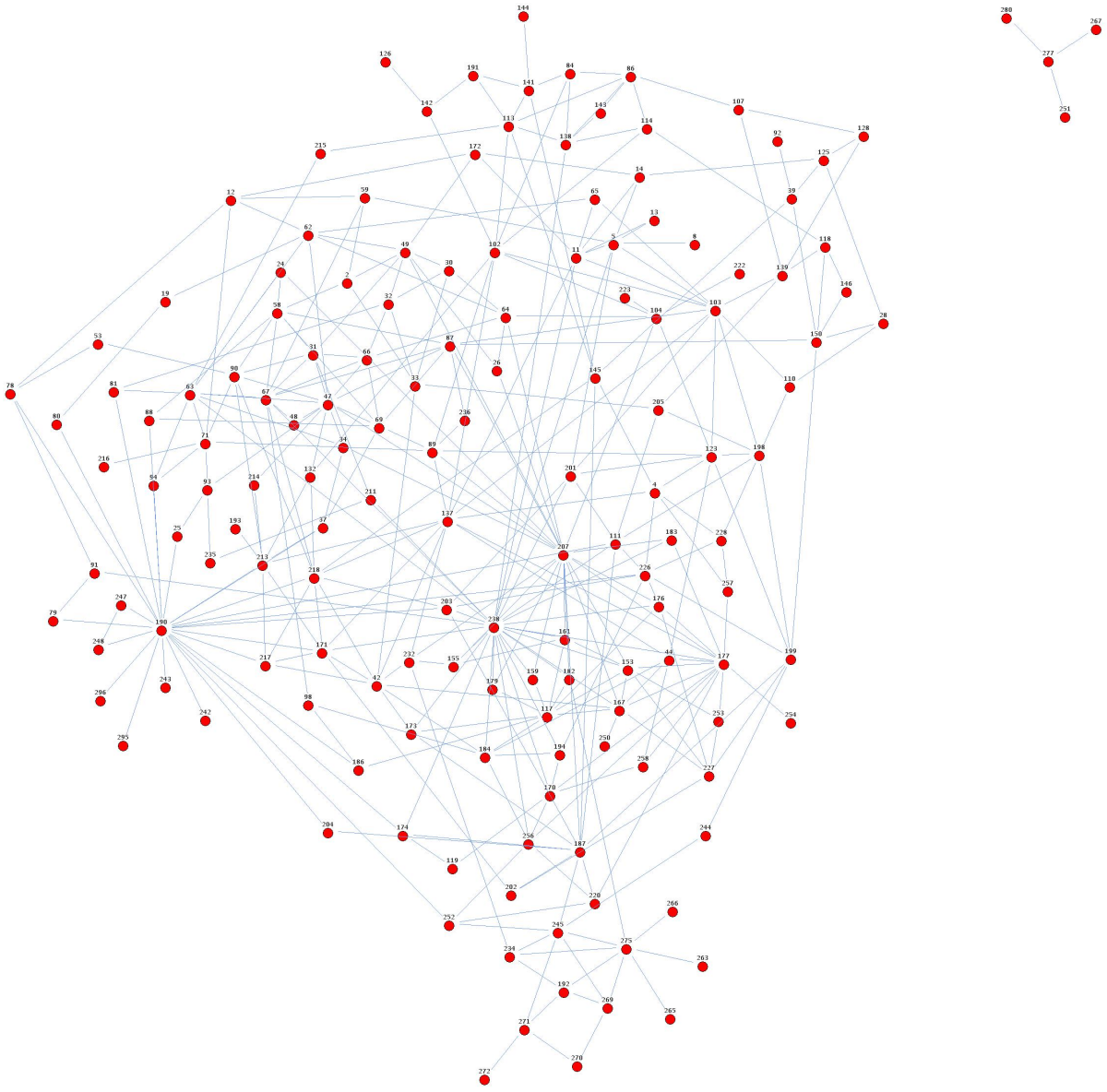


Figure A.18: Pre-determined representative graph for calegansneural

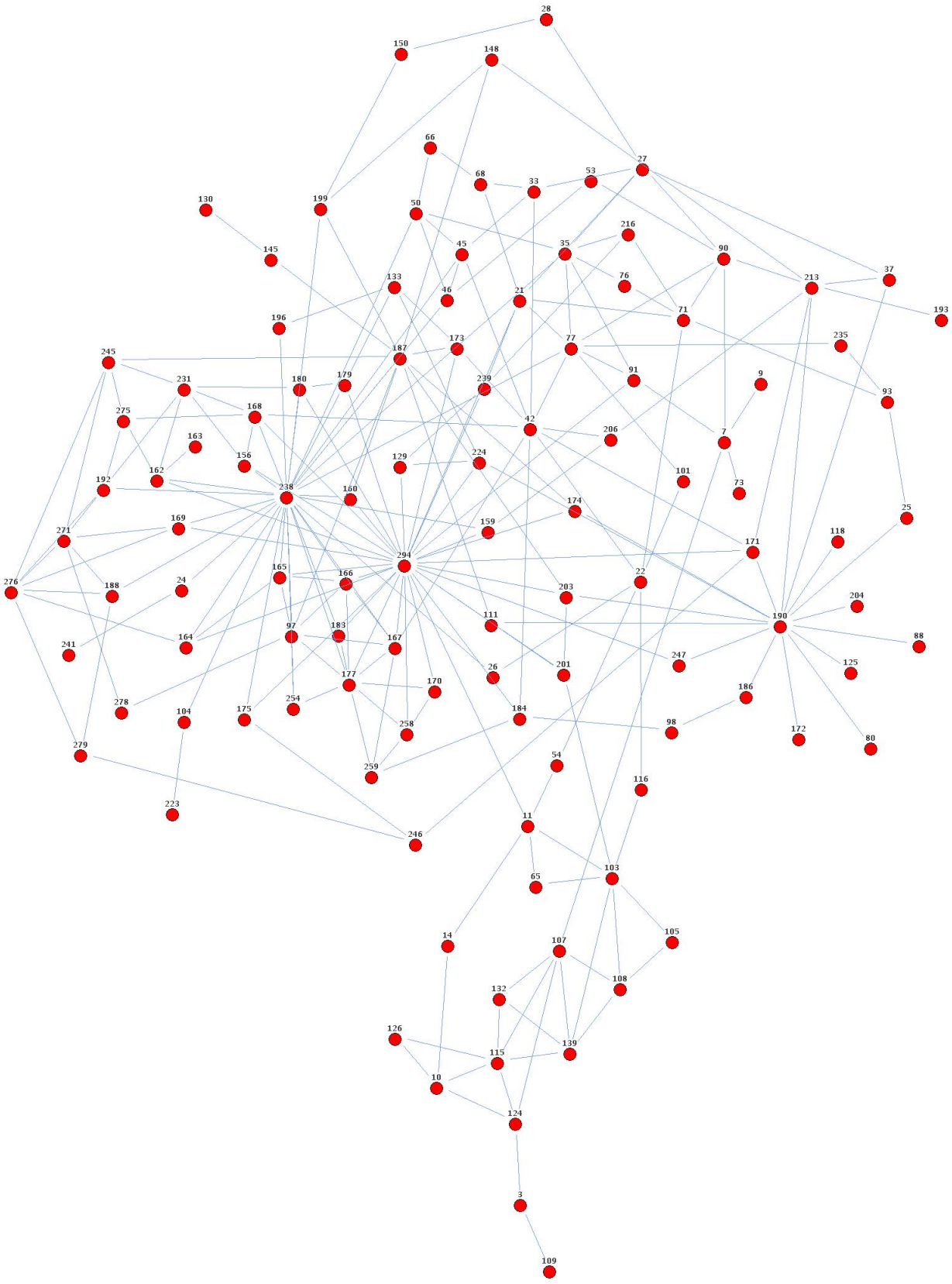


Figure A.19: Random representative graph for calegansneural

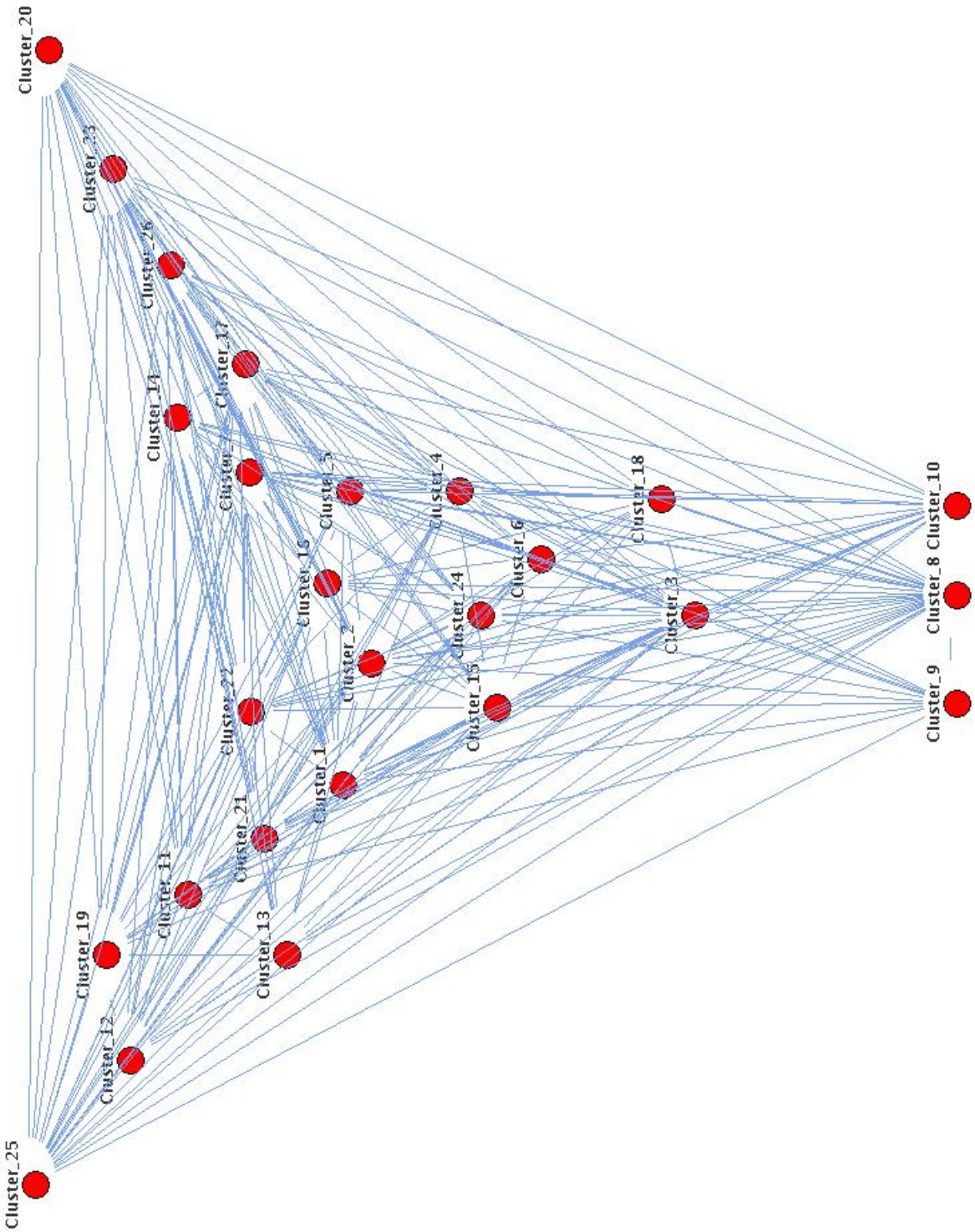


Figure A.20: Clustered graph for calegansneural

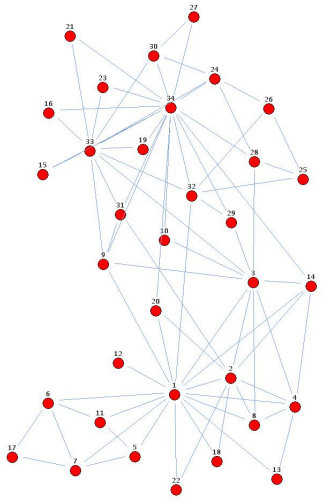


Figure A.21: Original graph for karate

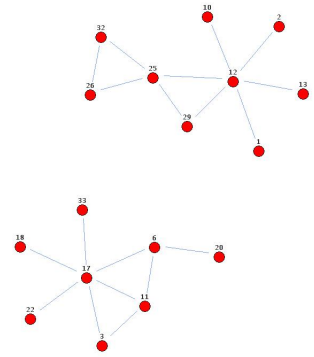


Figure A.22: Pre-determined representative graph for karate

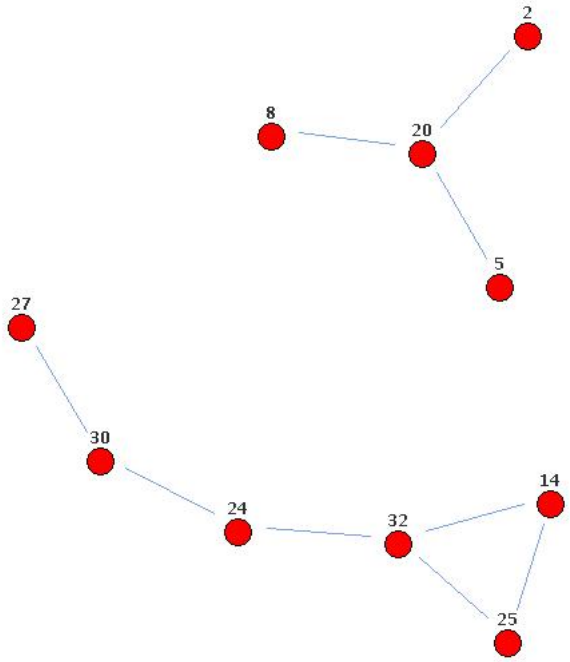


Figure A.23: Random representative graph for karate

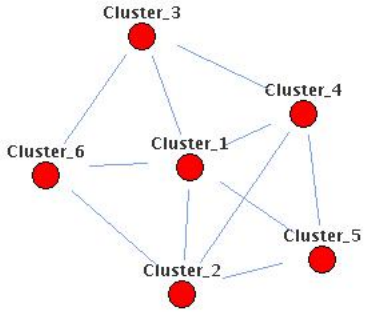


Figure A.24: Clustered graph for karate

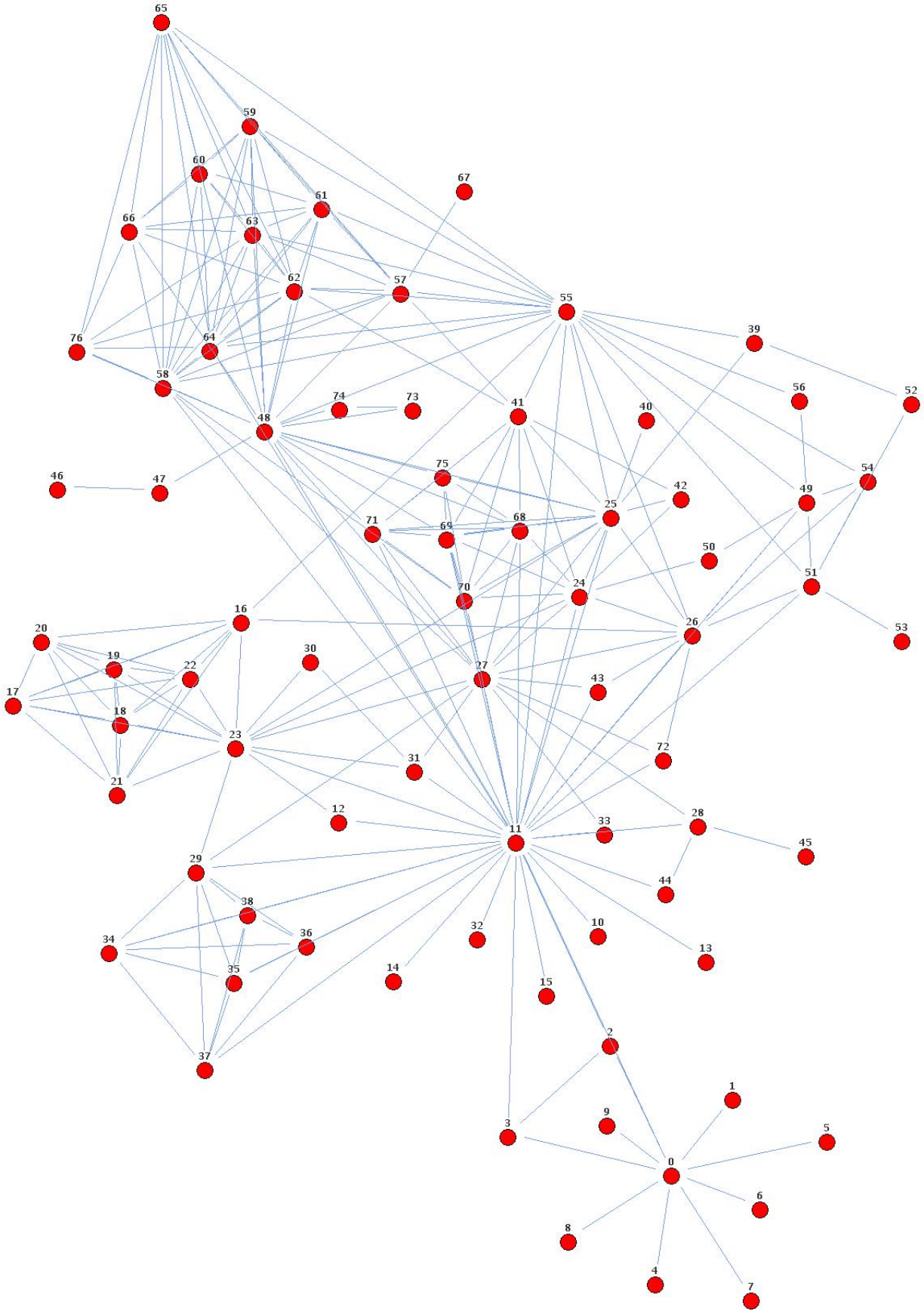


Figure A.25: Original graph for lesmis

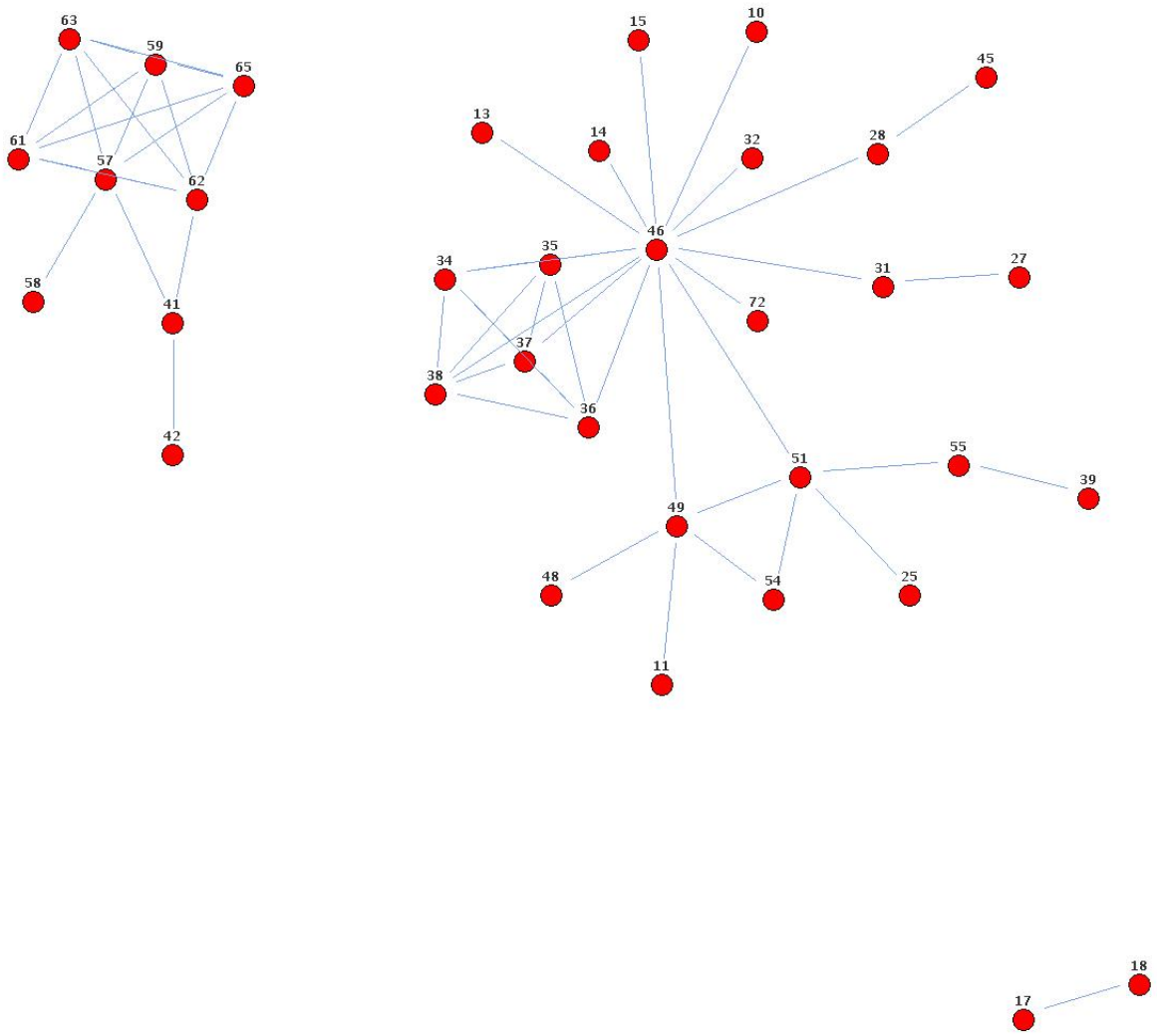


Figure A.26: Pre-determined representative graph for lesmis

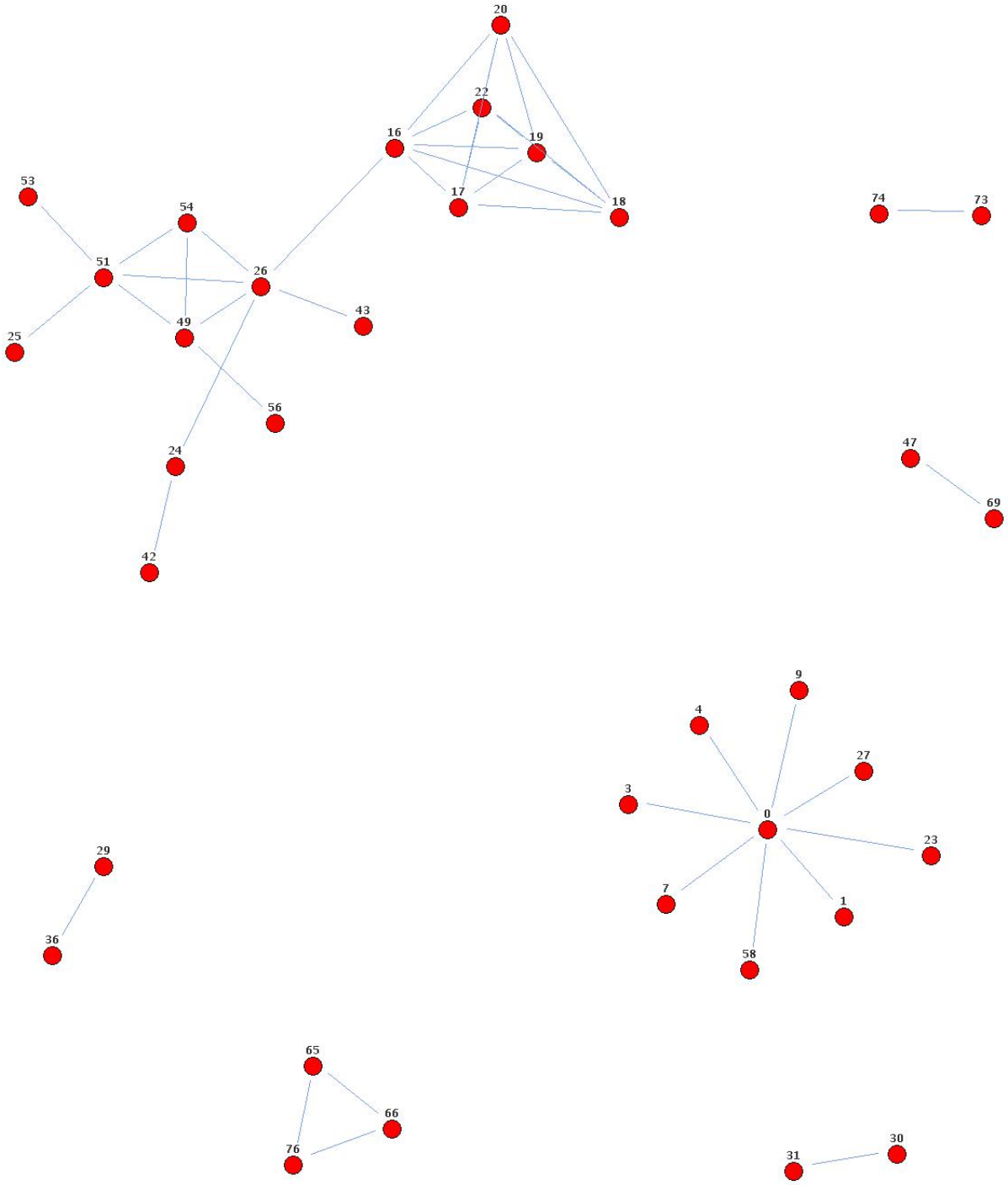


Figure A.27: Random representative graph for lesmis

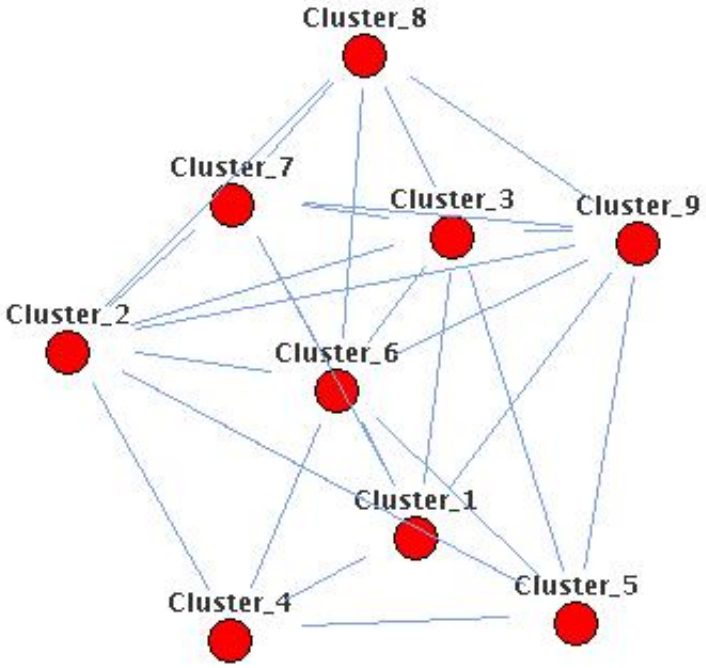


Figure A.28: Clustered graph for lesmis

VITA

Kasturi Chatterjee

June 30, 1979	Born, India
2004–present	Ph.D. candidate in Computer Science School of Computing and Information Sciences Florida International University Miami, Florida
2004–2006	M.S., Computer Science School of Computing and Information Sciences Florida International University Miami, Florida
1999–2003	B.Tech., Computer Science Institute of Engineering and Management Kalyani University Kolkata, India

PUBLICATIONS AND PRESENTATIONS

Kasturi Chatterjee, Shixia Liu and Shu-Ching Chen, (2010). *Social Network Preview using Graph Similarity.* ACM Transactions on Information Systems (under review).

Kasturi Chatterjee, S. Masoud Sadjadi, and Shu-Ching Chen, (2009). *A Distributed Multimedia Data Management over the Grid.* Multimedia Services in Intelligent Environments - Integrated Systems, Springer, 2009, in press.

Kasturi Chatterjee and Shu-Ching Chen, (2009). *HAH-tree: Towards a Multidimensional Index Structure Supporting Different Video Modelling Approaches in a Video Database Management System.* International Journal of Information and Decision Sciences (IJIDS), vol. 2, no. 2, pp. 188-207, 2010.

Kasturi Chatterjee and Shu-Ching Chen, (2009). *A Multimedia Data Management Approach with GeM-Tree.* Journal of Multimedia, in press.

Shu-Ching Chen, Min Chen, Na Zhao, Shahid Hamid, Kasturi Chatterjee, and Michael Armella, (2009). *Florida Public Hurricane Loss Model: Research in Multi-Disciplinary System Integration Assisting Government Policy Making.* Special Issue on Building the Next Generation Infrastructure for Digital Government, Government Information Quarterly, Volume 26, Issue 2, pp. 285-294, April 2009.

Yudan Li, Kasturi Chatterjee, Shu-Ching Chen, and Keqi Zhang, (2009). *A 3-D Traffic Animation System with Storm Surge Response.* accepted for publication, IEEE Inter-

national Symposium on Multimedia (ISM2009), Hyatt Regency Mission Bay Spa and Marina, San Diego, California, USA, pp. 257-262, December 14-16, 2009.

Kasturi Chatterjee and Shu-Ching Chen, (2008). *GeM-Tree: Towards a Generalized Multidimensional Index Structure Supporting Image and Video Retrieval.* accepted for publication, the Fourth IEEE International Workshop on Multimedia Information Processing and Retrieval (MIPR2008), in conjunction with IEEE International Symposium on Multimedia (ISM2008), Berkeley, California, USA, pp. 631-636, December 15-17, 2008.

Kasturi Chatterjee and Shu-Ching Chen, (2008). *Hierarchical Affinity-Hybrid Tree: A Multidimensional Index Structure to Organize Videos and Support Content-Based Retrievals.* accepted for publication, Proceedings of the 2008 IEEE International Conference on Information Reuse and Integration (IEEE IRI-08), Hilton Hotel, Las Vegas, USA, pp. 435-440, July 13-15, 2008.

Shu-Ching Chen, Min Chen, Na Zhao, Shahid Hamid, Khalid Saleem, and Kasturi Chatterjee, (2008). *Florida Public Hurricane Loss Model (FPHLM): Research Experience in System Integration.* accepted for publication, The 9th Annual International Conference on Digital Government Research, Montreal, Canada, pp. 99-106, May 18-21, 2008.

Kasturi Chatterjee, Shixia Liu, and Shu-Ching Chen, (2008). *Using Graph Similarity for Social Network Analysis.* in 6th LA Grid Summit, October 30-31, 2008, Boca Raton, Florida, USA (First Place).

Kasturi Chatterjee and Shu-Ching Chen, (2007). *A Novel Indexing and Access Mechanism using Affinity Hybrid Tree for Content-Based Image Retrieval in Multimedia Databases.* International Journal of Semantic Computing (IJSC), Vol. 1, Issue 2, pp. 147-170, June 2007.

Kasturi Chatterjee and Shu-Ching Chen, (2006). *Affinity Hybrid Tree: An Indexing Technique for Content-Based Image Retrieval in Multimedia Databases.* In proceedings of IEEE International Symposium on Multimedia (ISM2006), San Diego, CA, USA, pp. 47-54, December 11-13, 2006 (Best Paper Award).

Kasturi Chatterjee, Khalid Saleem, Na Zhao, Min Chen, Shu-Ching Chen, and Shahid Hamid, (2006). *Modeling Methodology for Component Reuse and System Integration for Hurricane Loss Projection Application.* In proceedings of IEEE International Conference on Information Reuse and Integration (IEEE IRI-2006), September 16-18, 2006, pp. 57-62, Hawaii, USA.