UNIVERSITÉ FRANÇOIS - RABELAIS DE TOURS

École Doctorale Santé, Sciences, Technologies

Laboratoire d'Informatique (EA 2101)
Équipe Reconnaissance des Formes et Analyse d'Images

Thèse en cotutelle entre l'Université François - Rabelais de Tours, France et l'Universitat Autònoma de Barcelona, Espagne.

THÈSE présenté par :

Muhammad Muzzamil LUQMAN

soutenue le : 02 mars 2012

pour obtenir le grade de : Docteur de l'université François - Rabelais

Discipline/ Spécialité : Informatique

Fuzzy Multilevel Graph Embedding for Recognition, Indexing and Retrieval of Graphic Document Images

DIRECTEURS DE THÈSE

RAMEL Jean-Yves	Professeur, Université François - Rabelais de Tours, France.
LLADOS Josep	Professeur, Université Autonoma de Barcelone, Espagne.

CO-ENCADRANT

BROUARD Thierry	Maître de conférences, Université François - Rabelais de Tours.

RAPPORTEURS

BUNKE Horst	Professeur émérite, Université de Bern, Suisse.
OGIER Jean-Marc	Professeur, Université de La Rochelle, France.

JURY DE THÈSE

BROUARD Thierry	Maître de conférences, Université François - Rabelais de Tours.
BUNKE Horst	Professeur émérite, Université de Bern, Suisse.
LLADOS Josep	Professeur, Université Autonoma de Barcelone, Espagne.
RAMEL Jean-Yves	Professeur, Université François - Rabelais de Tours, France.
TABBONE Salvatore-Antoine	Professeur, Université de Lorraine, France.
VALVENY Ernest	Maître de conférences, Université Autonoma de Barcelone.

Cotutelle thesis between **Universitat Autònoma de Barcelona, Spain** and **Université François - Rabelais de Tours, France**.

Fuzzy Multilevel Graph Embedding for Recognition, Indexing and Retrieval of Graphic Document Images

A dissertation submitted by **Muhammad Muzzamil Luqman** at Universitat Autònoma de Barcelona to fulfil the degree of **Doctor of Philosophy**.

Bellaterra, 2012.

Directors: **Dr. Josep Llados**
Professor
Autonoma University of Barcelona, Spain

Dr. Jean-Yves Ramel
Professor
François - Rabelais University of Tours, France

Co-director: **Dr. Thierry Brouard**
Assistant Professor
François - Rabelais University of Tours, France

This document was typeset by the author using LaTeX 2_ε.

The research described in this book was carried out at the Computer Vision Center, Universitat Autònoma de Barcelona and Computer Science Laboratory, Université François - Rabelais de Tours.

I am thankful to **Higher Education Commission**[1], **Government of Pakistan**, for award of scholarship under grant PD-2007-1/Overseas/FR/HEC/222, to pursue my master's and doctorate.

[1] `http://www.hec.gov.pk/`

Acknowledgments

Alhamdulilah.

I would like to present my foremost sincere gratitude to Professor Jean-Yves Ramel and Professor Josep Llados. I passed a long journey under their kind supervision. All credit for the successful completion of this thesis goes to their patient attitude, openness to new ideas and the freedom that they provided me to carry out the research work at my pace. Thank you very much Jean-Yves and Josep.

I am thankful to Dr. Thierry Brouard for his guidance during initial phase of the thesis research and for his support during all important phases of the thesis work.

I am thankful to Professor Horste Bunke and Professor Jean-Marc Ogier for carefully reading my dissertation and providing me very useful feedback.

I would also like to acknowledge the support of the administration staff of Computer Science Laboratory of Tours and Computer Vision Center Barcelona for their assistance and support during the course of my thesis research.

Thank you very much all my colleagues at Computer Science Laboratory of Tours and Computer Vision Center Barcelona for your support and the wonderful years that I passed among you.

Tons of thanks to my parents, brothers and sisters. Your love and moral support actually made me successfully complete this important phase of my life.

Bundles of thanks to all my friends. I don't want to miss any one and I will not cite any names. But all of you are very dear to me and all of you helped me at various stages to keep my moral high. You made me go through the crucial and difficult stages of my thesis.

Abstract

Structural pattern recognition approaches offer the most expressive, convenient and powerful but computational expensive representations of underlying relational information. These representations can benefit from mature, less expensive and efficient state-of-the-art machine learning models of statistical pattern recognition, only after being mapped to a low-dimensional vector space.

This thesis addresses the problem of lack of efficient computational tools for graph based structural pattern recognition approaches and proposes to exploit computational strength of statistical pattern recognition. The contribution of this thesis is two-fold.

The first contribution of this thesis is a new method of explicit graph embedding. The proposed graph embedding method exploits multilevel analysis of graph for extracting graph level information, structural level information and elementary level information from graphs. It embeds this information into a numeric feature vector. The method employs fuzzy overlapping trapezoidal intervals for addressing the noise sensitivity of graph representations and for minimizing the information loss while mapping from continuous graph space to discrete vector space. The method has unsupervised learning abilities and is capable of automatically adapting its parameters to underlying graph dataset.

The second contribution of this thesis is a framework for automatic indexing of graph repositories for graph retrieval and subgraph spotting. This framework exploits explicit graph embedding together with classification and clustering tools. It achieves the automatic indexing of a graph repository during its off-line learning phase, where its extracts the cliques of order 2 from graphs and embeds them into feature vectors by employing the aforementioned explicit graph embedding technique. It clusters the feature vectors into classes, learns a classifier and builds an index for the graph repository. During on-line spotting phase, it extracts the cliques of order 2 from query graph, embeds them into feature vectors and uses the index of the graph repository to retrieve the graphs from repository. The framework does not require a labeled learning set and can be easily deployed to a range of application domains, offering ease of query by example (QBE) and granularity of focused retrieval.

Experimentation on latest public graph datasets from International Association of Pattern

Recognition's Technical Committee on graph-based representations (TC-15) evaluates the power and applicability of our graph embedding framework for the problems of graph classification and graph clustering. A second set of experimentation evaluates the framework for automatic indexing of graph repositories for graph retrieval and subgraph spotting.

Application to the real problems of recognition, indexing and retrieval of graphic document images is also presented.

Keywords : Pattern recognition, graph clustering, graph classification, graph embedding, subgraph spotting, fuzzy logic, graphics recognition.

Résumé

Cette thése aborde le probléme du manque de performance des outils exploitant des représentations á base de graphes en reconnaissance des formes. Nous proposons de contribuer aux nouvelles méthodes proposant de tirer partie, á la fois, de la richesse des méthodes structurelles et de la rapidité des méthodes de reconnaissance de formes statistiques.

Deux principales contributions sont présentées dans ce manuscrit.

La premiére correspond á la proposition d'une nouvelle méthode de projection explicite de graphes procédant par analyse multi-facettes des graphes. Cette méthode effectue une caractérisation des graphes suivant différents niveaux qui correspondent, selon nous, aux point-clés des représentations á base de graphes. Il s'agit de capturer l'information portée par un graphe au niveau global, au niveau structure et au niveau local ou élémentaire. Ces informations capturées sont encapsulées dans un vecteur de caractéristiques numériques employant des histogrammes flous. L'approche floue á base d'intervalles trapézodaux, permet de mieux appréhender la sensibilité aux bruits des représentations á base de graphes et de réduire au minimum la déformation de l'information liée au positionnement arbitraire des frontièrs lors du passage d'un espace continu á un espace vectoriel discret choisi et utilisé pour décrire certaines caractéristiques des graphes. La méthode proposée utilise, de plus, un mécanisme d'apprentissage non supervisée pour adapter automatiquement ses paramètres en fonction de la base de graphes á traiter sans nécessiter de phase d'apprentissage préalable.

La deuxième contribution correspond á la mise en place d'une architecture pour l'indexation de masses de graphes afin de permettre, par la suite, la recherche de sous-graphes présents dans cette base. Cette architecture utilise la méthode précédente de projection explicite de graphes appliquée sur toutes les cliques d'ordre 2 pouvant être extraites des graphes présents dans la base á indexer afin de pouvoir les classifier. Un partitionnement des cliques permet de constituer l'index qui sert de base á la description des graphes et donc á leur indexation en ne nécessitant aucune base d'apprentissage pré-étiquetées. Cela procure á cette méthode une généricité très forte et donc un déploiement facile. Lors de la phase de recherche d'un sous-graphe dans la base pré-indexée, la réponse á la requête est formée de l'ensemble des graphes contenant toutes les cliques d'ordre 2 obtenues par décomposition

de la requête. Enfin, le sous-graphe associé á la requête est repéré dans chacun des graphes identifiés. La méthode proposée est applicable á de nombreux domaines, apportant la souplesse d'un système de requête par l'exemple et la granularité des techniques d'extraction ciblée (focused retrieval).

Des expérimentations sur les bases de graphes publiques récentes proposées par le TC15 de l'IAPR (groupe de rechercher sur les méthodes á base de graphes) sont présentées et permettent d'évaluer les performances et l'applicabilité des propositions faites dans cette thèse sur des problèmes de classification et de partitionnement de graphes. Un deuxième jeu d'expérimentations présente les résultats obtenus par notre méthode d'indexation automatique de bases de graphes pour la recherche de sous-graphes.

Des exemples d'applications des approches proposées á des problèmes réels de reconnaissance, d'indexation et de repérage d'images de documents graphiques sont aussi présentés tout au long du manuscrit.

Mots clés : Reconnaissance des formes, partitionnement de graphes, classification de graphes, projection de graphes, repérage de sous-graphes, logique floue, reconnaissance de graphiques.

Contents

List of Tables

List of Figures

Introduction

ABILITY to recognize patterns is among the most crucial capabilities of human beings for their survival, which enables them to employ their sophisticated neural and cognitive systems [Duda et al., 2000], for processing complex audio, visual, smell, touch and taste signals. Man is the most complex and the best existing system of pattern recognition. Without any explicit thinking, we continuously compare, classify and identify huge amount of signally data everyday [Kuncheva, 2004], start from the time we get up in the morning till the last second we fall asleep. This includes recognizing the face of a friend in a crowd, a spoken word embedded in noise, the proper key to lock the door, smell of coffee, the voice of a favorite singer, the recognition of alphabetic characters and millions of more tasks that we perform on regular basis.

The scientific domains of artificial intelligence (AI) and pattern recognition (PR) can be seen as the transportation of the human capability of analyzing - to compare, to classify and to identify - the audio and visual signals, to computers. So that computers may assist humans for pattern recognition tasks and to replace humans for some of them.

Pattern recognition has emerged as an important research domain and has supported the development of numerous applications in many different areas of activity. Robot assisted manufacture, medical diagnostic systems, forecast of economic variables, exploration of earth resources, analysis of satellite data, face detection, verification and recognition, object detection and recognition, handwritten digit and character recognition, speech and speaker verification and recognition, information and image retrieval, text detection and categorization, gender classification and prediction ([Byun, 2003], [De Sa, 2001] and [Friedman and Kandel, 1999]), being some of the important to mention.

The problems of pattern recognition are often very complex and it is nearly impossible to write an explicitly programmed solution for them. For example it is impossible to write an analytical program for recognizing a face in a photo [Shawe-Taylor and Cristianini, 2004]. The pattern recognition research community has overcome this problem by adapting a learning methodology which is highly inspired by human ability to learn. A learning methodology refers to the approach where instead of precisely defining a set of specifications for solving a problem analytically, the machine is trained on data and it learns the concept of a class by discriminating between groups of similar objects. Based on the

inferred rules and learning performed during training, the machine is able to make predictions about new and unseen data. More formally, the machine acquires generalization power through learning [Riesen, 2010].

A pattern recognition task for computers can be looked upon as consisting of two main steps - (i) the representation of the signal data by a data structure and (ii) the computation of desired operation (pattern recognition). The two important sub domains of pattern recognition - the structural pattern recognition and the statistical pattern recognition - each have its strength only in one of the two aforementioned steps, respectively.

The structural pattern recognition offers the most powerful relational data structure of graph. For the last three decades, graphs have been used for pattern recognition and image analysis, for extracting and representing complex relations in underlying data. However, there is still a lack of efficient computational tools and learning models which can process this data structure.

On the other side, the statistical pattern recognition offers highly efficient computational models of machine learning, classification and clustering, by employing the well established theory of statistics. But these computational models can work only on simple numeric vectors and can not process complex high dimensional relational data structures.

Over decades of parallel research in both of these sub domains of pattern recognition - structural and statistical pattern recognition - powerful representations and efficient computational models have been built. But little progress has been made towards the long desired objective (of research in pattern recognition), to join the advantages of the structural and statistical pattern recognition approaches for building more powerful and efficient algorithms.

This thesis is a step forward to achieve this objective of joining the advantages of structural and statistical pattern recognition approaches. We propose an algorithm which permits the pattern recognition applications to employ the powerful relational data structure of attributed graphs along with the computational strengths of state of the art statistical models of machine learning, classification and clustering.

The contribution of this thesis is two-fold.

The first contribution of this thesis is a new method of explicit graph embedding. The proposed graph embedding method exploits multilevel analysis of graph for extracting graph level information, structural level information and elementary level information from graphs. It embeds this information into a numeric feature vector. The method employs fuzzy overlapping trapezoidal intervals for addressing the noise sensitivity of graph representations and for minimizing the information loss while mapping from continuous graph space to discrete vector space. The method has unsupervised learning abilities and is capable of automatically adapting its parameters to underlying graph dataset.

The second contribution of this thesis is a framework for automatic indexing of graph repositories for graph retrieval and subgraph spotting. This framework exploits explicit graph embedding together with classification and clustering tools. It achieves the automatic indexing of a graph repository during its off-line learning phase, where its extracts the cliques of order 2 from graphs and embeds them into feature vectors by employing the aforementioned explicit graph embedding technique. It clusters the feature vectors into classes, learns a classifier and builds an index for the graph repository. During on-line spotting phase, it extracts the cliques of order 2 from query graph, embeds them into feature vectors and uses the index of the graph repository to retrieve the graphs from repository. The framework does not require a labeled learning set and can be easily deployed to a range of application domains, offering ease of query by example (QBE) and granularity of focused retrieval.

The dissertation is organized in six chapters and two appendices. A brief introduction to the contents of each chapter is given below:

In chapter 1 we present important definitions and concepts and we formalize the notations that have been used in this dissertation.

In chapter 2 we present a literature review on state of the art of structural pattern recognition, statistical pattern recognition, graph representation of images, exact graph matching, graph isomorphism, subgraph isomorphism, error tolerant graph matching, distance between graphs, graph embedding, graph classification and fuzzy logic. We summarize the important contributions of our work in light of the limitations of existing methods.

In chapter 3 we outline our explicit graph embedding method i.e. the Fuzzy Multilevel Graph Embedding (FMGE). We first present an overall global description of FMGE and then present details on the FMGE framework. We conclude this chapter by introducing the application of FMGE to graph classification, graph clustering and graph retrieval.

In chapter 4 we present a framework for automatic indexing of attributed graph repositories. We demonstrate a practical application of Fuzzy Multilevel Graph Embedding (FMGE) together with classification and clustering tools, for achieving graph retrieval and subgraph spotting.

In chapter 5 we present the experimental evaluations of FMGE for the problems of graph clustering and graph classification. We present the experimental evaluations of the framework for automatic indexing of graph repositories for graph retrieval and subgraph spotting, with an application to content spotting in graphic document image repositories. We provide experimental results for the application of the thesis work to the real problems of recognition, indexing and retrieval of graphic document images.

In chapter 6 we present a discussion on the presented work. We highlight the improvements of the proposed method which should be further studied and conclude this dissertation. We point out the possible lines of future research.

In appendix A we provide details on the graph databases used in this thesis.

In appendix B we provide details on the graph repository extracted from electronic diagrams and architectural floor plans document images.

Chapter 1

Definitions and notations

In this chapter we present important definitions and concepts and we formalize the notations that have been used in this dissertation.

1.1 Introduction

This chapter presents important definitions and concepts adapted to the way we have used them and to the vocabulary of this dissertation. We first define the terms which are used throughout the dissertation. These include graph, subgraph, clique and attributed graph. This is followed by a section on defining important features of graphs and a section introducing important concepts on representation and processing of graphs. Along with presenting the definitions, in the meantime we also formalize the notations for facilitating the understanding of the work presented in upcoming chapters of the dissertation.

1.2 Terminology on graphs

Definition 1: Graph

Let V denote the set of vertices and E denote the set of edges. A graph G is a set of vertices (V) connected by edges (E). It is given by the ordered pair:

$$G = (V(G), E(G))$$

where,

$V(G)$ is the set of vertices in graph G and

$E(G)$ is a set of 2 element subsets of V(G), given by:
$E \subseteq \{\{u, v\} : u, v \in V(G)\}$

For **a directed graph**, the ordered pair (u, v) represents an edge from vertex u to vertex v. Where as, for **an undirected graph** it represents an edge between vertex u and vertex v without signifying any direction.

Definition 2: Subgraph

A graph $subG = (V(subG), E(subG))$ is called a subgraph of a graph $G = (V(G), E(G))$ if it contains no vertices or edges that are not in graph G.

Mathematically,

$V\,(subG) \subseteq V\,(G)$ and

$E\,(subG) \subseteq E\,(G)$

Figure 1.1 shows a subgraph in a graph.

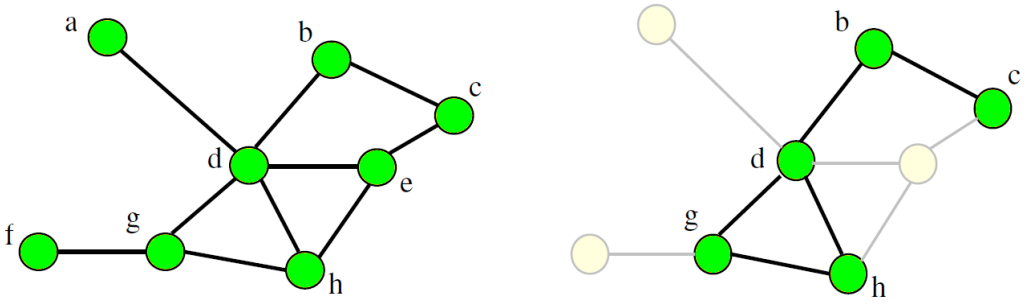


Figure 1.1: Example of a graph (left) and a subgraph (right).

Definition 3: Clique

A clique in an undirected graph G is a subset of its vertices $V(G)$ such that each pair of vertices in the subset is connected by an edge. Figure 1.2 shows a clique (represented by $\{d, e, h\}$) in a graph.

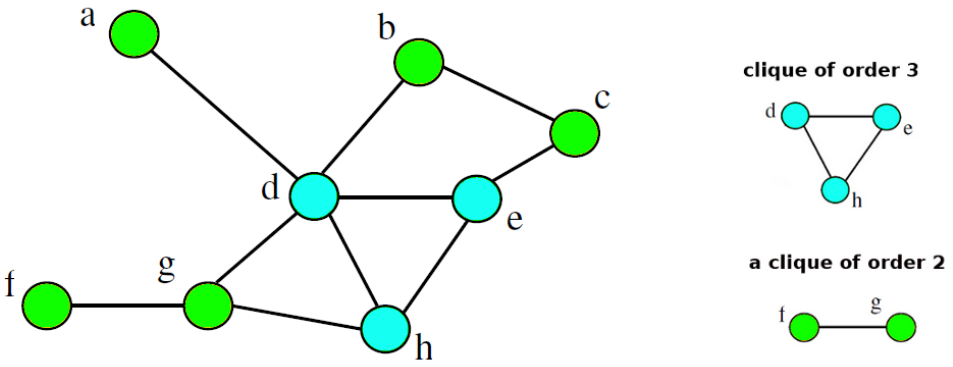


Figure 1.2: Example of a clique in a graph.

A **maximum clique** is a clique of the largest possible size in a given graph. The **clique number** of a graph is the number of vertices in its maximum clique.

Definition 4: Attributed Graph (AG)

Let A_V and A_E denote the domains of possible values for attributed vertices and edges respectively. These domains are assumed to include a special value that represents a null value of a vertex or an edge. An attributed graph AG over (A_V, A_E) is defined to be a four-tuple:

$$AG = (V, E, \mu^V, \mu^E)$$

where,

V is a set of vertices,

$E \subseteq V \times V$ is a set of edges,

$\mu^V : V \longrightarrow A_V^k$ is function assigning k attributes to vertices and

$\mu^E : E \longrightarrow A_E^l$ is a function assigning l attributes to edges.

In this dissertation we use the term attributed graph to refer to an undirected attributed graph, unless otherwise explicitly specified.

1.3 Important features of graphs

We present the definitions in this section in terms of attributed graphs, as we have used them in this dissertation. However, these definitions are equally applicable to graphs in general.

Definition 5: Graph order

The order of an attributed graph $AG = (V, E, \mu^V, \mu^E)$ is given by $|V|$ i.e. the number of vertices in AG.

Let AG_1 and AG_2 be two attributed graphs, then:

AG_1 is smaller than AG_2 iff $|V_1| < |V_2|$

AG_1 and AG_2 are equal ordered iff $|V_1| = |V_2|$

AG_1 is larger than AG_2 iff $|V_1| > |V_2|$

Definition 6: Graph Size

The size of an attributed graph $AG = (V, E, \mu^V, \mu^E)$ is given by $|E|$ i.e. the number of edges in AG.

Let AG_1 and AG_2 be two attributed graphs, then:

AG_1 is thinner than AG_2 iff $|E_1| < |E_2|$

AG_1 and AG_2 are equal sized iff $|E_1| = |E_2|$

AG_1 is thicker than AG_2 iff $|E_1| > |E_2|$

Definition 7: Node Degree

The degree of a (vertex or) node V_i in graph $AG = (V, E, \mu^V, \mu^E)$ refers to the number of edges connected to V_i.

If AG is a directed graph then each of its nodes has an in-degree and an out-degree associated to it. The in-degree refers to the number of incoming edges and out-degree refers to the number of outgoing edges for a node.

Generally, the terms *densely connected graph* and *sparsely connected graph* are used for abstractly categorizing a graph on the basis of its node degrees.

1.4 Representation and processing of graphs

The most simple and generic data structure used for representation of graphs is a set of nodes and a set of edges (each linking a pair of nodes together). However because of its ease of manipulation and computation, the widely used representation of a graph is by an adjacency matrix. An adjacency matrix of a graph is a square matrix having size of the graph order. The coefficients of the matrix are boolean values - representing the existence of an edge between the two corresponding nodes (represented by indexes of the square matrix) in the graph.

Generally, lists are used for storing the attributes of nodes and edges of graph.

Definition 8: Adjacency matrix of a graph

Let $G = (V(G), E(G))$ be a graph of order $n = |V(G)|$. The adjacency matrix A of graph G has the size of $n \times n$. The coefficients of the matrix are given as:

$$A_{ij} = \begin{vmatrix} 1 & \text{an edge exists between vertex } i \text{ and vertex } j \\ 0 & \text{otherwise} \end{vmatrix}$$

The adjacency matrix of an undirected graph is a symmetric matrix.

Definition 8: Laplacian matrix of a graph

Let $G = (V(G), E(G))$ be a graph of order $n = |V(G)|$. The Laplacian matrix L of graph G has the size of $n \times n$. The coefficients of the matrix are given as:

$$A_{ij} = \begin{vmatrix} |v_i| & \text{if i = j} \\ -1 & \text{if i} \neq \text{ j and an edge exists between vertex i and j} \\ 0 & \text{otherwise} \end{vmatrix}$$

Definition 9: Exact graph matching and graph isomorphism

Let $G_1 = (V(G_1), E(G_1))$ and $G_2 = (V(G_2), E(G_2))$ be two graphs. Exact graph matching problem is to find a one-to-one mapping:

$$f : V(G_2) \longrightarrow V(G_1)$$

such that:

$$(u, v) \in E(G_2) \text{ iff } (f(u), f(v)) \in E(G_1)$$

If such a mapping f exists, G_1 is called isomorphic to G_2 and the phenomenon is called graph isomorphism.

Figure 1.3 shows an example of graph isomorphism.

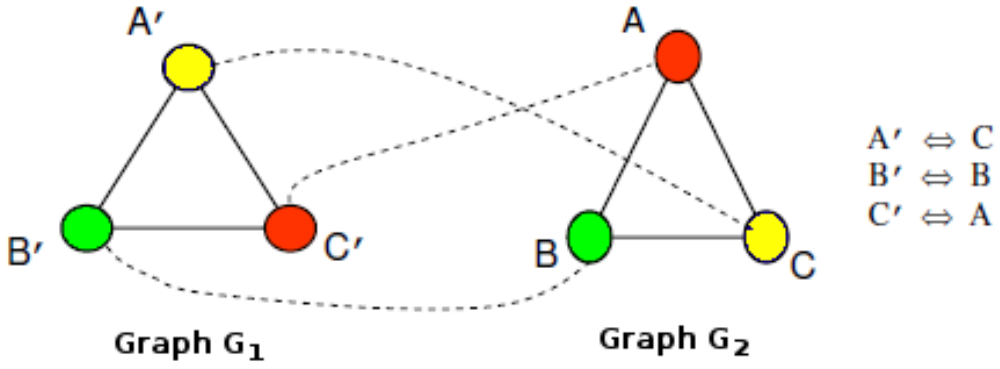


Figure 1.3: Example of graph isomorphism.

Graph isomorphism is reflexive, symmetric and transitive.

Definition 10: Subgraph isomorphism

Let $G_1 = (V(G_1), E(G_1))$ and $G_2 = (V(G_2), E(G_2))$ be two graphs, such that:

$V(G_2) \subset V(G_1)$ and

$E(G_2) \subset E(G_1)$

Subgraph isomorphism refers to the problem to find a mapping between G_1 and G_2, as given by:

$$f' : V(G_2) \longrightarrow V(G_1)$$

such that:

$$(u, v) \in E(G_2) \text{ iff } (f'(u), f'(v)) \in E(G_1)$$

If such a mapping f' exists, there exists a subgraph isomorphism from G_1 to G_2.

Figure 1.4 shows an example of subgraph isomorphism.

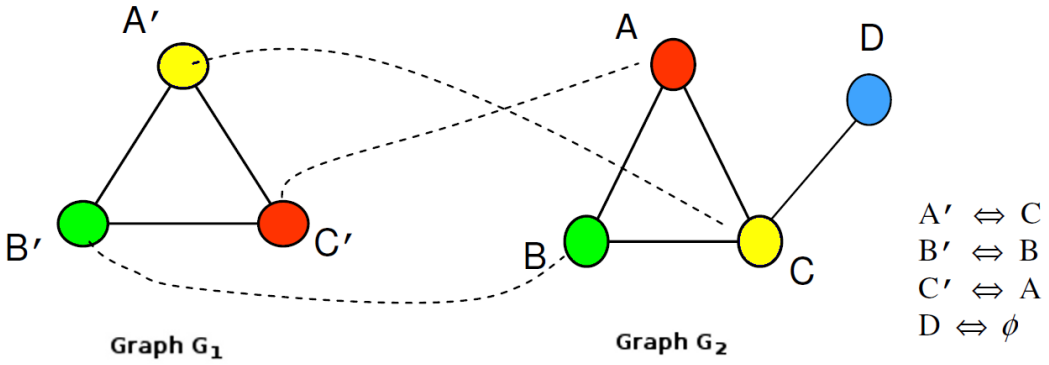


Figure 1.4: Example of subgraph isomorphism.

Definition 11: Maximum common subgraph (mcs)

Let $G_1 = (V(G_1), E(G_1))$ and $G_2 = (V(G_2), E(G_2))$ be two graphs.

A graph $G = (V(G), E(G))$ is said to be a common subgraph of G_1 and G_2 if there exist subgraph isomorphism from G to G_1 and from G to G_2. The largest common subgraph (w.r.t. graph order i.e. $|V(G)|$) is called the maximum common subgraph of G_1 and G_2.

Definition 12: Median graph

A median graph of a collection of graphs C is a graph m that minimizes the sum of distances to all other graphs in this collection. Mathematically,

$$m = \underset{g_1 \in C}{\operatorname{argmin}} \sum_{g_2 \in C} d\{g_1, g_2\}$$

where,

$d\{g_1, g_2\}$ is represents the distance between graph g_1 and graph g_2.

A commonly used distance measure in graph domain is the graph edit distance.

Definition 13: Graph edit distance (GED)

Let $g_1 = (V(g_1), E(g_1))$ and $g_2 = (V(g_2), E(g_2))$ be two graphs. The graph edit distance between g_1 and g_2 is defined as:

$$d(g_1, g_2) = \min_{(e_1, \ldots, e_k) \in \gamma(g_1, g_2)} \sum_{i=1}^{k} c(e_i)$$

where,

$\gamma(g_1, g_2)$ denotes the set of edit paths transforming g_1 into g_2 and

c denotes the cost function measuring the strength $c(e_i)$ of edit operation e_i.
The edit operations include e.g. addition of node/edge and deletion of node/edge.

We will describe graph edit distance in further details, while reviewing the state of the art in Chapter 2 (Section 2.3.3.1).

Definition 14: Graph Embedding (GEM)

Graph Embedding is a methodology aimed at representing a whole graph (with attributes attached to its nodes and edges) as a point in a suitable vector space; preserving the similarity/distance between the graphs that have been embedded i.e. the more two graphs are considered similar, the closer should be the corresponding points in the vector space.

Definition 15: Explicit Graph Embedding

Explicit graph embedding maps a graph to a point in suitable vector space. It encodes the graphs by equal size vectors and produces one vector per graph.

Mathematically, for a given graph $AG = (V, E, \mu^V, \mu^E)$, explicit graph embedding is a function ϕ, which maps graph AG from graph space G to a point $(f_1, f_2, ..., f_n)$ in n dimensional vector space $\mathbb{R}^n$. It is given as:

$$\phi : G \longrightarrow \mathbb{R}^n$$

$$AG \longmapsto \phi(AG) = (f_1, f_2, ..., f_n)$$

The main contribution of the thesis work is a method of explicit graph embedding.

1.5 Graph retrieval and subgraph spotting

Graph retrieval deals with retrieving a graph G from a graph repository, based on the similarity of graph G with an example (or query) graph.

Subgraph spotting takes the definition of graph retrieval to further granularity. Based on the similarity of a subgraph in G with the example (or query) graph, subgraph spotting refers to retrieving graph G from graph repository.

The second contribution of the thesis work concerns these research fields.

Chapter 2

State of the art

In this chapter we present a literature review on state of the art of structural pattern recognition, statistical pattern recognition, graph representation of images, exact graph matching, graph isomorphism, subgraph isomorphism, error tolerant graph matching, distance between graphs, graph embedding, graph classification and fuzzy logic. We summarize the important contributions of our work in light of the limitations of existing methods.

2.1 Introduction

Pattern recognition has emerged as an important research domain and has supported the development of numerous applications in many different areas of activity [De Sa, 2001] and [Friedman and Kandel, 1999]. The methods for pattern recognition are broadly categorized as statistical, structural or syntactic approaches [Bunke et al., 2001]. The statistical approaches are characterized by the use of numeric feature vectors, the structural approaches by the use of symbolic data structures and the syntactic approaches are characterized by the use of grammars. These three categories of methods have their own advantages and limitations.

This dissertation lies at the frontiers of the structural and statistical pattern recognition. Before proceeding with an in-depth review of state of the art on interesting topics for the work presented in the dissertation, we first briefly introduce the two aforementioned categories of pattern recognition.

The presentation of state of the art on graph matching and graph embedding is inspired by the PhD dissertations of [Qureshi, 2008] and [Sidère, 2012], respectively.

2.1.1 Structural pattern recognition

Structural pattern recognition is characterized by the utilization of symbolic data structures. The widely used symbolic data structures are graphs, strings and trees. Overtime the use of graph representations has become very popular in structural pattern recognition. This is because of the fact that both strings and trees are special instances of graphs [Bunke and Riesen, 2011b]. Thus we can safely term graphs to be the representative of symbolic data structures.

Graph based representations have their application to a wide range of domains, as graphs provide a convenient and powerful representation of relational information. They are able to represent not only the values of both symbolic and numeric properties of an object, but can also explicitly model the spatial, temporal and conceptual relations that exist between its parts. Graph do not suffer from the constraint of fixed dimensionality. For example the number of nodes and edges in a graph is not limited a priori and depends on the size and the complexity of the actual object to be modeled [Riesen and Bunke, 2009]. The most important advantage that graphs have is that they have foundations in strong mathematical formulation and have a mature theory at their basis.

However, along with the various advantages of graphs they have a serious drawback. Graph based representations are computational expensive. The much needed operations of graph matching and graph isomorphism are NP-complete.

A second serious drawback of graphs is that they are sensitive to noise.

We recommend [Riesen and Bunke, 2009], [Conte et al., 2004], [Bunke et al., 2005], [Bunke and Riesen, 2011b] and [Shokoufandeh et al., 2005] for an in-depth reading on structural pattern recognition.

2.1.2 Statistical pattern recognition

Statistical pattern recognition is characterized by the utilization of numeric feature vectors. The feature vectors are very basic representations. A very important advantage of these representations is that because of their simple structure, the basic operations that are needed in machine learning can easily be executed on them. This makes a large number of mature algorithms for pattern analysis and classification immediately available to statistical pattern recognition. And as a result of this fact, the statistical pattern recognition offers state of the art computational efficient tools of learning, classification

and clustering.

However, feature vector based representations have associated representational limitations. These limitations arise from their simple structure and the fact that feature vectors have same length and structure regardless of the complexity of object to be modeled [Ferrer et al., 2010].

We recommend [Duda et al., 2000] for a more detailed reading on statistical pattern recognition and classification.

2.2 Graph representation of images

Graph based structural pattern recognition representations are widely and successfully employed for image analysis and pattern recognition, at-least for the last 3 decades. Among the first few works using graphs for pattern recognition, the work of [Pavlidis, 1972] was based on the representation of topological properties of polygonal shapes by labeled graphs.

However, the problems of graph matching, graph isomorphism, maximum common subgraph and computation of edit distance between two graphs, which are very useful for pattern recognition, requires exponential computation time and are NP-complete.

Another important issue for pattern recognition with graph representations is their sensitivity to noise. As a result of noise and distortions in images, some variability in structural representation of different instances of same object may occur.

This makes it very important to increase their robustness against noise and distortions along with increasing computational strength of graph based representations of image content for pattern recognition.

Generally, graph representations of images are obtained by defining the image contents by using a set of primitives. These primitives form the vertices of the graph and the binary relations of compatibility between the primitives define the edges of the graph. The attributes of the graph (if any) represent the properties of the primitives and their compatibility relations in underlying image.

In this section we present some common graph based methods used in pattern recognition and image analysis with a focus on graphics recognition.

2.2.1 Graph of pixels

The graph of pixels is the classical representation of image content. The graph is constructed by representing each pixel by a vertex. The 4-connectivity or 8-connectivity of pixels defines the edges of graph.

Figure 2.1 shows an example for representation of an image of a character by graph of pixels, as used by [Franco et al., 2003] for character recognition and graphic symbol recognition.

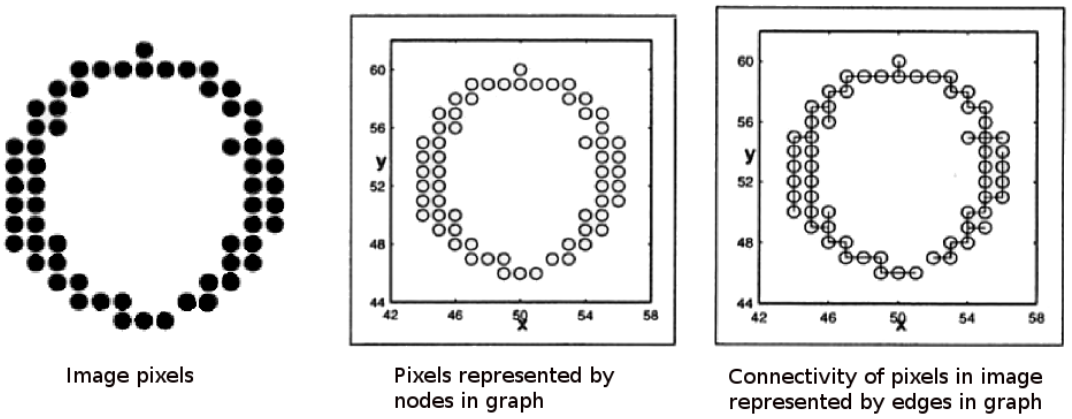


Figure 2.1: Representation of an image by graph of pixels.[1]

Graph of pixels is a mere change of representation of data and does not attempt to extract information from the image. The size of these graphs is often large. The computational limitation of classical graph processing algorithms prohibits the use of graph of pixels representation of images for real problems.

[1]Image from [Qureshi, 2008]

2.2.2 Graph of characteristic points

The graph of characteristic points can be seen as an extension to graph of pixels. They use only certain characteristic points in the image. A very common method to extract graph of characteristic points for object images is to first skeletonize the main parts of the object and then represent endpoints and junctions as vertices. The branches of skeleton between the endpoints and junctions define the edges between vertices of the graph.

Figure 2.2 shows an example for representation of an object by graph of characteristic points, as used by [Brunner and Brunnett, 2004] for 3D mesh segmentation and pattern recognition.

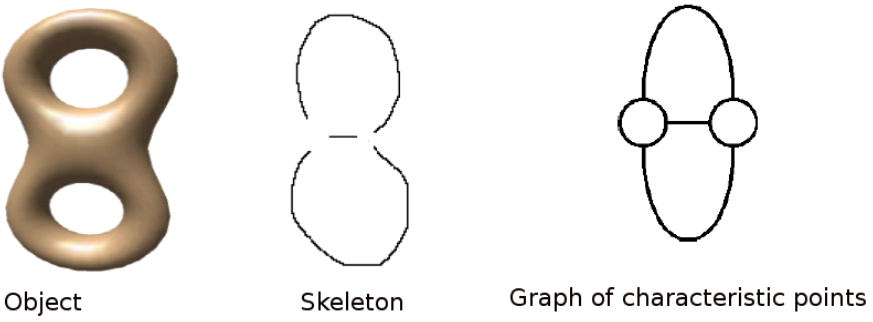


Figure 2.2: Representation of an image by graph of characteristic points.[1]

The methods for skeletonizing are not very computational expensive. However, skeletonizing of an image is not stable as the information in small segments may not necessarily be very representative of the image - considering the bigger aggregate.

[1] Image from [Qureshi, 2008]

2.2.3 Graph of primitives

Graph of primitives are based on more higher level of information than pixels or characteristic points, called primitives. Graph representation of image is achieved by representing the primitives in image as vertices of graph and the topological relationships between them as edges.

Figure 2.3 shows an example for representation of a graphic symbol by graph of primitives, as proposed by [Qureshi et al., 2007]. The image is vectorized to obtain a set of vectors (lines). The vectors are represented as vertices in graph and their topological relationships define the edges of the graph.

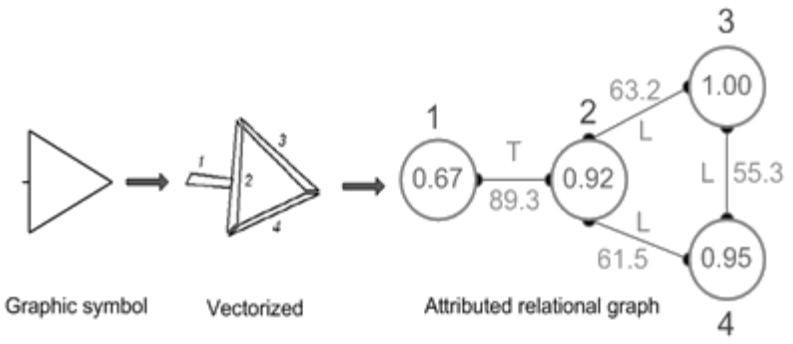


Figure 2.3: Representation of a graphic symbol by graph of primitives.

2.2.4 Region adjacency graph

Region adjacency graph represent the contents of an image by topology in which the contextual relationships are very important. Graph representation of an image is achieved by segmenting the image into a limited number of regions, such that each region represents the pixels that compose it. These regions are represented by vertices of the graph and the neighborhood relations between them define the edges of the graph.

Figure 2.4 shows an example of a region adjacency graph, as defined by [Ramel, 1992].

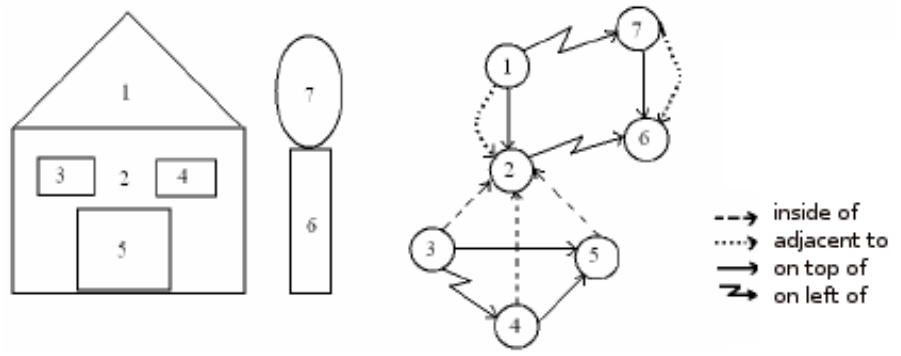


Figure 2.4: Representation of graphics content by a region adjacency graph.[1]

[1]Image from [Ramel, 1992]

2.2.5 Conclusion

Many other graph representations of images exists in literature. An exhaustive review of these graphs representations is not interesting for this dissertation, and thus we have discussed only the common graph representations which are used for pattern recognition.

The choice of the representation is very important for the genericity of the final system. The selection of attributes to be associated to the nodes and edges is a very important issue and it greatly influences performance of the tools that are eventually used to process and analyze the graphs. These attributes raise many computational constraints, which prohibit the use of classical methods from graph theory.

A important difference between the graphs used in pattern recognition for representing images, and the graphs used in other fields (for example the classical operational research dealing with problems of shortest path, graph coloring and graph partitioning etc.), is that in pattern recognition for representing various properties of the pixels regions in images, there is a necessity of both symbolic and numeric attributes on nodes and edges of the graphs.

Furthermore, in pattern recognition and image analysis, we often have a huge collection of graphs for representing the underlying data, which are often deformed by the noise in the data. These graphs, including the deformed ones, have to be compared for determining classes in data. This makes graph matching the most important operation for graph based pattern recognition and image analysis. In next section we outline some important methods of graph matching used for pattern recognition.

2.3 Graph matching

Graph based pattern recognition provides two very important advantages over statistical pattern recognition. These methods are flexible to adapt to underlying data and they have the representation power to model the spatial, temporal and conceptual relations in data. However, the much needed operation of graph comparison (i.e. determining if two graphs represent the objects in same class), is not simple and is computationally expensive.

The first works on graph matching, such as [Winston, 1970], [Pavlidis, 1972] or [Fischler and Elschlager, 1973], used exact algorithms. These graph matching algorithms are non tolerant to variations and try to find a exact match between two graphs. However, the recent works in literature, such as [Riesen et al., 2007] or [Solnon, 2010], more often employ the error tolerant versions of graph matching algorithms. The use of error tolerant algorithms permits to handle the variations between object of same class.

In this section we summarize the most representative techniques of graph matching for image processing and pattern recognition. For a comprehensive reading on the former, we recommend the reading of the classical reference [Conte et al., 2004] and the recent state of the art review in [Bunke and Riesen, 2011b].

Graph matching methods for pattern recognition are divided into several categories. An exhaustive review of all the methods is not possible in this dissertation. In this section we will only review the interesting methods for better placing our work w.r.t. the existing works in literature.

The first category of graph matching methods is the exact graph matching. This type of graph matching tries to find an exact match between the corresponding substructures of two graphs under consideration.

The second category of graph matching methods introduces some flexibility and tolerance to variations and is called error-tolerant graph matching. This type of graph matching considers the fact that some variations in graphs may occur as a result of noise in underlying data.

The third category of graph matching methods, discussed in this section, take boolean graph matching (i.e. aforementioned categories) one step forward and quantify the similarity between graphs. This type of graph matching defines a distance between graphs, for measuring the similarity between them.

The fourth category of graph matching methods, answers the computational complexity of distance based methods of graph matching. This type of graph matching embeds the graphs into feature vectors and exploits the computational strength of feature vector based statistical pattern recognition.

2.3.1 Exact graph matching and graph isomorphism

Formal definitions of the exact graph matching and graph isomorphism have been provided in Section 1.4 of the dissertation.

The exact graph matching algorithms tries to find a mapping between the nodes of two graphs, between their edges and between their labeling functions. The nodes and edges in a graph have no order associated to them. This makes the problem of exact graph matching more difficult and normally brute force approach is used to map the nodes, edges and labeling function of one graph to those of the other. The exact graph matching algorithms tries to find an isomorphism between graphs or between subgraphs (of two graphs). These methods are based on the use of a search tree with backtracking or are based on the use of adjacency matrix of graph for finding isomorphism between two graphs. Like all exact methods, these methods share the problem of combinatorial search space explosion - at worst exact graph matching algorithms are NP complete.

Graph isomorphism can be considered as the concept of formal equality of graphs. A related concept to graph isomorphism is the subgraph isomorphism, which could be considered as the concept of formal equality of subgraphs. The definition of subgraph isomorphism is given in Section 1.4 of the dissertation.

The algorithm presented in [Reingold et al., 1997] determines the exact isomorphism between two graphs by combinatorics. In a brute force manner, it finds and tests all the mapping functions f for isomorphism between the two graphs. In worst case this search requires $n!$ functions to be generated (n being the size of the graph). The complexity of this algorithm is considered to be in class NP [Garey and Johnson, 1979], there is no algorithm to solve it in polynomial time (P) and its membership in class NP-complete is not yet demonstrated [Köbler et al., 1993]. However, subgraph isomorphism has been shown to be in NP-complete [Read and Corneil, 1977]. The literature therefore focuses on reducing the complexity of graph isomorphism.

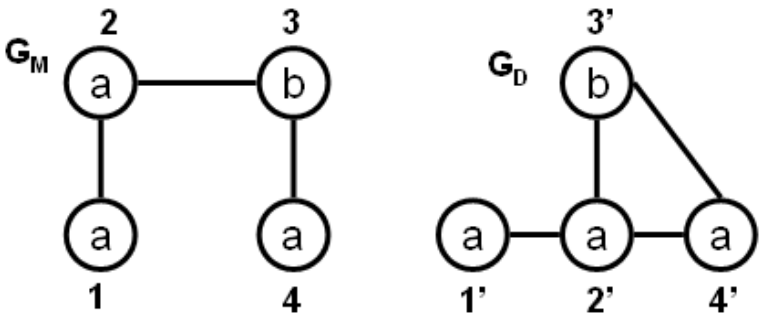
Below we introduce some of the methods for resolving the problem of graph isomorphism.

2.3.1.1 Graph isomorphism through association graphs

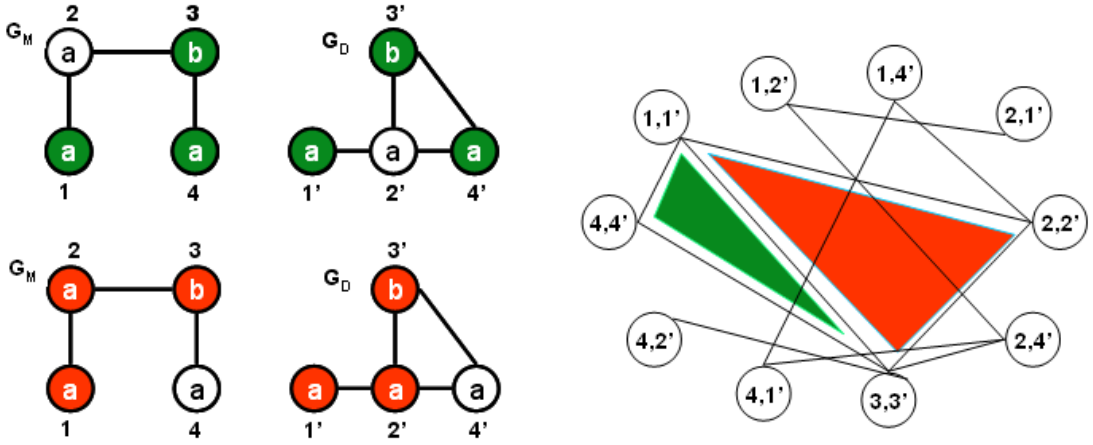
[Messmer, 1995] has proposed a method to find isomorphism between two graphs by employing an association graph. To find isomorphism between two graphs, an association graph is constructed from these two graphs. The total number of nodes in association graph is equal to the product of the number of nodes in given graphs. Each node in association graph is represented by a pair of nodes from the given graphs. The arcs between the nodes of the association graph are drawn according to following criteria; draw

an arc in association graph if there are arcs between both the pairs in original graphs or if there aren't any arcs between both the pairs in original graphs. Each clique of association graph corresponds to a subgraph isomorphism. We find the maximum clique in association graph to find the largest common sub graph between two given graphs. This method can have a complexity of $O((nm)^n)$, where n and m are the number of nodes of two graphs.

An illustration of this method is shown in Figure 2.5.



(a) Two graphs G_M and G_D.



(b) Association graph for G_M and G_D. The cliques are marked with a green and a red triangle in association graph.

Figure 2.5: Graph isomorphism through association graph.[1]

[1]Image from [Qureshi, 2008]

2.3.1.2 Method based on decision trees

Decision trees have been employed for graph mapping by [Corneil and Gotlie, 1970]. To find a mapping between two graphs G_M and G_D; each node of G_D is iteratively mapped to the nodes of the G_M, provided that the structure of the arcs of the G_D is preserved. Thus if the two graphs have same number of nodes and if all the nodes of the two graphs are mapped, then there exists an isomorphism between them. This method is based on brute force or exhaustive search and has a complexity of $O(N^4)$. [Ullman, 1976] has improved this method by introducing backtracking and a procedure of forward checking for reducing the search space.

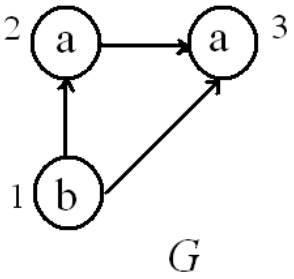
2.3.1.3 Method based on ordered graphs

[Jiang and Bunke, 1999] have proposed a graph isomorphism method based on ordered graphs. In this method, first the original graph is transformed to an Eulers Graph. Then by using Eulers circuit a code (string) is generated for each graph. The isomorphism between given two graphs can then be found by using a simple algorithm on these codes. This method solves the isomorphism problem in quadratic time to the number of nodes of graphs.

2.3.1.4 Method based on decision trees and adjacency matrix

[Messmer, 1995] has also proposed a method based on the use of decision trees and adjacency matrices for finding isomorphism between two graphs or their sub graphs. Different permutations of the adjacency matrix are generated for each graph. Then a decision tree is generated from these adjacency matrices. This method solves the isomorphism problem in quadratic time to the number of nodes of graphs.

Figure 2.6 shows a graph and different permutations of its adjacency matrices. And Figure 2.7 illustrates the graph isomorphism method of [Messmer, 1995].



(a) Graph.

A

1	2	3
b	1	1
0	a	1
0	0	a

B

1	3	2
b	1	1
0	a	0
0	1	a

C

2	1	3
a	0	1
1	b	1
0	0	a

D

2	3	1
a	1	0
0	a	0
1	1	b

E

3	1	2
a	0	0
1	b	1
1	0	a

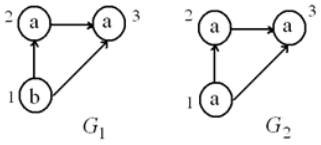
F

3	2	1
a	0	0
1	a	0
1	1	b

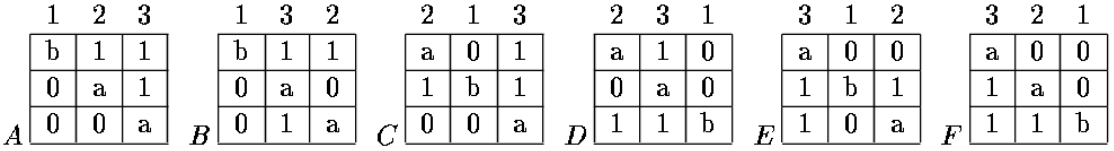
(b) Adjacency matrices.

Figure 2.6: A graph and its adjacency matrices.[1]

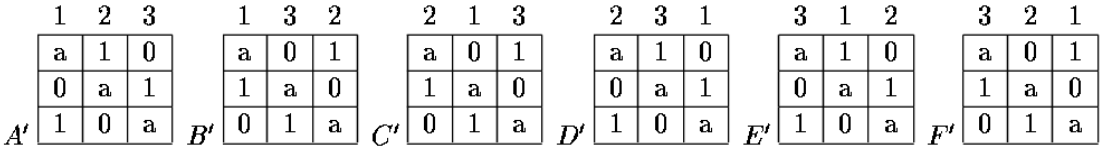
[1]Image from [Qureshi, 2008]



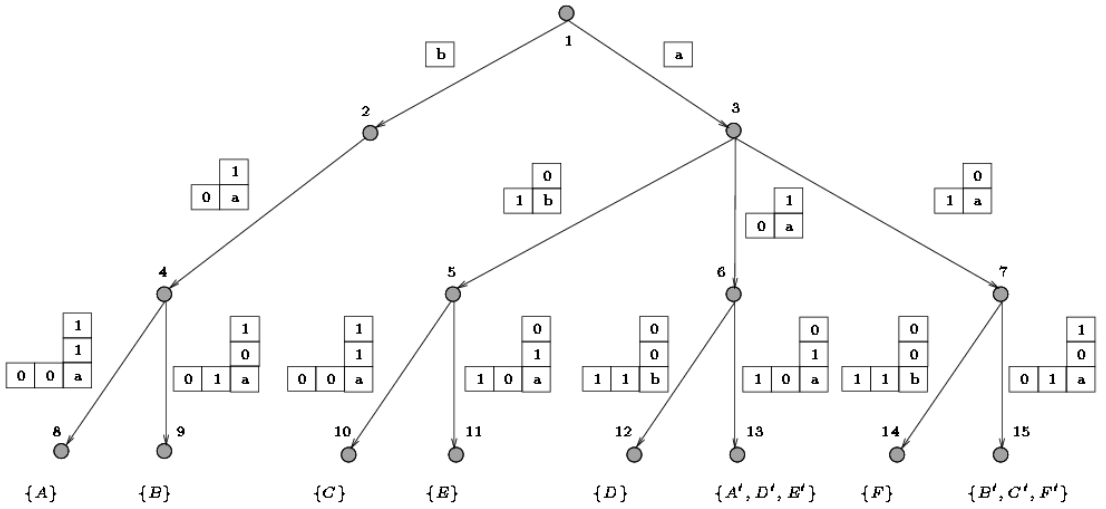
(a) Graphs.



(b) Adjacency matrices G_1.



(c) Adjacency matrices G_2.



(d) Decision tree.

Figure 2.7: Decision tree constructed from the adjacency matrices of two graphs G_1 and G_2.[1]

[1]Image from [Qureshi, 2008]

2.3.1.5 Methods based on decomposition of graphs

[Sonbaty and Ismail, 1998] have proposed a method of graph isomorphism based on decomposition of graphs. The idea behind these methods is to decompose the graph into several small graphs, which are named Basic Attributed Relational Graphs (BARG). One BARG is generated for each node of the original graph. The BARG of a node X has node X as root node and the nodes which are directly connected to X in original graph as direct children of root. The problem of graph mapping is, in this way, reduced to mapping between the BARG. A matrix of graph edit distances is constructed by computing distance between every pair of BARGs. The distance between two BARGs is computed on basis of the number of basic operations required for converting one to other (e.g. addition of node/edge, deletion of node/edge). The complexity of the algorithm is $O(m^2 * n^2)$, where m and n are the number of nodes of two graphs. Another algorithm under this category decomposes the graphs repeatedly into subgraphs to obtain small parts, which are composed of a single node. A hierarchical structure is constructed from these small parts. The isomorphism can then be found incrementally, by progressively traversing this hierarchical structure of sub graphs.

2.3.2 Error tolerant graph matching

In certain applications, the representation of an object by graph can slightly fluctuate because of noise and distortion in images. It therefore becomes necessary to introduce an error model or to integrate the concept of tolerance during the mapping of the graphs. Thus, the term "error tolerant" applied to certain problems of graph matching means that it is not always possible to find an isomorphism between the two graphs. This can arise in cases where the number of nodes is different in the two graphs. Consequently, the problem of graph matching no longer remains the problem of finding an exact match but finding the similarity between graphs.

2.3.2.1 Method based on decision trees

[Messmer and Bunke, 1998] have adapted their method of graph isomorphism [Messmer, 1995], to achieve error tolerant graph matching. They have envisaged the correction of errors during the creation of decision tree. For each model graph, examples with distortions are generated and compiled in the decision tree. The number of examples depends upon the maximum acceptable error. During the matching (recognition), the decision tree is used in a traditional way.

The complexity of execution time of this process remains quadratic compared to the number of nodes in the query graph. However, the size of the decision tree increases exponentially with the number of nodes in the model graphs and thus depends much on the envisaged degree of deformations. Consequently, this approach is limited to the graphs of very small size. Moreover, it is very difficult to envisage the types of errors which will occur in the real cases.

In a second approach in [Messmer and Bunke, 1998], the corrections of errors are considered during a later stage. The decision tree representing the whole of model graphs does not incorporate any information about the possible errors. It is, in fact, the query graph which is transformed to produce a set of deformed copies of the graph. Each graph is then classified by the decision tree. Execution time complexity of this method is $O(d * n^{2(d+1)})$ where n is the number of nodes in the query graph and d is a threshold which defines the maximum number of acceptable operations of edition to carry out the deformations [Messmer and Bunke, 1998].

2.3.2.2 Spectral methods

The adjacency and Laplacian matrices of graphs provide an interesting representation, given the available mathematical tools available for matrices. However, the matching of two graphs using the matrices is complex. Actually the adjacency and Laplacian matrices of graphs are based on the ordering of numbers assigned to the nodes. The latter is assigned randomly. The rows and columns can be swapped between two matrices representing the same graph. The problem of error tolerant graph isomorphism can be considered a combinatorial problem since it requires a step for assigning rows of a matrix to another. One of the methods employed, is the Hungarian algorithm which is of polynomial complexity (thus the advantage of the use of matrices is limited).

In contrast, the methods based on spectral theory of graphs, [Chung, 1997] and [Godsil and Royle, 2011], analyze the values and eigenvectors of adjacency or Laplacian matrices. The eigenvalues are independent of permutations of vertices. In fact, if two graphs are isomorphic, then their spectrum are equal, without using an assignment in advance. The reverse is not true - non isomorphic graphs can have the same spectrum (they are called co-spectral).

Among the first methods, is a method proposed by [Umeyama, 1988]. The author uses the decomposition of eigenvalues and eigenvectors of adjacency matrices, to deduce the orthogonal matrix that is optimized later. For this, suppose a priori that graphs are isomorphic. If they are, then the method finds the optimal permutation. If graphs are close to the isomorphism, then the solution is suboptimal. However, for non-isomorphic graphs the quality of results decreases.

This idea is used in combination with an approach of clustering by [Jain et al., 1999]. A first work propose to use a clustering of the vertices before matching to associate the clusters first and then the vertices [Carcassoni and Hancock, 2001]. A second work on the other hand propose to project the vertices in the graphs eigen space and then use the clustering in this space to find the matching of nodes [Caelli and Kosinov, 2004]. The spectral methods also give rise to methods exploiting the characteristics random paths [Gori et al., 2005].

The weakness of these approaches is that they are based on the matrices of graphs which do not take the labels of nodes and edges into account. This limits them only to some restricted set of applications.

2.3.3 Distance between two graphs

The graph matching methods mentioned above are based on exact or error tolerant graph matching. These algorithms makes it possible to find if two graphs are identical or if they meet the possible tolerance constraints. Two graphs with certain differences, not enough to be matched, are considered dissimilar. Which means that the similarity between them is deemed void and distance between them is regarded as infinity.

In many applications the comparison is seldomly done as a boolean operator of identical and non-identical. Generally, comparison is regarded more as a continuous measure of similarity. From this point of view, the graph matching methods (exact and error tolerant) doest not always effectively respond to the issues of pattern recognition. To overcome this shortcoming, the methods detailed in this section propose to establish a measure of distance between graphs.

Formally in mathematics, the distance is an application on a set E, called metric space, such as:

$$d : E \times E \longrightarrow \mathbb{R}^n$$

In addition, it must verify the following properties:

- symmetry: $\forall x, y \in E,\ d(x, y) = d(y, x)$
- Separation: $\forall x, y \in E,\ d(x, y) = 0 \Leftrightarrow x = y$
- triangular inequality: $\forall x, y \in E,\ d(x, z) \leq d(x, y) + d(y, z)$

This definition is directly applicable to two shapes which are represented by feature vectors. The Euclidean space in which the vectors (for statistical pattern recognition methods) are defined, by definition allows to use all the metrics defined in this space (including the distances).

In this section we introduce two widely employed distances for graphs, namely the graph edit distance and the distance from the largest common subgraph.

2.3.3.1 Graph edit distance

A formal definition of graph edit distance is introduced in Section 1.4 of the dissertation.

The graph edit distance is a method inspired by the work on string edit distance [Levenshtein, 1966], and generalized to the case of graphs. The main idea of this distance is a dissimilarity measure based on the number and strength of the transformations to be applied on a set - a string - to transform it into another. The concept of edit distance has been extended from strings to more complex structures like trees in a first attempt ([Selkow, 1977] and [Tai, 1979]) and then to attributed graphs ([Tsai and Fu, 1979] and [Eshera and Fu, 1984]).

The general idea of graph edit distance is to define the dissimilarity of two graphs by the distortion required to transform one graph into another. Specifically, the graph edit distance is modeled by the path of least edition cost representing the sequence of processing operations by associating a cost. The cost is low if the graphs have many similarities, and otherwise the cost is higher.

Let $g_1 = (V^1, E^1, \mu^{V^1}, \mu^{E^1})$ and $g_2 = (V^2, E^2, \mu^{V^2}, \mu^{E^2})$ be two graphs. The graph edit distance between g_1 and g_2 is defined as:

$$d(g_1, g_2) = \min_{(e_1,\ldots,e_k)\in\gamma(g_1,g_2)} \sum_{i=1}^{k} c(e_i)$$

where,

$\gamma(g_1, g_2)$ denotes the set of edit paths transforming g_1 into g_2 and

c denotes the cost function measuring the strength $c(e_i)$ of edit operation e_i.

The function c includes all possible costs of edit operations. The edit operations (distortions) are usually adding or removing a node, the addition or deletion of an edge and changing a node label and changing an edge label. The strengths and weaknesses of this distance are concentrated to the function c. It allows to adapt the cost, depending on the problem domain, by penalizing, more or less, the various distortions to apply. And because of the very same reason, these cost functions parameters are sensitive and difficult to resolve [Bunke, 1999]. Some work propose the automatic learning of these costs, thus facilitating the parameter setting of the cost function ([Neuhaus and Bunke, 2004], [Neuhaus and Bunke, 2007]). One of the strengths of this technique is its genericity. This distance is therefore often referred to evaluate the quality of a similarity measure between two graphs.

The calculation of the edit distance is based on a search tree representing all solutions of matches, each node of the tree represents a pairing and a path - path from the root to a leaf - is a possible match, but not necessarily optimal, between two graphs. This tree is built dynamically - like the pairing with the tree - by optimizing the probable search cases of vertices to vertices matching by employing a heuristic ($A*$ for example ([Hart et al., 1968])). The complexity of the algorithm is still exponential in the number of vertices. The flexibility of the edit distance can thus be used on a wide variety of graphs without constraints on labels or on the topology, but its application is restricted to small graphs only. In [Neuhaus, M. et Bunke, 2006], the authors propose a solution to reduce this complexity at the cost of suboptimal matching.

2.3.3.2 Distance from the largest common subgraph

The search for the largest common subgraph is also a common method used for the calculation of distance between graphs. Other methods based on its complement, the search for the smallest super graph, also exist.

The largest common subgraph of two graphs g_1 and g_2 is the largest graph that is both subgraph of g_1 and subgraph of g_2. It can therefore be seen as an intersection of g_1 and of g_2. A standard approach to extract the greatest common graph of two graphs is related to the notion of finding maximum clique. Methods based on this approach and their comparison are presented in [Bunke et al., 2002]. The approach of McGregor [McGregor, 1982] is recognized as a reference.

The largest common subgraph permits to measure the similarity between two objects. Intuitively, a larger common subgraph of g_1 and g_2 will be large if the two graphs are very similar, small otherwise. From the largest common subgraph determined, more distance calculations are envisaged. The distance so determined is used to translate the similarity between graphs. [Pelillo, 1999] formalizes the pelillo metric space and the definition of distance from larger common subgraph. For example, we present the two distances the most commonly used in the literature.

In [Bunke, 1997], Bunke defines a distance such that:

$$d(g_1, g_2) = |g_1| + |g_2| - 2 \times |mcs(g_1, g_2)|$$

where mcs refers to maximum common subgraph.

This distance establishes a relationship between the graph edit distance and the size of the largest common subgraph.

In [Bunke and Shearer, 1998], the authors define the distance:

$$d(g_1, g_2) = \frac{|mcs(g_1,g_2)|}{max(|g_1|,|g_2|)}$$

The distance is normalized between 0 and 1.

2.3.4 Graph embedding

Over decades of research in pattern recognition, the research community has developed a range of expressive and powerful approaches for diverse problem domains. Graph based structural representations are usually employed for extracting the structure, topology and geometry, in addition to the statistical details of underlying data. During the next step in the processing chain, generally these representations could not be exploited to their full strength because of limited availability of computational tools for them. On the other hand, the efficient and mature computational models, offered by statistical approaches, work only on vector data and cannot be directly applied to these high-dimensional representations. Recently, this problem has been addressed by the emerging research domain of graph embedding.

Graph embedding is a natural outcome of parallel advancements in structural and statistical pattern recognition. It offers a straightforward solution, by employing the representational power of symbolic data structures and the computational superiority of feature vectors [Bunke et al., 2005]. It acts as a bridge between structural and statistical approaches [Bunke et al., 2001, Roth et al., 2003], and allows a pattern recognition method to benefit from computational efficiency of state-of-the-art statistical models and tools [Chen et al., 2007] along-with the convenience and representational power of classical symbolic representations. This permits the last three decades of research on graph based structural representations in various domains [Conte et al., 2004], to benefit from the state-of-the-art machine learning models and tools. Graph embedding has its application to the whole variety of domains which are entertained by pattern recognition and where the use of a relational data structure is mandatory for performing high level tasks. Apart from reusing the computational efficient methods for vector spaces, another important motivation behind graph embedding methods is to solve the computationally hard problems geometrically [Shaw and Jebara, 2009]. The pattern recognition research community acknowledges the emerging importance of graph embedding methods [Foggia and Vento, 2010] for realizing the classical idea of using the structural and statistical methods together. We refer the interested reader to [Lee and Madabhushi, 2010] for further reading on the applications of graph embedding.

The graph embedding methods are formally categorized as implicit graph embedding or explicit graph embedding.

The implicit graph embedding methods are based on graph kernels. A graph kernel is a function that can be thought of as a dot product in some implicitly existing vector space. Instead of mapping graphs from graph space to vector space and then computing their dot product, the value of the kernel function is evaluated in graph space. Such an embedding satisfies the main mathematical properties of dot product. Since it doest not explicitly map a graph to a point in vector space, a strict limitation of implicit graph embedding is that it does not permit all the operations that could be defined on vector spaces. We refer the interested reader to [Riesen and Bunke, 2010a],

[Riesen and Bunke, 2010b, Foggia and Vento, 2010] for further reading on graph kernels and implicit graph embedding.

On the other hand, the more useful, explicit graph embedding methods explicitly embed an input graph into a feature vector and thus enable the use of all the methodologies and techniques devised for vector spaces. The vectors obtained by an explicit graph embedding method can also be employed in a standard dot product for defining an implicit graph embedding function between two graphs [Bunke and Riesen, 2011b]. Explicit graph embedding requires that graphs must be embedded in pattern spaces in a manner that similar structures come close to each other and different structures goes far away i.e. an implicit clustering is achieved [Wilson et al., 2005]. Another important property of explicit graph embedding is that the graphs of different size and order needs to be embedded into a fixed size feature vector. This means that for constructing the feature vector, an important step is to mark the important details that are available in all the graphs and are applicable to a broad range of graph types. We refer the interested reader to [Foggia and Vento, 2010] for further reading on explicit graph embedding.

Graph embedding is an interesting solution for addressing the computational limitations of structural pattern recognition and in particular for addressing the problem of graph isomorphism which belongs to the class of NP-complete problems. The mapping of high dimensional graph to a point in suitable vector space permits to perform the basic mathematical computations which are required by various statistical pattern recognition techniques. This makes graph embedding a good solution to address the problems of graph clustering and classification. However, in our opinion, graph embedding lacks the capabilities to address the problem of graph matching. This is because of the strict limitation of the resulting feature vector which is not capable of preserving the matching between nodes of graphs.

Recent research surveys on graph embedding are presented by [Lee and Madabhushi, 2010, Foggia and Vento, 2010, Bunke and Riesen, 2011b].

In literature the problem of explicit graph embedding has been approached by three important families of algorithms.

- Graph probing based methods
- Spectral based graph embedding
- Dissimilarity based graph embedding

2.3.4.1 Graph probing based methods

The first family of graph embedding methods is based on the frequencies of appearance of specific knowledge-dependent substructures in graph. This embedding is based on feature extraction on structural data. These numeric features capture both topology information (number of nodes, arcs, degree of nodes) and the contents of the graph (histogram of the labels for example). This embeds a graph into a feature vector in Euclidean space.

One of the first methods of graph probing dates back to 1940's ([Wiener, 1947]). In this method, each graph is represented by an index called Wiener index. The index is a topological descriptor defined by the sum of all shortest paths in the graph, such that if $G = (G(V), G(E))$ is a graph, then the Wiener index $W(G)$ of G is:

$$W(G) = \sum_{v_i \in G(V)} \sum_{v_j \in G(V)} l(v_i, v_j)$$

where $l(v_i, v_j)$ is a function that defines the length of the shortest path between the node and the node v_i and v_j in the graph G.

Another approach is introduced in [Papadopoulos, A. N. et Manolopoulos, 1999]. For undirected and unattributed graphs, the authors calculate the degree of each node then form a sorted histogram of degrees. So they get the vector $\phi(G) \in \mathbb{R}^n$ such that if $G = (V(G), E(G))$ is a graph then $\phi(G) = (n_0, n_1, n_2, ...)$ where $n_i = \{v \in V | d_G(v) = i\}$. The features of the resulting vector are the number of nodes of degree 1, the number of nodes of degree 2, and so on.

An extension of this technique is proposed in [Lopresti and Wilfong, 2003]. The authors also suggest a representation based on local descriptors of nodes and generalize the method for all types of graphs. This representation is based on the degree of vertices in the case of undirected and unlabeled graphs. In the case of directed graphs, the representation uses in degree and out degree of nodes. This representation can be adapted to take into consideration labeled graphs, taking into account, in addition to the degree of the nodes, the labels of the connected edges. On the other hand, the authors present an interesting relationship between the graph edit distance and the distance between the embedding of two graphs. They have shown that the distance between the embedding of two graphs g_1 and g_2, given by $d_{GEM}(g_1, g_2)$, is less than 4 times of the graph edit distance between them, given by $d_{GED}(g_1, g_2)$. Mathematically this is given as $d_{GEM}(g_1, g_2) \leq 4 \times d_{GED}(g_1, g_2)$.

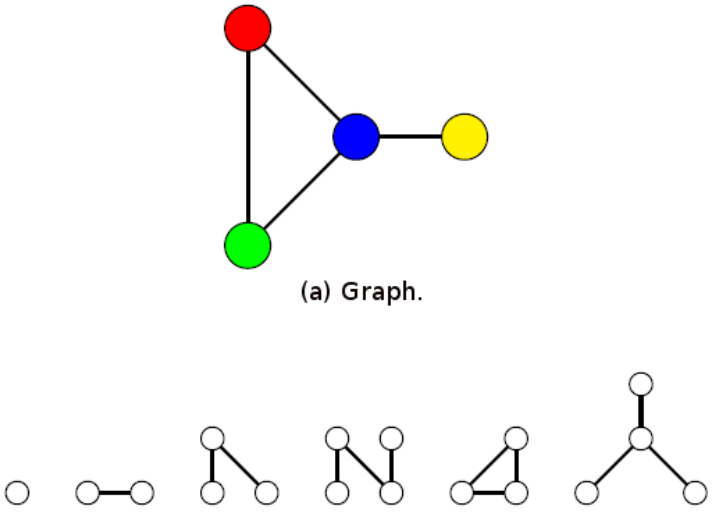
Recently, in [Gibert et al., 2011a] and [Gibert et al., 2011c], the authors have proposed a graph embedding method by counting the frequency of appearance of specific set of representatives of labels of nodes and their corresponding edges. In [Gibert et al., 2011b] the authors propose an improvement of their graph embedding technique by dimensionality reduction of the obtained feature vector.

The algorithms presented in the cited works provide an embedding of graph into feature vector, in linear time complexity. Their simplicity of implementation is an important advantage of this family of methods. However, the features they use, are very localized to nodes and arcs. The graph embedding contains little information on the topology, which can have a negative impact on the classification results.

A sub family of this category of graph embedding methods is based on the frequencies of appearance of specific knowledge-dependent substructures in graph. These works are mostly proposed for chemical compounds and molecular structures. Graph representation of molecules are assigned feature vectors whose components are the frequencies of appearance of specific knowledge-dependent substructures in graphs ([Kramer and Raedt, 2001] and [Inokuchi et al., 2000] for example).

Recently, Sidere in [Sidère et al., 2008], [Sidère et al., 2009a], [Sidère et al., 2009b] and PhD thesis [Sidère, 2012], has proposed a graph embedding method. The method is based on the extraction of substructures of 2 nodes, 3 nodes , 4 nodes and so on, from a graph. The feature vector representation is then obtained by counting the frequencies of these substructures in the graph (Figure 2.8). The method proposed by Sidere is rich in topological information and is very interesting for pattern recognition.

The methods based on the frequencies of appearance of specific knowledge-dependent substructures in graph, are based on finding subgraphs in graph and are capable of exploiting domain knowledge. However, they have a drawback that finding substructures in graphs is computationally challenging.



(a) Graph.

(b) A lexicon of non-isomorphic graph network.

Figure 2.8: A graph and a lexicon generated from a non-isomorphic graph network.

2.3.4.2 Spectral based graph embedding

The second family of graph embedding algorithms is spectral based embedding. Spectral based embedding is a very prominent class of graph embedding methods and is proposed by lots of works in literature. In order to embed graphs into feature vectors, this family of methods extract features from graphs by eigen-decomposition of adjacency and Laplacian matrices and then apply a dimensionality reduction technique on the eigen-features. Many work for graph embedding exploit the spectral theory of graphs ([Chung, 1997]) and are interested in the properties of the spectrum of a graph and the characterization of the topology graph using eigenvalues and eigenvectors of the adjacency matrix or Laplacian matrix.

In [Harchaoui, 2007], graph embedding using the spectral approach is proposed. The authors use the leading eigenvectors of the adjacency matrix to define the eigenspaces of adjacency matrices. Spectral properties are then calculated for each eigenmode. In [Luo et al., 2003], the authors also use the adjacency matrices of graphs as a support and then compute their eigenvectors and thus obtain modes to define the vector space (perimeter, volume, Cheeger number etc.). Construction of the vector is completed by applying the dimensionality reduction using PCA (Principal Component Analysis), ICA (Independent Component Analysis) or MDS (Multidimensional Scaling).

Graph embedding has also been approached by using Laplacian matrix. For example, in [Robles-Kelly and Hancock, 2007], a spectral approach is proposed using the Laplace-Beltrami operator to embed the graphs in a Riemannian manifold or a metric where one can define the length of a path (called a geodesic) between two points of the manifold. This length of the path is then used to calculate similarity between graphs. In [Kosinov and Caelli, 2002], the embedding of the nodes of a graph is performed on an eigenspace defined by the first eigenvectors. In [Wilson et al., 2005], the authors propose to use a spectral decomposition of Laplacian matrix, and construct the symmetric polynomials. The coefficients of these polynomials are then used for graph embedding.

As for graph matching techniques, the spectral theory of graphs shows interesting properties that can be used for graph embedding. Its main benefit is the linear complexity associated with operations on matrices. The spectral family of graph embedding methods provide solid theoretical insight into the meaning and significance of extracted features. However these approaches have some limitations. Spectral methods are sensitive to noise, removing a node, for example, changes the matrices (adjacency or Laplacian) and these errors affect the embedding functions. In addition, their use is restricted to unlabeled graphs making it difficult to use these approaches in most applications of pattern recognition.

2.3.4.3 Dissimilarity based graph embedding

Finally the third family of graph embedding algorithms is based on dissimilarity of a graph from a set of prototypes. The dissimilarity based graph embedding can handle arbitrary graphs. The dissimilarity based graph embedding methods usually use graph edit distance and exploit domain knowledge. But since graph edit distance is computationally expensive, the dissimilarity based graph embedding methods may become computationally challenging.

Unlike the aforementioned approaches of graph embedding, the dissimilarity based graph embedding techniques do not focus on the extraction of a vector from the graph. Rather, the idea is to construct a vector by comparing a graph with a selection of graphs - called prototypes.

In [Pekalska and Duin, 2005], the authors present dissimilarity based representation as an alternative to the feature vectors. Starting from the postulate that the dissimilarity is used to separate a class of objects from another, the authors offer to characterize objects not by absolute attributes but as a vector of dissimilarity from objects of other classes. An object is defined by its embedding in a dissimilarity space.

In [Riesen et al., 2007], [Ferrer et al., 2008], [Riesen and Bunke, 2009] and [Riesen and Bunke, 2010c], the authors propose to adapt this dissimilarity space for the construction of a graph embedding function. The main idea of this work is to construct a vector of graph edit distances from the graph to be embedded and a set of k prototypes selected in the graph database. The embedding of the graph is thus a vector of k distances. Formally, let $\Gamma = g_1, ..., g_n$ be a set of graphs and $p = p_1, ..., p_k \subset \Gamma$ be a subset of selected prototypes from Γ. The graph embedding is defined as the function $\Phi : \Gamma \longmapsto (\mathbb{R})^k$, such that $\Phi(g) = [d(g, p_1), ..., d(g, p_k)]$ where $d(g, p_i)$ is the graph edit distance between graph g and the i^{th} prototype graph in p.

In [Bunke and Riesen, 2011a] and [Bunke and Riesen, 2011b], the authors propose an improvement of the graph embedding method by using feature selection method.

This type of projection is very efficient. The difficulty of setting the edit distance - mentioned above - is found in this method. In addition, the choice of prototype graphs is also a significant parameter as it determines the size of the vector and its capacity to effectively represent the graph in the vector space. In addition, it remains highly dependent on the application and its learning set. In [Riesen and Bunke, 2009] and [Riesen, 2010], the authors have given some indications on the choice of the prototype graphs.

2.4 Fuzzy logic

Fuzzy logic is a soft computing paradigm and is a form of knowledge representation suitable for notions that cannot be defined precisely, but which depend upon their contexts. It is a superset of conventional boolean logic that has been extended to handle the concept of partial truth i.e. truth values between "completely true" and "completely false".

Fuzzy logic recognizes more than simple true and false values. In contrast with conventional boolean logic theory, where binary sets have two-valued logic i.e. true or false, fuzzy logic variables may have a truth value that ranges in degree between 0 and 1.

With fuzzy logic, propositions can be represented with degrees of truthfulness and falsehood. For example, the statement, today is sunny, might be 100% true if there are no clouds, 80% true if there are a few clouds, 50% true if it's hazy and 0% true if it rains all day.

Figure 2.9 represents the meaning of the expressions cold, warm, and hot by functions mapping a temperature scale. A point on the scale has three "truth values" one for each of the three functions. The vertical line represents a particular temperature that the three arrows (truth values) gauge. Since the red arrow points to zero, this temperature may be interpreted as "not hot". The orange arrow (pointing at 0.2) may describe it as "slightly warm" and the blue arrow (pointing at 0.8) "fairly cold".

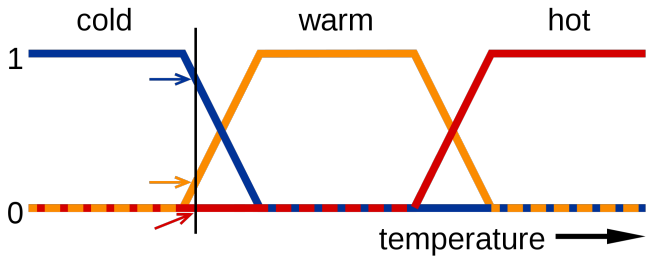


Figure 2.9: Example of fuzzy logic (temperature).[1]

[0]Image courtesy of `http://en.wikipedia.org/wiki/Fuzzy_logic`

Fuzzy sets were first proposed by Lofti A. Zadeh in [A. and Zadeh, 2008]. This paper laid the foundation for the modern fuzzy logic. It mathematically defined fuzzy sets and their properties, as given by:

Let X be a space of points, with a generic element of X denoted by x. Thus $X = \{x\}$. A fuzzy set A in X is characterized by a membership function $f_A(x)$ which associates with each point in X a real number in the interval $[0, 1]$, with the values of $f_A(x)$ at x representing the "grade of membership" of x in A. Thus, the nearer the value of $f_A(x)$ to unity, the higher the grade of membership of x in A.

This definition of a fuzzy set is like a superset of the definition of a set in the ordinary sense of the term. The grades of membership of 0 and 1 correspond to the two possibilities of truth and false in an ordinary set. The ordinary boolean operators that are used to combine sets no longer apply; we know that 1 AND 1 is 1, but what is 0.7 AND 0.3?

Membership functions for fuzzy sets can be defined in any number of ways as long as they follow the rules of the definition of a fuzzy set. The shape of the membership function used defines the fuzzy set and so the decision on which type to use is dependent on the purpose. The membership function choice is the subjective aspect of fuzzy logic, it allows the desired values to be interpreted appropriately. Figure 2.10 shows the commonly employed shapes of trapezoid, triangle and Gaussian, for the fuzzy membership functions.

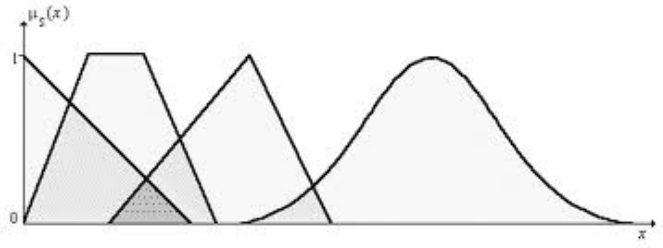


Figure 2.10: Some shapes commonly employed for the membership function $S(x)$.[1]

[1]Image from [Borges, 1996]

Traditional boolean logic uses the boolean operators AND, OR, and NOT to perform the **intersection, union and complement operations**. These operators work well for boolean sets but fuzzy logic does not have a finite set of possibilities for each input. The operators need to be defined as functions for all possible fuzzy values, that is, all real numbers from 0 to 1 inclusive. The operators for fuzzy sets are given in Table 2.1. Figure 2.11 shows an illustration of these operators for triangular membership function of fuzzy logic.

Table 2.1: Defining operators for fuzzy sets.

Operator	**For fuzzy sets**
A AND B	min(A, B)
A OR B	max(A, B)
NOT A	(1 - A)

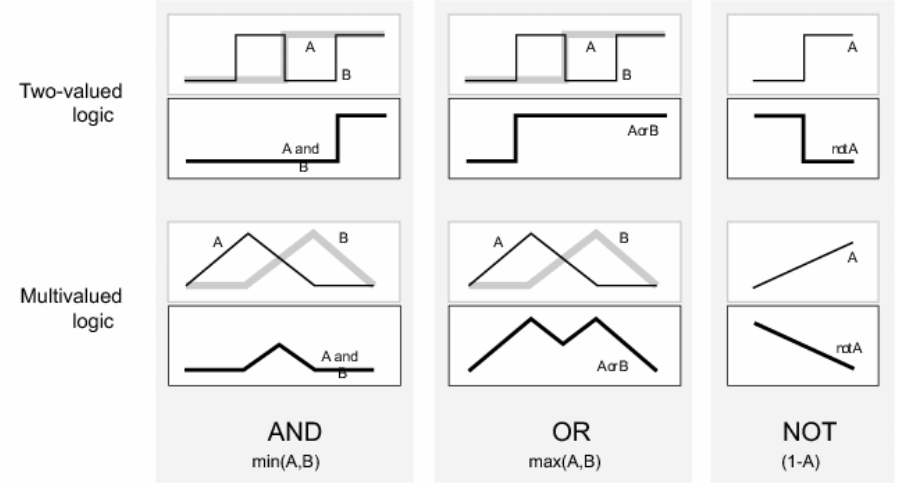


Figure 2.11: Pictorial illustration of operations on boolean and fuzzy logic.[1]

[1]Image courtesy of `http://www.mathworks.fr/help/toolbox/fuzzy/bp78l6_-1.html`

We refer the interested reader to [Klir and Yuan, 1995], [A. and Zadeh, 2008], [Radzikowska and Kerre, 2002] and [Liu, 2010] for further reading on fuzzy logic and soft computing.

We have employed fuzzy logic to incorporate smooth transitions between the overlapping intervals of data (representing the various attributes of graphs). This enables our method to minimize the information loss while mapping from continuous graph space to discrete feature vector space, and to gracefully handle noise (and deformations) in graphs, for proposing a robust embedding of graphs into numeric feature vectors.

2.5 Limitations of existing methods and our contributions

In this section, we discussed the broad classification of graph comparison methods by providing a summary of the important methods in each category.

First we introduced the methods of exact graph matching. From a graph theory approach, such as the isomorphism of graphs and subgraphs, these techniques are designed to verify the similarity between two graphs. However, their rigid character penalize the distortions in graphs. Their performance is related to the quality of representation of data by graphs. In addition to computing the similarity between two graphs, these methods of graph matching provide details on the comparison of the topologies of the two graphs with a proposed mapping between the nodes of the two graphs nodes. This information turns out to be very useful in a system of information retrieval where the query is a partial object that we are looking to locate in a bigger object. Pattern recognition by graphs allows this type of search through the use of matches. On the other hand the statistical approaches require a priori segmentation of objects, or using a sliding window, involving a difficult parameter which highly influences the the quality of results. The graph matching algorithms provide the node to node mapping between two graphs only if an isomorphism exists (and is established) between them.

All distortions result into a no-match, whatsoever the are. To address this problem, the error tolerant graph matching methods suggest to relax the constraints on the matching of nodes and arcs. They find the isomorphism, tolerant to noise. This allows them to be applicable to the data that contains distortions (which exist very often in pattern recognition and image analysis). The inexact matching methods offer greater tolerance to match two nodes (or two edges) with slight difference. But these methods are complex to implement.

We have seen that the other proposed methods allow more tolerance to noise but lose the matching between vertices of graphs. The distances between graphs are defined as the most intuitive to quantify similarities between graphs. They provide a measure that is effective when used in classification and robust to different noise. However, the algorithms are complex and this complexity can be a significant obstacle in some cases.

Implicit graph embedding in a vector space, bridge the effectiveness of statistical classification tools with the flexibility and descriptive power of graphs. However, the algorithms used are very complex. The distance calculation performed in the graph space is itself complex.

The vector representation of graphs by an explicit graph embedding algorithm seems to be the track to meet the technical obstacles for pattern recognition. This is because of the fact that they can, in theory, combine effectively the power and flexibility of graph representation with the diversity and complexity of statistical tools. However, no efficient and simple method to adapt to the data is yet available.

By analyzing all the methods described in this chapter, we can mention the following important limitations of graph matching methods:

- Exact methods cannot deal with noise and deformations.
- Exact methods cannot deal with attributes on nodes and edges of the graph.
- Exact methods cannot deal with huge graphs.
- Error tolerant methods are time consuming.
- Error tolerant methods do not have the ability to achieve unsupervised learning.
- Error tolerant methods need training dataset.
- Error tolerant methods have the worst management of discretization step.
- The computation of distance between graphs is computational expensive.

To the best of our knowledge, most of the existing works on graph embedding can handle only the graphs which are comprised of edges with a single attribute and vertices with either no or only symbolic attributes. These methods are only useful for specific application domains for which they are designed. Our proposed method of explicit graph embedding can embed attributed graphs with many symbolic as well as numeric attributes on both nodes and edges.

Our proposed method of explicit graph embedding does not require any dissimilarity measure between graphs and proposes a natural embedding of topological, structural and attribute information of an input graph into a feature vector.

The method is applicable to undirected as well as directed attributed graphs, without imposing any restriction on the size of graphs.

We employ fuzzy overlapping trapezoidal intervals for minimizing the information loss while mapping from continuous graph space to discrete feature vector space. The use of fuzzy logic enables our method to gracefully handle noise (deformations) in graphs and to propose a robust embedding of graphs into numeric feature vectors.

Our method does not necessarily need any labeled training set and has built-in unsupervised learning abilities. In scenarios where no additional training set is available, the method has the capability to exploit the graph dataset to be embedded into feature vectors, for adapting its parameters during an off-line unsupervised learning phase. Once the learning is performed it offers real-time embedding of attributed graphs into feature vectors.

Many existing solutions for graph embedding are based on spectral analysis of the adjacency matrix of graphs and offer to utilize the statistical significant details in graphs for embedding them into feature vectors. Our method exploits the topological, structural and attribute information of the graphs along-with the statistical significant information, for constructing feature vectors. The proposed feature vector is very significant for application domains where the use of graphs is mandatory for representing rich structural and topological information, and an approximate but computational efficient solution is needed.

Our proposed method of graph embedding does not require any labeled learning set, another important extension to the existing works is the inexpensive deployment of our graph embedding method to a wide range of application domains.

We propose to extend the methods in literature by offering the embedding of directed and undirected attributed graphs with numeric as well as symbolic attributes on both nodes and edges. We propose a consistent embedding of graphs i.e. two similar graphs are mapped to points near to each other in feature vector space. Multi facet information from global, topological and local point of view seems very important to be preserved. The extraction of topological and structural level details and multilevel distribution analysis of graph are the novelty of our work.

In the rest of the dissertation we present our work on explicit graph embedding. We propose an efficient method to achieve simple and straightforward embedding of attributed graphs into feature vectors, for answering the problems of graph classification and graph clustering.

Chapter 3

Fuzzy Multilevel Graph Embedding

In this chapter we outline our explicit graph embedding method i.e. the Fuzzy Multilevel Graph Embedding (FMGE). We first present an overall global description of FMGE and then present details on the FMGE framework. We elaborate the description of FMGE by example graphs of primitive geometric shapes. We briefly introduce the application of FMGE to graph classification, graph clustering and graph retrieval.

3.1 Introduction

This chapter presents detailed description of our proposed method of explicit graph embedding i.e. the *Fuzzy Multilevel Graph Embedding*. For improving the continuity and readability of the dissertation we use the abbreviation *FMGE* for referring to the Fuzzy Multilevel Graph Embedding.

For embedding an attributed graph into a numeric feature vector, FMGE performs multilevel analysis of graph to extract discriminatory information of three different levels from a graph. This includes the graph level information, structural level information and the elementary level information. The three levels of information represent three different views of graph for extracting coarse (global) details, then going more deep to the details

on the topology of graph and then penetrating into details on the elementary building units of the graph. The multilevel analysis of a graph enables FMGE to look upon it from three different perspectives. Graph level information permits FMGE to extract general information about a graph. The structural level information allows FMGE to extract homogeneity of subgraphs in a graph. And the elementary level information allows FMGE to penetrate into more in-depth and granular view of a graph, for extracting discriminatory information from the elementary building blocks of graphs.

FMGE encodes the numeric part of each of the different levels of information by fuzzy histograms. It constructs the fuzzy histograms by employing fuzzy logic. The use of fuzzy logic permits FMGE to use smooth transition between the intervals of the histograms and enables FMGE to minimize the information loss while embedding graphs into feature vectors. FMGE encodes the symbolic part of the multilevel information of a graph by crisp histograms. Once the histograms are constructed, FMGE employs the histogram representation of the multilevel information of a graph for embedding it into a numeric feature vector.

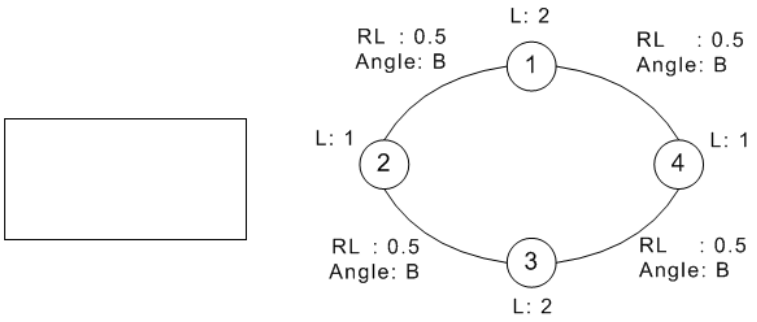
FMGE adapts the parameters for constructing the fuzzy and crisp histograms to underlying graphs by its unsupervised learning capabilities, without requiring any labeled learning set.

In this chapter we first describe the input, output and the multilevel information in the feature vector of FMGE. Afterwards, we present a detailed description of FMGE framework: we present the unsupervised learning phase of FMGE and the graph embedding phase of FMGE.

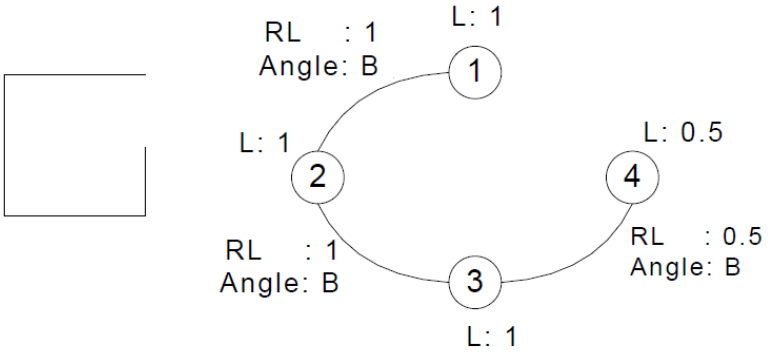
Initial versions of this work have been published in [Luqman et al., 2009b], [Luqman et al., 2009a], [Luqman et al., 2010c], [Luqman et al., 2010d] and [Luqman et al., 2010a].

To facilitate the explanation and comprehension of the description of FMGE, we have used example graphs of primitive geometric shapes, of unit length, of a rectangle, an occluded square and a triangle. The example graphs and their corresponding shapes are shown in Figure 3.1.

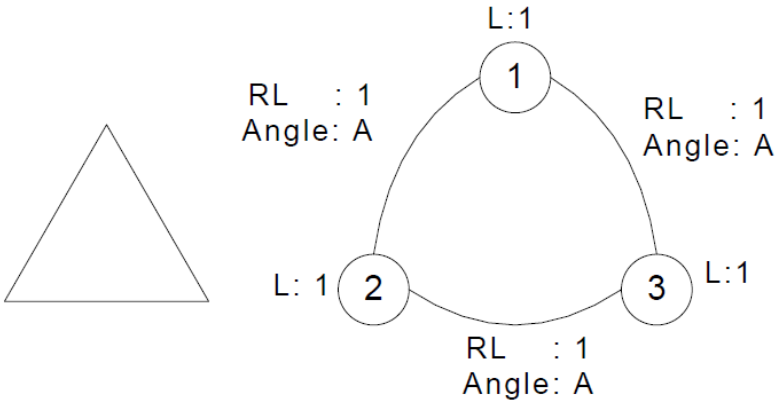
In Figure 3.1, a vertex in graph is an abstract representation of a primitive line in underlying content, with an attribute of length L. An edge in graph represents the connectivity relationship between two primitive lines in underlying content, with their relative length RL and angle $Angle$ between them as the edge attributes.



(a) A rectangle.



(b) An occluded square.



(c) A triangle.

Figure 3.1: Attributed graph representation of basic geometric shapes of unit length.

3.2 Overview of fuzzy multilevel graph embedding (FMGE)

A block diagram of FMGE is presented in Figure 3.2. It accepts a collection of m attributed graphs as input and encodes their topological, structural and attribute details into m equal size feature vectors.

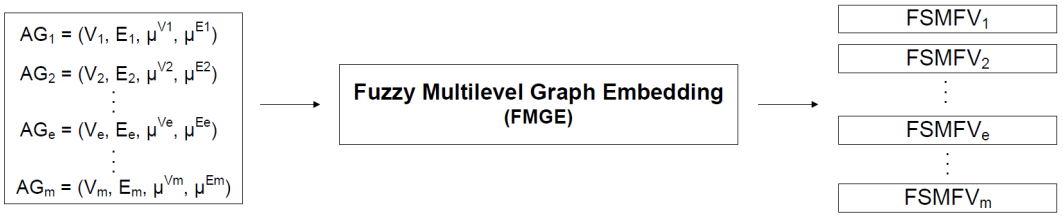


Figure 3.2: Overview of Fuzzy Multilevel Graph Embedding (FMGE).

As shown in Figure 3.2, FMGE accepts a collection of m attributed graphs as input, as given by:

$$\{AG_1, AG_2, ..., AG_e, ..., AG_m\}$$

where the e^{th} graph is denoted by:

$$AG_e = (V_e, E_e, \mu^{V_e}, \mu^{E_e}).$$

For the input collection of m attributed graphs, FMGE produces a collection of m same size feature vectors as output, as given by:

$$\{FSMFV_1, FSMFV_2, ..., FSMFV_e, ..., FSMFV_m\}$$

The e^{th} input graph AG_e is embedded into feature vector $FSMFV_e$:

$$AG_e \longmapsto \phi(AG_e) = FSMFV_e$$

where $FSMFV_e$ is a point in n dimensional vector space $\mathbb{R}^n$:

$$FSMFV_e = (f_{e_1}, f_{e_2}, ..., f_{e_n})$$

3.2.1 Description of feature vector of FMGE

We have named the feature vector of FMGE as *Fuzzy Structural Multilevel Feature Vector* and abbreviated it as *FSMFV*. It contains features extracted from three levels of information in graph:

i) Graph level information

ii) Structural level information

iii) Elementary level information

The features for graph level information represent a coarse view of graph and give a general information about the graph. These features include the graph order and graph size.

The features for structural level information represent a deeper view of graph and are extracted from the node degrees and subgraph homogeneity in graph.

The third level of information is extracted by penetrating into further depth and more granular view of graph and employing details of the elementary building blocks of graph. These features represent the information extracted from the node and edge attributes.

FSMFV is a vector in n dimensional vector space $\mathbb{R}^n$, given as:

$$FSMFV = (f_1, f_2, ..., f_n)$$

The overall structure of FSMFV is presented in Figure 3.3.

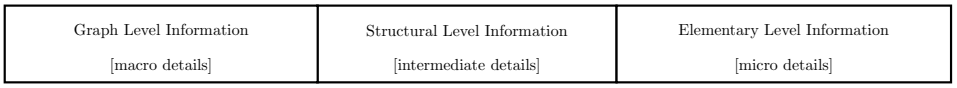


Figure 3.3: The Fuzzy Structural Multilevel Feature Vector (FSMFV).

In subsequent subsections we present detailed discussion on each of the three different levels of information in FSMFV.

3.2.1.1 Embedding of graph level information

The graph level information in FSMFV is embedded by two numeric features, encoding the order and the size of graph.

Graph order: A graph vertex is an abstract representation of the primitive components of underlying content. The order of a graph provides very important discriminatory topological information on the graph.

Graph order ($|V|$) allows to discriminate between a small graph in Figure 3.1(c) and bigger graphs in Figure 3.1(b) and Figure 3.1(a) ($|V| = 3$ vs $|V| = 4$). At the same time it permits to define a similarity between two equal ordered graphs in Figure 3.1(b) and Figure 3.1(a) ($|V| = 4$).

Graph size: An edge is an abstract representation of the relationship between the primitive components of underlying content. Graph size also provides important discriminatory information on the topological details of graph.

Two equal ordered graphs in Figure 3.1(b) and Figure 3.1(a) are differentiated by graph size ($|E|$) i.e. a thin graph in Figure 3.1(b) is differentiated from a thicker graph in Figure 3.1(a) ($|E| = 3$ vs $|E| = 4$). At the same time it permits to define a similarity between two equal sized graphs in Figure 3.1(c) and Figure 3.1(b) ($|E| = 3$).

3.2.1.2 Embedding of structural level information

The embedding of structural level information is a novelty and the most critical part of FMGE. Very few existing works on graph embedding clearly use this information. We use node degrees information and subgraph homogeneity measure embedded by histograms of node attributes resemblance and edge attributes resemblance, for embedding structural level information. Figure 3.4 outlines this part of FSMFV.

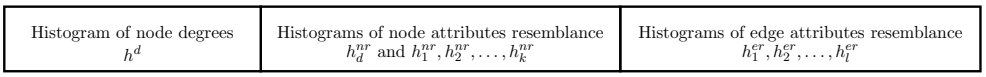


Figure 3.4: Embedding of structural level information.

Node degrees: The degrees of nodes represent the distribution of edges in graph and provide complementary discriminatory information on the structure and topology of graph. It permits to discriminate between densely connected graphs and sparsely connected graphs. Node degrees information is encoded by a histogram of s_i fuzzy intervals. The fuzzy intervals are learned during a prior learning phase, which employs degrees of all the nodes of all graphs in dataset. Node degrees features (h^d in Figure 3.4), for an attributed graph, embeds the histogram of its nodes for the s_i fuzzy intervals. In Figure 3.4, the histogram h^d is a fuzzy histogram as node degrees is a numeric information.

For directed graphs this feature is represented by two sub-features of in-degree and out-degree i.e. a fuzzy histogram for encoding the distribution of in-degrees and an other fuzzy histogram for encoding the distribution of out-degrees of nodes.

The node degree information for the graphs in Figure 3.1(c), 3.1(b) and 3.1(a) permits to increase the precision of the similarity between graphs. For example, the graphs in Figure 3.1(c) and Figure 3.1(b) represent two different topologies ($|V| = 3$ and $|V| = 4$) and a relatively similar geometry ($|E| = 3$). On the other hand, the graphs in Figure 3.1(b) and Figure 3.1(a) represent two different geometries ($|E| = 3$ and $|E| = 4$) and a quite similar topology ($|V| = 4$). Furthermore, the graphs in Figure 3.1(c) and Figure 3.1(a) represent two different topologies ($|V| = 3$ and $|V| = 4$) and geometries ($|E| = 3$ and $|E| = 4$). By incorporating the node degree information, it is very straightforward to conclude that graphs in Figure 3.1(b) and Figure 3.1(a) are more similar to each other than the graph in Figure 3.1(c).

Node attribute's resemblance for edges: The resemblance between two primitive components that have a relationship between them, in a graph, is a supplementary information available in the graph. The node attribute's resemblance for an edge encodes structural information for the respective node-couple. To compute resemblance information for an edge, the node degrees of its two nodes and the list of node attributes as given by μ^V are employed for extracting additional information. This additional information is represented as new edge attributes and is processed like other edge attributes.

Given an edge between two nodes, say $node_1$ and $node_2$ in a graph. The resemblance between a numeric node attribute a is computed by Equation 3.1 and the resemblance between a symbolic node attribute b is computed by Equation 3.2.

For each numeric node attribute, the resemblance attribute nr is represented by s_{nr} features in FSMFV. This resemblance information is encoded by a fuzzy histogram of s_{nr} fuzzy intervals. The fuzzy intervals are learned during a prior learning phase, which employs resemblance attribute nr of all the edges of all graphs in dataset.

$$resemblance(a_1, a_2) = min(|a_1|, |a_2|)/max(|a_1|, |a_2|) \tag{3.1}$$

where,
$a \in \{\text{node degree}, \mu^V\}$,
a_1 is the value of the attribute a for $node_1$ and
a_2 is the value of the same attribute a for $node_2$.

$$resemblance(b_1, b_2) = \begin{vmatrix} 1 & b_1 = b_2 \\ 0 & otherwise \end{vmatrix} \tag{3.2}$$

where,
$b \in \mu^V$,
b_1 is the value of the attribute b for $node_1$ and
b_2 is the value of the same attribute b for $node_2$.

For each symbolic node attribute, we represent the resemblance attribute nr by exactly two possible numeric features. The resemblance for symbolic attributes can either be 0 or 1 (Equation 3.2). The cardinalities of the two resemblance values in input graph, are encoded by a crisp histogram which is used as features in FSMFV.

In Figure 3.4 the histogram h_d^{nr} represents the features for encoding resemblance attribute for node degrees. Whereas, the histograms $h_1^{nr}, h_2^{nr}, \ldots, h_k^{nr}$ represent the features for encoding resemblance attributes for k node attributes μ^V. The histogram h_d^{nr} is a fuzzy histogram since node degree is a numeric information. Each of the histograms

$h_1^{nr}, h_2^{nr}, \ldots, h_k^{nr}$, is a crisp histogram if its corresponding attribute is symbolic and is a fuzzy histogram if the attribute that it is encoding is a numeric attribute.

Node attribute's resemblance for edges of example graphs is shown in Figure 3.5. Two new attributes are added to the edges of the example graphs: the resemblance attribute "resemblanceL" and the resemblance attribute "resemblanceNodeDegree". These new attributes on edges provides more precision to FMGE for discriminating between the graphs.

Edge attribute's resemblance for nodes: The resemblance among the relationships associated to a primitive component, is a supplementary information available in the graph. The edge attribute's resemblance for a node encodes the structural information for the respective edges of node and brings more topological information to FSMFV. To compute resemblance information for a node, each attribute of its edges (as given by μ^E) is employed for extracting additional information. This additional information is represented as new node attributes and is processed like other node attributes.

Given a node, say *node* in a graph, the resemblance for the edges connected to *node* is computed as the mean of the resemblances between all the pair of edges connected to *node*. For a pair of edges, say $edge_1$ and $edge_2$ connected to *node*, the resemblance for a numeric attribute c is computed by Equation 3.3 and the resemblance between a symbolic edge attribute d is computed by Equation 3.4.

$$resemblance(c_1, c_2) = min(|c_1|, |c_2|)/max(|c_1|, |c_2|) \tag{3.3}$$

where,
$c \in \mu^E$,
c_1 is the value of the attribute c for $edge_1$ and
c_2 is the value of the same attribute c for $edge_2$.

$$resemblance(d_1, d_2) = \begin{vmatrix} 1 & d_1 = d_2 \\ 0 & otherwise \end{vmatrix} \tag{3.4}$$

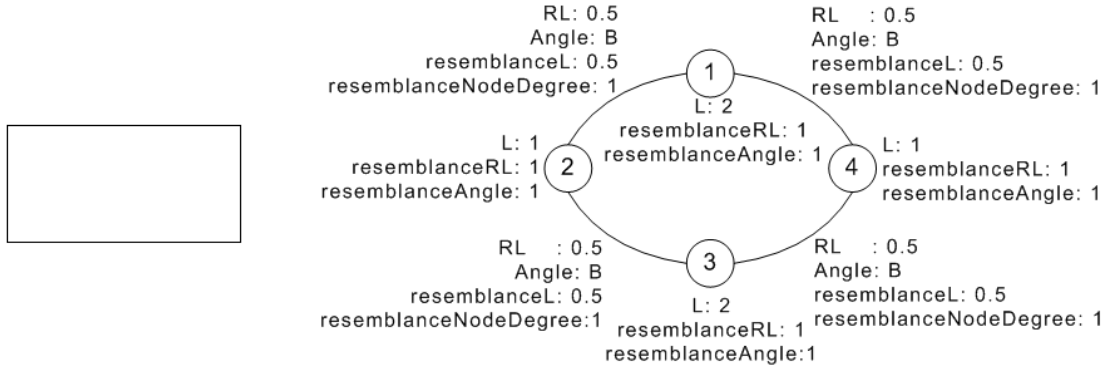
where,
$d \in \mu^E$,
d_1 is the value of the attribute d for $edge_1$ and
d_2 is the value of the same attribute d for $edge_2$.

For each symbolic edge attribute, the resulting resemblance attribute *er* is treated as a numeric attribute and is embedded by a fuzzy histogram. The resemblance for symbolic edge attributes is computed as mean of the resemblances between all the pair of edges

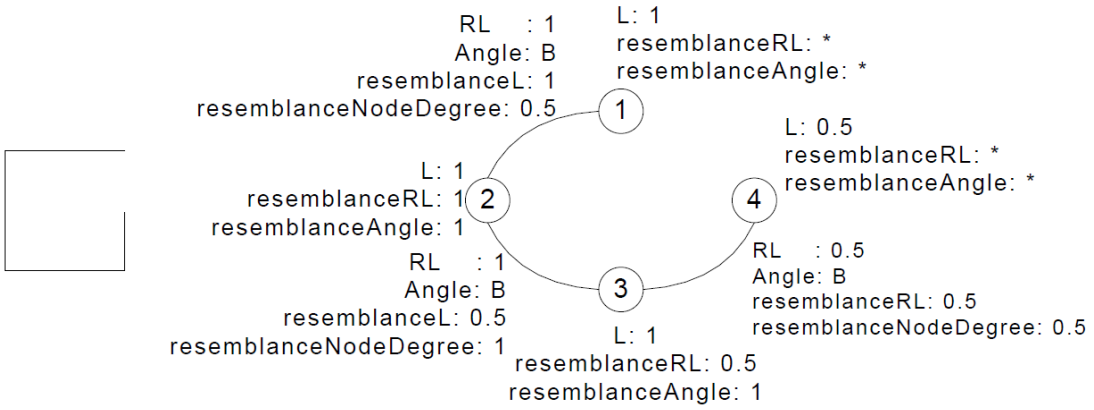
connected to a node. Although the resemblance for a pair of edges is always either 0 or 1 but mean resemblance for all the pair of edges of a node can be any numeric value (this is different from the symbolic node attribute's resemblance which is always either 0 or 1). Therefore, for each symbolic and numeric edge attribute, the resemblance attribute er is represented by s_{er} features in FSMFV. This resemblance information is encoded by a fuzzy histogram of s_{er} fuzzy intervals. The fuzzy intervals are learned during a prior learning phase, which employs resemblance attribute er of all the nodes of all graphs in dataset.

In Figure 3.4 the histograms $h_1^{er}, h_2^{er}, \ldots, h_l^{er}$ represent the features for encoding resemblance attributes for l edge attributes μ^E. Each of the histograms $h_1^{er}, h_2^{er}, \ldots, h_l^{er}$, is a fuzzy histogram.

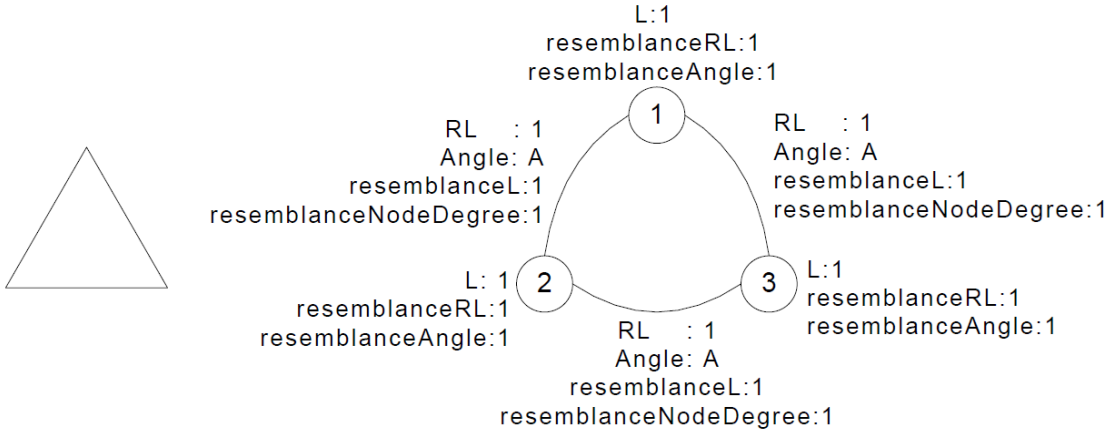
Edge attribute's resemblance for nodes of example graphs is shown in Figure 3.5. Two new resemblance attributes are added on the nodes of the example graphs: the resemblance attribute "resemblanceRL" and the resemblance attribute "resemblanceAngle". These new attributes on nodes compliments the node attribute's resemblance on edges and enables FMGE to extract information for discriminating between the graphs.



(a) Rectangle.



(b) Occluded square.



(c) Triangle.

Figure 3.5: Resemblance attributes for the attributed graph representation of basic geometric shapes of unit length.

3.2.1.3 Embedding of elementary level information

Embedding of elementary level information allows FMGE to extract discriminatory information from individual nodes and edges of the graph. The symbolic attributes are encoded by crisp histograms and the numeric attributes by fuzzy histograms. FMGE can embed attributed graphs with many symbolic and numeric attributes on both nodes and edges.

For every symbolic node attribute, each modality that can be taken by this attribute is represented by exactly one numeric feature in FSMFV. This feature encodes the cardinality of this modality in an input graph.

Each numeric node attribute is encoded by a fuzzy histogram of its s_i fuzzy intervals. The fuzzy intervals are learned for each of the numeric node attributes in the input graph dataset during a prior learning phase. The features for a numeric attribute of an input graph, embeds the histogram for these s_i fuzzy intervals.

Figure 3.6 outlines this part of FSMFV.

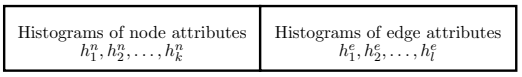


Figure 3.6: Embedding of elementary level information.

Node attributes: The node attributes provide additional details on primitive components of underlying content and aid FSMFV to discriminate between two similar structured graphs. In Figure 3.6 the histograms $h_1^n, h_2^n, \ldots, h_k^n$ represent the features for encoding k node attributes μ^V. Each of the histograms $h_1^n, h_2^n, \ldots, h_k^n$, is a crisp histogram if its corresponding attribute is symbolic and is a fuzzy histogram if the attribute that it is encoding is a numeric attribute.

The node attributes are very important information for discriminating between two equal ordered and equal sized graphs, which are representing quite similar structure as well (for example the graph of a small square can be differentiated from that of a big square by using a length attribute on the nodes).

The length attribute L of the graph nodes in Figure 3.5(a) provide additional details on the graph and clearly discriminates it from the graph in Figure 3.5(b).

Edge attributes: The edge attributes provide supplementary details on the relationships between the primitive components of the underlying content and aid FSMFV to discriminate between two similar structured graphs. In Figure 3.6 the histograms $h_1^e, h_2^e, \ldots, h_l^e$ represent the features for encoding l edge attributes μ^E. Each of the histograms $h_1^e, h_2^e, \ldots, h_l^e$, is a crisp histogram if its corresponding attribute is symbolic and is a fuzzy histogram if the attribute that it is encoding is a numeric attribute.

The edge attributes are very important information for discriminating between two equal ordered and equal sized graphs, which are representing quite similar structure as well (for example the graph of a square can be differentiated from that of a rhombus by using angle attribute on the edges).

The relative length RL and angle A attribute of the graph edges in Figure 3.5(b) provide additional details on the graph and clearly discriminates it from the graph in Figure 3.5(c).

3.3 Framework of fuzzy multilevel graph embedding (FMGE)

In FMGE framework, the mapping of input collection of graphs to appropriate points in a suitable vector space $\mathbb{R}^n$ is achieved in two phases i.e. the off-line unsupervised learning phase and the on-line graph embedding phase.

The unsupervised learning phase learns a set of fuzzy intervals for features linked to distribution analysis of the input graphs i.e. features for node degree, numeric node and edge attributes and the corresponding resemblance attributes. We refer each of them as an attribute i. For symbolic node and edge attributes and the corresponding resemblance attributes, the unsupervised learning phase employs the modalities taken by this attribute and treat them as crisp intervals.

The graph embedding phase employs these intervals for computing different bins of the respective fuzzy or crisp histograms.

3.3.1 Unsupervised learning phase

The off-line unsupervised learning phase of FMGE is outlined in Figure 3.7. It learns a set of fuzzy intervals for encoding numeric information and crisp intervals for encoding symbolic information in graphs.

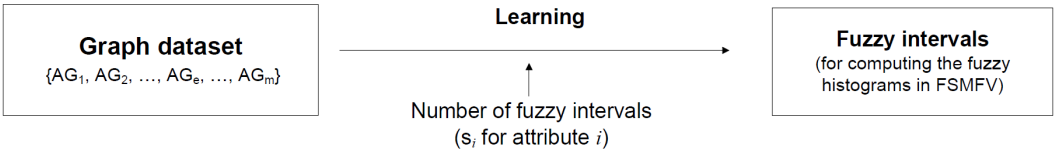


Figure 3.7: The unsupervised learning phase of FMGE.

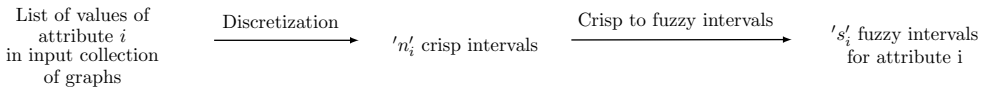


Figure 3.8: Learning fuzzy intervals for an attribute i.

3.3.1.1 Input and output

The input to unsupervised learning phase of FMGE is the collection of m attributed graphs, given by:

$$\{AG_1, AG_2, ..., AG_e, ..., AG_m\}$$

where the e^{th} graph is denoted by:

$$AG_e = (V_e, E_e, \mu^{V_e}, \mu^{E_e})$$

A second input to this phase, could be if necessary, for each feature the desired number of fuzzy intervals. This is referred by s_i for an attribute i. Some methods of discretization are able to find the "optimal" number of intervals by themselves (for example, equal frequency bins, as discussed in next subsection).

As output the unsupervised learning phase of FMGE produces s_i fuzzy overlapping trapezoidal intervals for an attribute i in input graphs (example in Figure 3.9).

3.3.1.2 Description

The main steps for learning of fuzzy intervals for an attribute i are outlined in Figure 3.8 and are explained in subsequent paragraphs.

The first step is the computation of crisp intervals from the list of values of attribute i for all the graphs in input collection of graphs. This is straightforward and is achieved by any standard data discretization technique. A survey of popular discretization techniques is presented by Liu et al. [Liu et al., 2002].

We propose to use equally spaced bins for obtaining an initial set of crisp intervals, as they avoid over-fitting and offers FMGE a better generalization capability to unseen graphs during graph embedding phase. Algorithm 3.3.1 outlines the pseudo-code for computing an initial set of crisp intervals for an attribute i. It uses a pseudo-call '$GetEqualSpacBin$'

for getting an initial set of equally spaced bins. This pseudo-call refers to an appropriate data discretization function (available in underlying implementation platform).

Another possibility is to use the equal frequency bins. An example of this type of discretization is the technique proposed by [Colot et al., 1994] for discretization of continuous data. It is based on use of Akaike Information Criterion (AIC). It starts with an initial histogram of data and finds optimal number of bins for underlying data. The adjacent bins are iteratively merged using an AIC-based cost function until the difference between AIC-beforemerge and AIC-aftermerge becomes negative. Thus, we get an optimal set of crisp intervals for the underlying data. This set of intervals is representative of the underlying data. The use of an equal frequency discretization technique is demonstrated in Section 5.4 for graph clustering.

The initial set of equally spaced bins (or equal frequency bins) are used to construct a data structure, which stores the crisp intervals. This data structure is employed for computing s_i fuzzy intervals for attribute i.

After computing the initial set of crisp intervals, in next step, these crisp intervals are arranged in an overlapping fashion to get fuzzy overlapping intervals. Normally trapezoidal, triangular and Gaussian arrangements are popular choices for fuzzy membership functions [Ishibuchi and Yamamoto, 2003]. We propose to use the trapezoidal membership function, which is the generally used fuzzy membership function. It allows a range of instances to lie under full membership and assigns partial membership to the boundary instances.

Figure 3.9 outlines a trapezoidal interval defined over crisp intervals. A trapezoidal interval is defined by four points; as is given by points a, b, c, d in Figure 3.9.

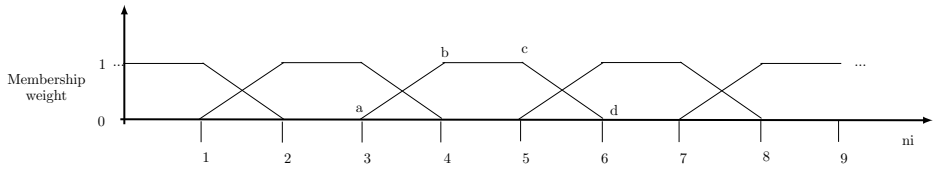


Figure 3.9: 5 fuzzy overlapping trapezoidal intervals (s_i) defined over 9 equally spaced crisp intervals (n_i).

It is very important to highlight here that the first fuzzy overlapping trapezoidal interval covers all values till $-\infty$ and the last fuzzy overlapping trapezoidal interval is limited by ∞. This makes sure that during the graph embedding phase every attribute instance falls under the range of fuzzy overlapping trapezoidal intervals and it further strengthens the generalization abilities of the method to unseen graphs.

A fuzzy interval defined in trapezoidal fashion assigns a membership weight of 1 (full membership) between points b and c. The membership weight gradually approaches 0 as we move from b to a and from c to d. This trapezoidal behavior allows to assign full membership, partial membership and no membership to attribute instances. This is important to highlight here that the total membership assigned to an instance is always exactly equal to 1 i.e. either one full membership or two partial memberships are assigned to each attribute instance.

Algorithm 3.3.2 outlines the pseudo-code for computing fuzzy overlapping trapezoidal intervals from an initial set of crisp intervals for an attribute i in input collection of graphs. It first computes the initial set of crisp intervals using Algorithm 3.3.1 and then arranges them in an overlapping trapezoidal fashion for obtaining a set of fuzzy overlapping trapezoidal intervals for attribute i. The number of fuzzy intervals for attribute i depends upon the number of features desired for attribute i in FSMFV, and is controlled by parameter s_i which can either be manually specified, automatically learned by using an equal frequency based discretization or empirically learned and optimized on validation set.

The number of crisp intervals n_i for desired number of fuzzy intervals s_i is computed by Equation 3.5.

$$n_i = 2 \times s_i - 1 \tag{3.5}$$

For the sake of continuity and readability, we have used the terms fuzzy intervals, fuzzy overlapping intervals and fuzzy overlapping trapezoidal intervals interchangeably, in this dissertation.

Algorithm 3.3.1: GETINITCRISPINTERVAL($listvaluesAttribute_i, n_i$)

comment: Computes equally spaced crisp intervals.

comment: Requires: List of values of an attribute i

comment: Requires: Number of crisp intervals for attribute i ($n_i \geq 2$)

comment: Returns: 'n_i' crisp intervals for attribute i

$equallySpacedBins \leftarrow$ GETEQUALSPACBIN($listvaluesAttribute_i, n_i$)

$crispIntervals \leftarrow empty$
$st \leftarrow -\infty$
$en \leftarrow equallySpacedBins[1]$

$j \leftarrow 1$
repeat
- $crispIntervals[j].start \leftarrow st$
- $crispIntervals[j].end \leftarrow en$
- $st \leftarrow en$
- $en \leftarrow equallySpacedBins[j+1]$
- $j \leftarrow j+1$

until $j > n_i$

return ($crispIntervals$)

Algorithm 3.3.2: GETFUZZYOVLAPTRAPZINTERVAL($listvaluesAttribute_i, s_i$)

comment: Computes fuzzy intervals for an attribute i.

comment: Requires: List of values of an attribute i

comment: Requires: Number of fuzzy intervals for attribute i ($s_i \geq 2$)

comment: Returns: 's_i' fuzzy intervals for attribute i

$n_i \leftarrow 2 * s_i - 1$
$crispIntervals \leftarrow$ GETINITCRISPINTERVAL($listvaluesAttribute_i, n_i$)

$fuzzyIntervals \leftarrow empty$

$fuzzyIntervals[1].a \leftarrow -\infty$
$fuzzyIntervals[1].b \leftarrow -\infty$
$fuzzyIntervals[1].c \leftarrow crispIntervals[1].end$
$fuzzyIntervals[1].d \leftarrow crispIntervals[2].end$

$j \leftarrow 1$
$jcrisp \leftarrow 0$
repeat
  $j \leftarrow j + 1$
  $jcrisp \leftarrow jcrisp + 2$
  $fuzzyIntervals[j].a \leftarrow fuzzyIntervals[j-1].c$
  $fuzzyIntervals[j].b \leftarrow fuzzyIntervals[j-1].d$

  if $(jcrisp + 1 \geq n_i)$
    then $fuzzyIntervals[j].c \leftarrow \infty$
             $fuzzyIntervals[j].d \leftarrow \infty$
             $break$

  $fuzzyIntervals[j].c \leftarrow crispIntervals[jcrisp + 1].end$
  $fuzzyIntervals[j].d \leftarrow crispIntervals[jcrisp + 2].end$
until $j \geq s_i$

return $(fuzzyIntervals)$

3.3.2 Graph embedding phase

The graph embedding phase of FMGE is outlined in Figure 3.10. It employs the crisp intervals and fuzzy intervals (from unsupervised learning phase) to compute respective histograms for embedding an input attributed graph into a feature vector. This achieves the mapping of the input graphs to appropriate points in a suitable vector space $\mathbb{R}^n$.

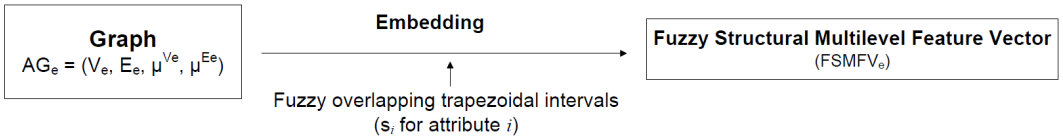


Figure 3.10: The graph embedding phase of FMGE.

3.3.2.1 Input and output

The input to graph embedding phase of FMGE is an attributed graph AG_e to be embedded, as given by:

$$AG_e = (V_e, E_e, \mu^{V_e}, \mu^{E_e})$$

A second input to this phase is the s_i fuzzy overlapping trapezoidal intervals for the attribute i (from learning phase).

The graph embedding phase of FMGE produces a feature vector $FSMFV_e$ for input graph AG_e. The length of this feature vector is uniform for all graphs in an input collection and is given by Equation 3.6.

$$\text{Length of FSMFV} = 2 + \sum s_i + \sum c_i \tag{3.6}$$

where,

- 2 refers to the features for graph order and graph size,
- $\sum s_i$ refers to the sum of number of bins in fuzzy interval encoded histograms for numeric information in graph (i.e. attribute i),
- $\sum c_i$ refers to the sum of number of bins in crisp interval encoded histograms for symbolic information in graph.

The length of the feature vector is strictly dependent on the size of histograms used for encoding the three levels of information. In order to give an idea on the maximum length of the feature vector, we can safely precise that its upper limit is 150.

3.3.2.2 Description

The values of attribute i in input graph AG_e are fuzzified by employing its s_i fuzzy intervals (learned during unsupervised learning phase) and trapezoidal membership function.

Mathematically, the membership function α defined over a trapezoidal interval x is given by Equation 3.7.

$$\alpha(x) = \begin{vmatrix} (x-a)/(b-a) & a \leq x < b \\ 1 & b \leq x \leq c \\ (x-d)/(c-d) & c < x \leq d \\ 0 & otherwise \end{vmatrix} \tag{3.7}$$

In Equation 3.7, x refers to an instance of attribute i to be fuzzified and a, b, c, d refers to the limits of a trapezoidal fuzzy interval for attribute i. Function $\alpha(x)$ computes the degree of membership of an instance x with the trapezoidal interval defined by a, b, c, d. The possible memberships can be *a full membership* if x is between b and c, *a partial membership* if x is between a and b or is between c and d, or it can be *a no membership* if x is outside the interval a, b, c, d.

We represent the fuzzy histogram of attribute i as h_i^{num}, which actually refers to the fuzzy histogram for node degrees (h^d in Figure 3.4), the fuzzy histograms for numeric node attributes resemblance (h_d^{nr} and $h_1^{nr}, h_2^{nr}, \ldots, h_k^{nr}$ in Figure 3.4), the fuzzy histograms for numeric and symbolic edge attributes resemblance ($h_1^{er}, h_2^{er}, \ldots, h_l^{er}$ in Figure 3.4), the

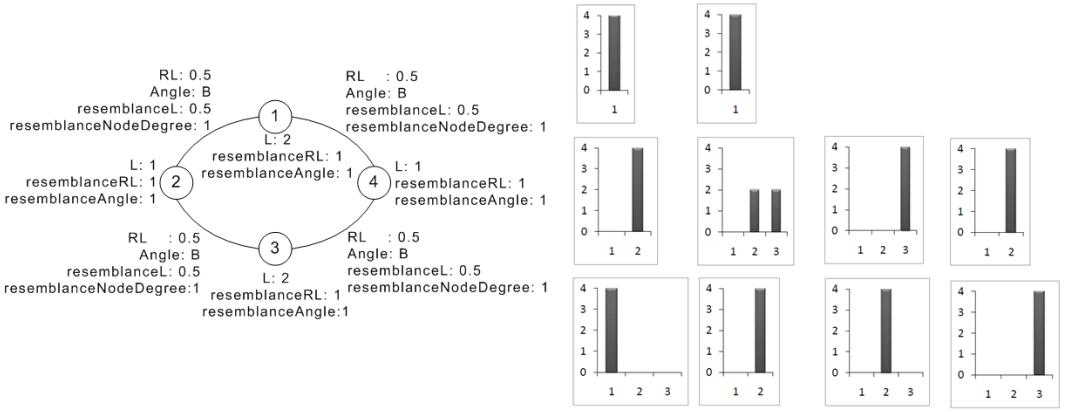
fuzzy histograms for numeric node attributes ($h_1^n, h_2^n, \ldots, h_k^n$ in Figure 3.6) and the fuzzy histograms for numeric edge attributes ($h_1^e, h_2^e, \ldots, h_l^e$ in Figure 3.6). The fuzzy histogram of an attribute i represents the embedding of attribute i for input graph AG_e. For an input graph AG_e, the fuzzy histogram h_i^{num} for attribute i is constructed by first computing the degree-of-memberships of all instances of attribute i in AG_e for the s_i fuzzy intervals. And then summing the memberships for each of the s_i fuzzy interval.

Each symbolic attribute j in input graph AG_k is encoded by a histogram of all its possible modalities (or labels). This histogram encodes the number of instances for each possible label of the symbolic attribute. We call this histogram as a crisp histogram (in-contrary to fuzzy histogram for numeric attributes). We call a crisp histogram for a symbolic attribute j as h_j^{sym}, which actually referrers to the crisp histograms for symbolic node attributes resemblance ($h_1^{nr}, h_2^{nr}, \ldots, h_k^{nr}$ in Figure 3.4), the crisp histograms for symbolic node attributes ($h_1^n, h_2^n, \ldots, h_k^n$ in Figure 3.6) and the crisp histograms for symbolic edge attributes ($h_1^e, h_2^e, \ldots, h_l^e$ in Figure 3.6).

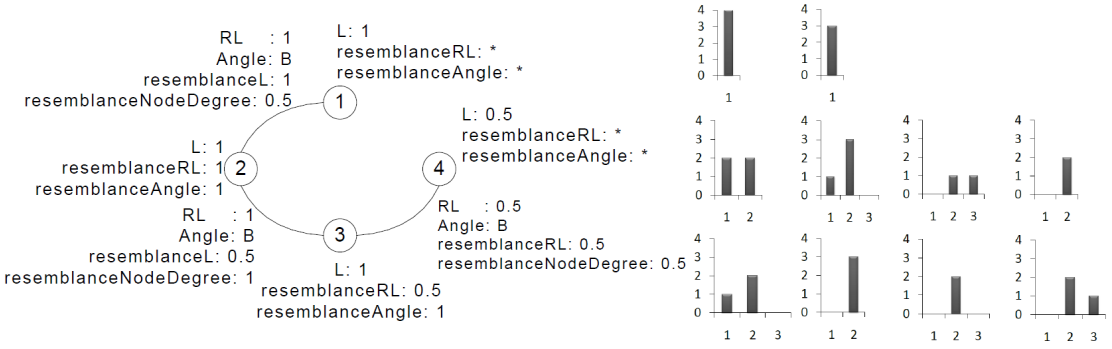
After constructing the fuzzy interval encoded histograms for each numeric attribute i and crisp histogram for each symbolic attribute j, the $FSMFV_e$ for input graph AG_e is constructed from the value of graph order, the value of graph size and fuzzy interval encoded histograms h_i^{num} of its node degree, numeric node and edge attributes appended by the crisp histograms h_j^{sym} for symbolic node and edge attributes. This gives an embedding of the input graph AG_e into a feature vector $FSMFV_e$.

The histogram illustration of the graph embedding phase and the resulting feature vectors for the example attributed graphs of basic geometric shapes are given in Figure 3.11. *Two* fuzzy trapezoidal intervals are used for embedding node degree ([$-\infty$,$-\infty$,1,2] and [1,2,∞,∞]) whereas *three* fuzzy trapezoidal intervals are used for node attribute 'L' and edge attribute 'RL' ([$-\infty$, $-\infty$, 0.5, 1], [0.5, 1, 1.5, 2] and [1.5, 2, ∞, ∞]) and *three* fuzzy intervals are used for embedding numeric resemblance attributes ([$-\infty$, $-\infty$, 0.25, 0.5], [0.25, 0.5, 0.75, 1.0], [0.75, 1.0, ∞, ∞]). *Two* fuzzy intervals are used for embedding resemblance attribute for 'Angle' ([$-\infty$, $-\infty$, 0, 1.0], [0, 1.0, ∞, ∞]). The symbolic edge attribute 'Angle' has two possible labels. Thus, in total the FSMFV for these graphs is comprised of 23 features (1 for graph order, 1 for graph size, 2 for node degree, 3 for node attribute 'L', 3 for resemblance on edge attribute 'RL', 2 for resemblance on edge attribute 'Angle', 3 for edge attribute 'RL', 2 for edge attribute 'Angle', 3 for resemblance on 'L', 3 for resemblance on 'nodeDegree').

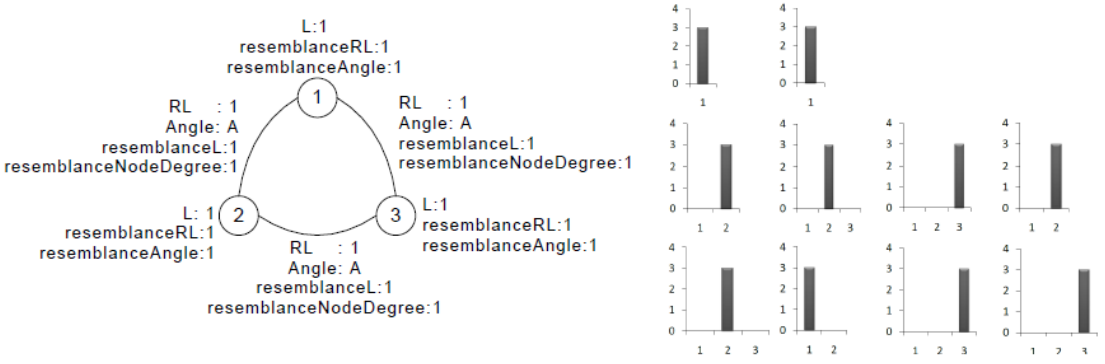
In Figure 3.11, the histograms in first row for each graph, present the histograms for graph order and graph size (left to right). The center row presents histograms for node degree, node attributes L, $resemblanceRL$ and $resemblanceAngle$ respectively from left to right. The last row presents the histograms for edge attributes RL, $Angle$, $resemblanceL$ and $resemblanceNodeDegree$ respectively from left to right.



(a) $|V|$=4, $|E|$=4, $edgeAtt$=$\{\{.5,.5,.5,.5\},\{B,B,B,B\},\{.5,.5,.5,.5\},\{1,1,1,1\}\}$, $nodeDegree$=$\{2,2,2,2\}$, $nodeAtt$=$\{\{2,1,2,1\},\{1,1,1,1\},\{1,1,1,1\}\}$ and
$FSMFV = 4,4,0,4,0,2,2,0,0,4,0,4,4,0,0,0,4,0,4,0,0,0,4$



(b) $|V|$=4, $|E|$=3, $edgeAtt$=$\{\{1,1,0.5\},\{B,B,B\},\{1,0.5,0.5\},\{0.5,1,0.5\}\}$, $nodeDegree$=$\{1,2,2,1\}$, $nodeAtt$=$\{\{1,1,1,0.5\},\{*,1,0.5,*\},\{*,1,1,*\}\}$ and
$FSMFV = 4,3,2,2,1,3,0,0,1,1,0,2,1,2,0,0,3,0,2,0,0,2,1$



(c) $|V|$=3, $|E|$=3, $edgeAtt$=$\{\{1,1,1\},\{A,A,A\},\{1,1,1\},\{1,1,1\}\}$, $nodeDegree$=$\{2,2,2\}$, $nodeAtt$=$\{\{1,1,1\},\{1,1,1\},\{0,0,0\}\}$ and
$FSMFV = 3,3,0,3,0,3,0,0,0,3,0,3,0,3,0,3,0,0,0,3,0,0,3$

Figure 3.11: Histogram encoding of information and Fuzzy Structural Multilevel Feature Vectors for the example attributed graphs.

3.4 Conclusion

In this chapter we have presented a method of explicit graph embedding - the Fuzzy Multilevel Graph Embedding (FMGE). The method is a straightforward, simple and computational efficient solution for facilitating the use of graph based powerful representations together with learning and computational strengths of state-of-the-art machine learning, classification and clustering. The method exploits multilevel analysis of graphs for extracting three levels of information from graphs (i.e. graph level, structural level and elementary level) and embedding them into numeric feature vectors. The method offers the embedding of attributed graphs with numeric as well as symbolic attributes on both nodes and edges. It has built-in learning abilities for adapting its parameters to underlying graph repositories.

However, the method is strongly dependent on the attributes of the nodes and edges in the graph. Apart, from the elementary level details, the structural level details, which are extracted by the homogeneity of subgraphs in the graph, are also defined in terms of the node and edge attributes. The proposed feature vector lacks in information on the topology of the graph. This makes this method very useful for the application domains which extract attribute rich graphs from data - i.e. there are lots of meaningful attributes. The method is less useful for application domains where graphs have less number of attributes and topological details (without using attributes) are required to be extracted for graph embedding.

FMGE is very useful for addressing the problems of graph classification, graph clustering and graph retrieval for large graph repositories.

The classification of a query graph to one of the graph classes is an important problem domain of pattern recognition and is very interesting for graphics recognition, object recognition and shape classification. For classification problems, generally, labeled learning and validation sets (with class information) are available. FMGE is fully capable of exploiting the learning set for adapting its parameters, for histogram construction, to underlying graphs and exploiting the validation set for optimizing these parameters and selecting the best parameters for embedding underlying graphs. For such problem domains the use of equal space bins permits FMGE to train on learning set, validate its parameters on validation set and generalize to unseen graphs (in test set). Application of FMGE to classification problem is further studied in Section 5.3 of the dissertation.

The clustering of a collection of graphs in groups (or clusters) is another very important problem domain of pattern recognition, which is very interesting for real application domains where the manual labeling of data is not possible or is expensive and an automatic (or semi-automatic) solution is desired. For clustering problems, generally, no labeled learning or validation sets are available. The goal of such problems is to identify groups of graphs in a given collection of graphs. FMGE is fully capable of solving such problems. It exploits the (same) collection of graphs during a first pass, for automatically adapting

its histogram construction parameters to underlying graph collection. And then during a second pass it embeds the graphs in the collection by using the learned parameters for histogram construction. For such problem domains the use of an equal frequency bins permits FMGE to over-fit its parameters to the underlying graphs and to offer the best possible partitioning of graphs into clusters.

For clustering problems, a second possible way of adapting the histogram construction parameters of FMGE is by exploiting the domain knowledge. This semi-automatic way allows a domain expert to manually adjust the number of intervals for (some or all) histograms of the multilevel information. FMGE then uses these parameters for embedding the graphs into numeric feature vectors. Application of FMGE to clustering problem is further studied in Section 5.4 of the dissertation.

Graph retrieval and subgraph spotting is an other widely employed problem domain of pattern recognition. It is useful for application domains where automatic indexing is desired for graph repositories. The use of FMGE not only offers automatic indexing of graph repositories but also the ease of query by example (QBE) and the granularity of focused retrieval. In next chapter we present details on the use of FMGE for building an unsupervised framework for such application domains.

Chapter 4

Graph retrieval and subgraph spotting through explicit graph embedding

---

In this chapter we present a framework for automatic indexing of attributed graph repositories. We demonstrate a practical application of Fuzzy Multilevel Graph Embedding (FMGE) together with classification and clustering tools, for achieving graph retrieval and subgraph spotting.

4.1 Introduction

This chapter presents detailed description on our proposed framework for automatic indexing of graph repositories, for performing graph retrieval and subgraph spotting. Graph retrieval refers to the problem of retrieving a graph from a graph repository, based on its similarity with an example (or query) graph. Whereas, subgraph spotting takes the graph retrieval to further granularity and refers to the problem of retrieving a graph from a repository of graphs, based on the similarity of its subgraph with the example (or query) graph.

In our proposed framework the subgraph spotting is achieved through explicit graph

embedding. The framework employs FMGE as the graph embedding method and it works in two phases: during an off-line unsupervised learning phase it constructs an index for the graph repository and during an on-line spotting phase it exploits the index for achieving subgraph spotting.

The automatic indexing of a graph repository is achieved during off-line learning phase, where we:

i) break the graphs into 2-node subgraphs (a.k.a. cliques of order 2), which are primitive building-blocks of a graph,

ii) embed the 2-node subgraphs into feature vectors by employing our recently proposed explicit graph embedding technique,

iii) cluster the feature vectors in classes by employing a classic agglomerative clustering technique,

iv) build an index for the graph repository and

v) learn a Bayesian network classifier.

The subgraph spotting is achieved during the on-line querying phase, where we:

i) break the query graph into 2-node subgraphs,

ii) embed them into feature vectors,

iii) employ the Bayesian network classifier for classifying query 2-node subgraphs and

iv) retrieve the respective graphs by looking-up in the index of the graph repository.

The graphs containing all query 2-node subgraphs form the set of result graphs for the query. Finally, we employ the adjacency matrix of each result graph along-with a score function, for spotting the query graph in it.

The method is equally applicable to a wide range of domains, offering ease of query by example (QBE) and granularity of focused retrieval.

In this chapter we present a detailed description of the two phases of our subgraph spotting framework. To facilitate the explanation and comprehension of the presented material, we have used the terms clique of order 2 and 2-node subgraph interchangeably as appropriate.

An initial version of this work has been published in [Luqman et al., 2010b]. A complete version of the work is published in [Luqman et al., 2011b] and [Luqman et al., 2012].

4.2 Automatic indexing of a graph repository

The automatic indexing of a graph repository is performed during the off-line unsupervised learning phase. This phase is outlined in Figure 4.1 and is explained in succeeding paragraphs.

The input to the off-line unsupervised learning phase is a collection of graphs, given by:

$$\{AG_1, AG_2, ..., AG_k, ..., AG_n\}$$

where, the k^{th} graph is:

$$AG_k = (V_k, E_k, \mu^{V_k}, \mu^{E_k})$$

First of all, these graphs are preprocessed and new resemblance attributes are added to nodes and edges of the graphs. These resemblance attributes incorporate information on the homogeneity in neighborhood of nodes and edges. The way these attributes are computed is described in section 3.2.1.2.

The next step of off-line unsupervised learning phase, extracts cliques of order 2 (i.e. the 2-node subgraphs) from all graphs in the input collection of graphs.

Our choice of extracting the cliques of order 2 is partly because of the fact that a clique of order 2 is the basic building block of a graph (with some structural information). And partly because the extraction of cliques of higher order (i.e. ≥ 2) is computational expensive, whereas extraction of cliques of order 2 is simple and straight forward. It is achieved by a mere look-up in the adjacency matrix of graph. The set of cliques for the input collection of graphs is given by:

$$\{\{subAG_1\}, \{subAG_2\}, ..., \{subAG_k\}, ..., \{subAG_n\}\}$$

where the set of subgraphs for k^{th} input graph (AG_k) is:

$$subAG_k = \{subAG_k^1, subAG_k^2, ..., subAG_k^i\}$$

and i^{th} 2-node subgraph for k^{th} input graph (AG_k) is:

$$subAG_k^i = (V_k^i, E_k^i, \mu^{V_k^i}, \mu^{E_k^i})$$

The next step in off-line unsupervised learning phase of our method, embeds the set of 2-node subgraphs by equal size feature vectors (the FSMFVs). This is achieved by Fuzzy Multilevel Graph Embedding (FMGE). The set of FSMFVs for the input collection of graphs is given by:

$$\{\{FSMFV_1\}, \{FSMFV_2\}, ..., \{FSMFV_k\}, ..., \{FSMFV_n\}\}$$

where, set of feature vectors for the 2-node subgraphs of k^{th} input graph (AG_k) is:

$$FSMFV_k = \{FSMFV_k^1, FSMFV_k^2, ..., FSMFV_k^i\}$$

and the i^{th} 2-node subgraph for k^{th} input graph (AG_k) is embedded by feature vector $FSMFV_k^i$.

The feature vectors are clustered into classes by an agglomerative (also called hierarchical) clustering method. The clustering process starts by computing a pairwise city-block-distance metric for the features in FSMFV and builds a linkage tree using the single link algorithm. We use a method from [Okada et al., 2005] for getting an optimal cutoff for clusters - [Okada et al., 2005] is based on an econometric approach to verify that variables in multiple regression are linearly independent.

The use of agglomerative clustering approach keeps our method free of any parameter for number of clusters. Each cluster represents (a class of) similar 2-node subgraphs. The clustering step of (off-line unsupervised) learning phase assigns cluster labels to the FSMFVs of 2-node subgraphs. The cluster labels for set of cliques ($FSMFV_k$) in graph AG_k are given as:

$$\{label\_FSMFV_k^1, label\_FSMFV_k^2, ..., label\_FSMFV_k^i\}$$

where, $label\_FSMFV_k^i$ represents the cluster label assigned to the FSMFV of the i^{th} clique in graph AG_k.

The labeled FSMFVs are employed for learning a Bayesian network classifier. This classifier serves as a computational efficient tool for recognizing the 2-node subgraphs in query subgraph, during the subgraph spotting phase of our method. All state of the art classifiers are equally qualified to be used in place of the Bayesian network classifier. Our choice of using a Bayesian network classifier is primarily motivated by the fact that

the probabilistic reasoning of Bayesian networks enables our framework to handle the uncertainties in the feature vector representation of the cliques in graphs.

Finally, as a last step of (off-line unsupervised) learning phase of our method, an index is constructed for the input collection of graphs. This index maps a graph to the cliques of order 2 in it. And the latter to the cluster labels. For a graph AG_k, this is given by:

$$AG_k \quad \textbf{vs} \quad subAG_k = \{subAG_k^1, subAG_k^2, ..., subAG_k^i\}$$

$$\textbf{vs} \quad \{FSMFV_k^1, FSMFV_k^2, ..., FSMFV_k^i\} \quad \textbf{vs}$$

$$\{label\_FSMFV_k^1, label\_FSMFV_k^2, ..., label\_FSMFV_k^i\}$$

To facilitate the maintenance of information and ease of querying we propose to use a standard DBMS for storing the index. Given a cluster label c, the index permits to retrieve the list of cliques (which are assigned cluster label c) and the corresponding graphs.

The use of index is further elaborated in next section.

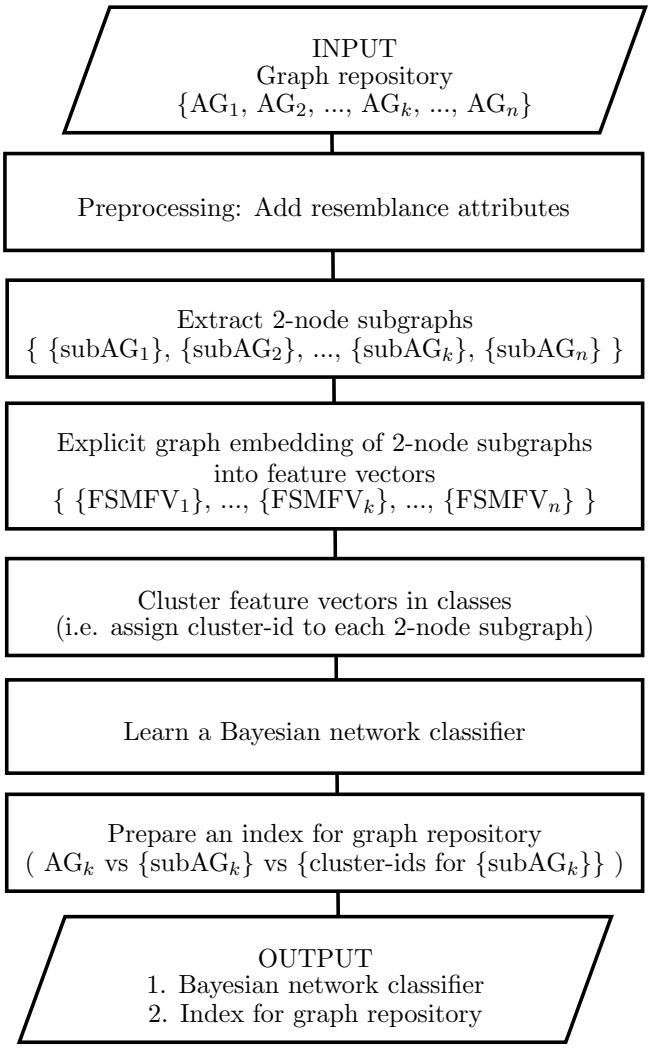


Figure 4.1: Automatic indexing of a graph repository.

4.3 Subgraph spotting

The subgraph spotting is performed during on-line querying phase. The goal of subgraph spotting phase is to retrieve the set of graphs based on a query graph AG_q, and to highlight the subgraphs in the retrieved graphs which are similar to AG_q. This phase is outlined in Figure 4.3 and is detailed in this section.

The input to on-line querying phase is a query graph comprising of at-least one clique of order 2. As detailed in previous section for automatic indexing of graph repository the query graph (AG_q) also passes through the steps of preprocessing and extraction of cliques of order 2, as given by:

$$subAG_q = \{subAG_q^1, subAG_q^2, ..., subAG_q^i\}$$

and i^{th} 2-node subgraph for query graph (AG_q) is:

$$subAG_q^i = (V_q^i, E_q^i, \mu^{V_q^i}, \mu^{E_q^i})$$

The next step is the embedding of the cliques of order 2 in query graph into feature vectors ($\{FSMFV_q\}$), as given by:

$$FSMFV_q = \{FSMFV_q^1, FSMFV_q^2, ..., FSMFV_q^i\}$$

and the i^{th} clique of order 2 in the query graph (AG_q) is embedded by feature vector $FSMFV_q^i$.

During the next step of on-line querying phase the Bayesian network classifier is employed to classify FSMFV of each query 2-node subgraph ($\{subAG_q\}$) as belonging to one of the cluster labels.

For each query 2-node subgraph, the Bayesian network classifier outputs a list of posterior-probabilities for all clusters. The query 2-node subgraph is classified as highest posterior-probability cluster.

We look-up this cluster label in repository index for getting a list of possible combinations of 2-node subgraphs corresponding to query. The graphs containing all the query 2-node subgraphs ($\{subAG_q\}$) form the set of result graphs for the query graph AG_q.

The result graph set is given by:

$$\{result\_AG_1, result\_AG_2, ..., result\_AG_k, ..., result\_AG_m\}$$

where, the k^{th} result graph is:

$$result\_AG_k = (V_k, E_k, \mu^{V_k}, \mu^{E_k})$$

For spotting the query graph AG_q in a result graph $result\_AG_k$, we employ the adjacency matrix of graph $result\_AG_k$ along-with a score function.

For two nodes "i" and "j", the possible values in the adjacency matrix are summarized in Equation 4.1. The adjacency matrix of graph $result\_AG_k$ has a value of "0" if there is no edge between "i" and "j" in the original graph $result\_AG_k$, a value of "1" if there is an edge between "i" and "j" in the original graph $result\_AG_k$ and a value of "2" if one of the query 2-node subgraphs is classified (by Bayesian network) as belonging to the cluster of this 2-node subgraph (which is comprising of edge between "i" and "j").

$$result\_AG_k(i,j) = \begin{vmatrix} 0 & \text{no edge between i and j} \\ 1 & \text{an edge between i and j} \\ 2 & \text{2-node subgraph in result} \end{vmatrix} \tag{4.1}$$

The query graph AG_q is finally spotted in the result graph $result\_AG_k$, by looking up in the neighborhood of each 2-node subgraph of $result\_AG_k$ which is in the result i.e. each "$AG_k(i,j) = 2$" in the adjacency matrix of result graph $result\_AG_k$.

We explore "w" connected neighbors of each "$result\_AG_k(i,j) = 2$". The parameter "w" is proportional to the graph order of query graph AG_q ($|V_q|$). This ensures that the retrieved graph contains a subgraph proportional in size to query graph.

We compute a score for each "$result\_AG_k(i,j) = 2$" using Equation 4.2.

$$score = \sum_{z=0}^{2}(z \times \frac{|z|}{w}) \tag{4.2}$$

In Equation 4.2,
z is a value in the adjacency matrix (either 0,1 or 2),
$|z|$ is number of times the value z occurs in neighborhood
and
w is number of connected neighbors that are looked-up.

For a clique in result graph $result\_AG_k$ having the same cluster label as one of the cliques in the query graph AG_q, Equation 4.2 actually computes a score considering w adjacent neighbors in the adjacency matrix of "$result\_AG_k(i,j) = 2$". The score is computed by summing the number of cliques of order 2 in the w adjacent neighbors ($z = 1$) and the number of cliques that have the same cluster label as one of the cliques in query graph AG_q ($z = 2$). The latter is given double importance than the former. The sum is normalized by the size w of the neighborhood under consideration.

The computed score of "$result\_AG_k(i,j) = 2$" gives a confidence value for subgraph spotting of query graph AG_q in result graph $result\_AG_k$. Score is calculated for subgraph around all cliques in a result graph having "$result\_AG_k(i,j) = 2$". And then based on a threshold on the confidence values (score), the subgraphs in the result graph $result\_AG_k$ are discarded or included in the final results for subgraph spotting.

Figure 4.2 illustrates the computation of the score for a subgraph in an example adjacency matrix. The score is computed for a clique in result graph $result\_AG_k$ having the same cluster label as one of the cliques in the query graph AG_q (i.e. $result\_AG_k(i,j) = 2$). The value of w is taken to be 9.

In Figure 4.2, score is computed for a clique in the result graph $result\_AG_k$. This clique is highlighted as green in the adjacency matrix of graph $result\_AG_k$. The cliques highlighted as yellow, are the cliques that have the same cluster label as a clique in query graph.

The cliques highlighted as cyan are the cliques of the graph $result\_AG_k$ which do not have the same label as a clique of the query graph. The inclusion of these cliques does not necessarily mean that the method finds only connected subgraphs in the graph.

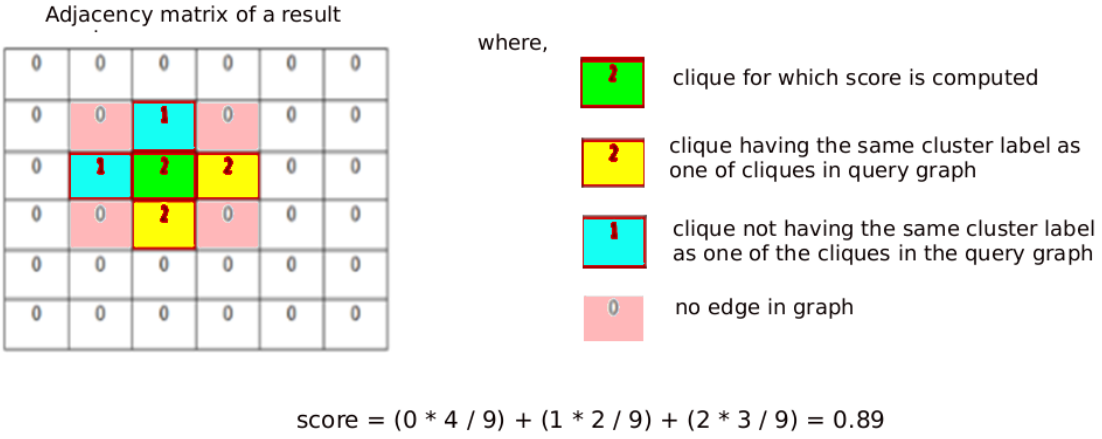


Figure 4.2: Illustration of score function computation for a subgraph around a clique of order 2 in a retrieved graph.

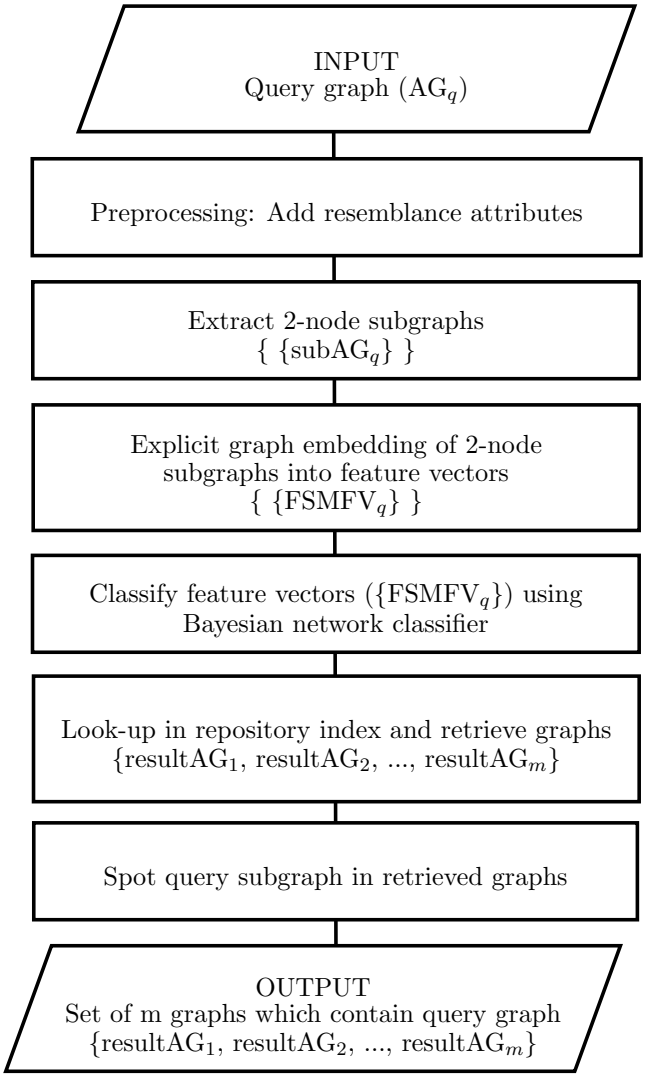


Figure 4.3: Graph retrieval and subgraph spotting.

4.4 Conclusion

In this chapter, we have presented a method of graph retrieval and subgraph spotting through explicit graph embedding. The method is a straightforward, simple and computational efficient solution clearly illustrating the use of graph based powerful representations together with learning and computational strengths of machine learning, classification and clustering. The method does not rely on any domain specific details and offers a very general solution to the problem of subgraph spotting. This enables its less expensive and fast deployment to a wide range of application domains. Apart from the advantages of incorporating learning abilities in structural representations (without requiring any labeled training set) and offering the ease of query by example (QBE) and the granularity of focused retrieval, the system does not impose any strict restrictions on the size of query subgraph.

We have used the cliques of order 2 for constructing the index of the graph repository. A clique of order 2 represents the basic structural unit of a graph. It contains discriminatory information that FMGE extracts and embeds into feature vector. The computation for obtaining the list of cliques of order 2 (for a given graph) is simple and is achieved by a simple lookup in the adjacency matrix of graph. Our choice of a low order clique for indexing a graph repository is also motivated from the fact that the extraction of cliques of higher order (≥ 2) is computational expensive and is a combinatorial problem.

Chapter 5

Experimentations

In this chapter we present the experimental evaluations of FMGE for the problems of graph clustering and graph classification. We present the experimental evaluations of the framework for automatic indexing of graph repositories for graph retrieval and subgraph spotting, with an application to content spotting in graphic document image repositories. We provide experimental results for the application of the thesis work to the real problems of recognition, indexing and retrieval of graphic document images.

5.1 Introduction

The experimental evaluation of FMGE is performed on various standard graph datasets, by International Association of Pattern Recognition's Technical Committee on graph-based representations (TC-15), from the fields of document image analysis, graphics recognition and object recognition, in context of clustering, classification, retrieval and subgraph spotting tasks.

A first set of experimentations is performed with an aim to confirm that FMGE and the subsequent classification and clustering in real vector spaces is applicable to different graph classification and graph clustering problems, and also to confirm that in certain cases it outperforms the classical techniques for the latter.

A second set of experimentations is performed to evaluate the graph retrieval and subgraph spotting framework, with an application to content spotting in graphic document image repositories.

Experimental results for the application of the thesis work to the real problems of recognition, indexing and retrieval of graphic document images are also presented.

5.2 Graph databases

IAM graph database: Six datasets from IAM graph database repository have been employed for the experimentations. The IAM graph database repository is originally proposed by [Riesen and Bunke, 2010c]. These graph datasets are publicly available[1]. The IAM graph database repository contains graphs from the field of document image analysis and graphics recognition. The six datasets used for the experimentation include three versions of letter dataset (low, medium and high level of distortions), GREC dataset, fingerprint dataset and the mutagenicity dataset.

Table 5.1 summarizes the details on the graphs in these datasets.

Table 5.1: IAM graph database details.

Dataset	**Size**			**Classes**	**Avg**		**Max**		**Attributes**[a]	
	Train	Valid	Test		$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	V	E
Letter LOW	750	750	750	15	4.7	3.1	8	6	2;0	0;0
Letter MED	750	750	750	15	4.7	3.2	9	7	2;0	0;0
Letter HIGH	750	750	750	15	4.7	4.5	9	9	2;0	0;0
GREC	836	836	1628	22	11.5	12.2	25	30	2;1	1;1
Fingerprint	500	300	2000	4	5.4	4.4	26	25	2;0	1;0
Mutagenicity	500	500	1500	2	30.3	30.8	417	112	0;1	1;0

[a] Number of attributes is given as a pair "numeric;symbolic".

[1] `http://www.greyc.ensicaen.fr/iapr-tc15/links.html`

GEPR graph database: GEPR graph database [Foggia and Vento, 2010] has been employed for experimentation on graphs from the field of object recognition. The GEPR database is comprised of three graph datasets, extracted from large publicly available image repositories, namely the Amsterdam Library of Object Images (ALOI), the Columbia Object Image Library (COIL) and the Object Data Bank by Carnegie-Mellon University (ODBK).

The summary of performance evaluation subset of ALOI, COIL and ODBK graph datasets in GEPR graph database, together with some characteristic properties, is given in Table 5.2.

Table 5.2: GEPR graph database details.

Dataset	**Size**	**Classes**	**Avg**		**Max**		**Attributes**[a]	
			$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	V	E
ALOI	1800	25	18.37	17.25	134	156	2;0	0;0
COIL	1800	25	34.88	32.33	100	92	2;0	0;0
ODBK	1248	104	56.91	54.37	528	519	2;0	0;0

[a] Number of attributes is given as a pair "numeric;symbolic".

Further details on the IAM and GEPR graph datasets are presented in Appendix A of this dissertation.

SESYD graph database: A new graph repository is constructed by extracting graphs from images of architectural floor plans and electronic diagrams in SESYD image dataset [Delalandre et al., 2010], for experimentation of the framework for automatic indexing of graph repositories for graph retrieval and subgraph spotting. The corresponding graph repository is made publicly available[2] for academia and scientific community for research purposes.

The summary of SESYD graph datasets, together with some characteristic properties, is given in Table 5.3.

Table 5.3: SESYD graph database details.

	Image		Attributed graph	
Electronic diagrams	Backgrounds	8	Avg. order	212
	Models	21	Avg. size	363
	Symbols	9600	Node attribs.	4
			Edge attribs.	2
	Documents	800	Graphs	800
	Queries	1000	Graphs	1000
Architectural floor plans	Backgrounds	2	Avg. order	359
	Models	16	Avg. size	733
	Symbols	4216	Node attribs.	4
			Edge attribs.	2
	Documents	200	Graphs	200
	Queries	1000	Graphs	1000

The extraction of graphs from SESYD image dataset is explained in detail in Appendix B of the dissertation.

[2] `http://www.rfai.li.univ-tours.fr/PagesPerso/mmluqman/public/SESYD_graphs.zip`

5.3 Graph classification

The graph classification experimentation has been performed on the six graph datasets from IAM graph database repository.

The three distorted versions of letter graph dataset, GREC graph dataset, fingerprint graph dataset and the mutagenicity graph dataset are all composed of training, validation and test sets. We have employed the training set for unsupervised learning phase of FMGE and the validation set to optimize the number of fuzzy intervals for structural level and elementary level information of graphs.

To obtain the initial set of crisp intervals for graph classification experimentation, we have used an equal spaced discretization technique. The use of an equal spaced discretization technique demonstrates the learning abilities of FMGE and its generalization abilities to unseen graphs in test set. Equal spaced discretization allows FMGE to ignore the shape of distribution of the attribute i instances (structural and elementary level numeric information), for learning the s_i fuzzy intervals for it. The shape of this distribution is not important and keeping FMGE independent of it allows FMGE to generalize to graphs in many diverse test sets.

The number of fuzzy intervals to be used for an attribute i in a given graph dataset has been optimized empirically on the respective validation set. The experimentation was repeated for 2 to 30 fuzzy intervals for each attribute i. The number of intervals maximizing the classification rate on validation set were used for evaluating the performance of FMGE on graphs in test set.

For graph classification experimentation we have employed a nearest neighbor classifier (1-NN) with Euclidean distance.

Table 5.4 presents the classification results of FMGE and compares them with the best state-of-the-art results from dissimilarity based graph embedding technique of Bunke et al. [Bunke and Riesen, 2011b], on these datasets.

Contrary to our choice of nearest neighbor classifier Bunke et al. have employed an SVM for their experimentation; which is a more sophisticated classifier. For comparison purposes this does not make a great difference and the use of a sophisticated classifier will only increase the performance of our graph embedding method.

The first column of Table 5.4 presents the classification results for the reference system. The reference system is comprised of a graph edit distance based nearest neighbor classifier [Bunke and Riesen, 2011b]. The choice of reference system is a direct outcome of the fact that there is a lack of general classification algorithms that can be applied to graphs. One of the few classifiers directly applicable to arbitrary graphs is the k-nearest-neighbor classifier (k-NN). Given a labeled set of training graphs, an unknown graph is assigned to the class that occurs most frequently among the k nearest graphs (in terms of edit distance) from the training set. The decision boundary of this classifier is a piecewise linear function which makes it very flexible [Bunke and Riesen, 2011b]. This classifier in the graph domain also serves as the reference system for our experimentation.

Table 5.4 presents the classification results for two setups of FMGE. These setups refer to the way in which we have computed edge attributes resemblance for nodes. Column 3 shows the classification results, when the average (or mean) resemblance of the couple of edges is used for defining edge attributes resemblance for nodes - the AVG setup. Column 4 shows the classification results, when the standard deviation of the resemblance of the couple of edges is used for defining edge attributes resemblance for nodes - the STD setup. Both setups give the same classification rates except for fingerprint graphs. This illustrates that for these graph datasets the resemblance between the couple of edges for the nodes remains stable in both AVG and STD setups.

The three letter graph datasets, the GREC graph dataset, the fingerprint graph dataset and the mutagenicity graph dataset are all comprised of graphs with only two numeric attributes (x,y position of primitive in underlying image) on nodes. The only dataset among these is the GREC graph dataset which has one symbolic attribute on nodes in addition to the numeric position attributes. Similarly only GREC graphs, fingerprint graphs and mutagenicity graphs have attributes on edges.

Given the fact that the structural and elementary level information extracted by FMGE is directly or indirectly linked with the attributes of nodes and edges in graphs. In our opinion the attributes, on graphs in these six datasets, are not enough for extracting a lot of discriminatory information by FMGE. However, even then the results are comparable to state-of-the-art results on these datasets.

FMGE has the highest classification rate on GREC graphs and mutagenicity graphs. For letter low graphs and fingerprint graphs the FMGE results are comparable to the state-of-the-art results and the reference system. But for letter MED graphs and letter HIGH graphs the FMGE results are not very good. This is because of the fact that the level of distortion for these graphs is very high. The medium and high levels of distortions totally change the topology of the graphs (and the underlying letter becomes unrecognizable). Most of the letters that undergo distortions of medium and high strength are difficult to be recognized, even for a human observer [Riesen, 2010].

If more information is put on the nodes and edges of the graphs the FMGE results for these datasets will be improved; as FMGE extracts most of the information from node and edge attributes (directly or indirectly). More interesting information that may be extracted from underlying image content may include color, shape, rotation and scale information. More details on the nodes and edges of graphs will result into extraction (and eventually embedding) of more discriminatory information by FMGE. The latter will result into improved classification results.

Table 5.4: Experimental results (%), for graph classification on IAM graph database repository.

Dataset	**Graph edit distance based reference system** [k-NN classifier]	**Dissimilarity based embedding** Bunke et al. [Bunke and Riesen, 2011b] [SVM classifier]	**FMGE** resemblance:AVG [1-NN classifier]	**FMGE** resemblance:STD [1-NN classifier]
Letter LOW	99.3	99.3	97.1	97.1
Letter MED	94.4	94.9	75.7	75.7
Letter HIGH	89.1	92.9	60.5	60.5
GREC	82.2	92.4	**97.5**	**97.5**
Fingerprint	79.1		74.9	73.5
Mutagenicity	66.9		**68.6**	**68.6**

5.4 Graph clustering

The graph clustering experimentation has been performed on the six graph datasets from IAM graph database repository (the three distorted versions of letter graph dataset, GREC graph dataset, fingerprint graph dataset and the mutagenicity graph dataset) and the GEPR graphs.

In this section we present details on the two sets of experimentations.

IAM graph database: The six graph datasets, namely, three letter graph datasets, GREC graph dataset, fingerprint graph dataset and the mutagenicity graph dataset, were employed for graph clustering experimentation. We merged the graphs in training, validation and test sets of the respective graph datasets for constructing datasets for graph clustering experimentation.

Generally, for clustering tasks no separate learning set is available. This scenario is very well simulated by our experimentation. The unsupervised learning phase of FMGE used the graphs to be clustered for learning the fuzzy intervals for various numeric attributes in graphs.

The experimentation was performed for each of the six graph datasets, independent of other datasets. For each of the six graph datasets, during a first pass (unsupervised learning phase) FMGE learned a set of fuzzy intervals for the numeric attributes (structural and elementary level information in graphs). These fuzzy intervals were then employed during the graph embedding phase of FMGE.

For clustering experimentation we have used a more sophisticated discretization technique for obtaining the initial set of crisp intervals. This technique is originally proposed by [Colot et al., 1994] for discretization of continuous data and is based on use of Akaike Information Criterion (AIC). It starts with an initial histogram of data and finds optimal number of bins for underlying data. The adjacent bins are iteratively merged using an AIC-based cost function until the difference between AIC-beforemerge and AIC-aftermerge becomes negative. Thus, we get an optimal set of crisp intervals for the underlying data. This set of intervals could be safely termed as equal frequency intervals and is representative of the underlying data.

In other words we can say that the number of fuzzy intervals for node degree, numeric node attributes and numeric edge attributes is calculated from the spread of the respective attributes values for the graph dataset. The resulting fuzzy intervals are true representative of the shape of distribution of these attribute values and are very interesting for embedding the respective graphs by FMGE.

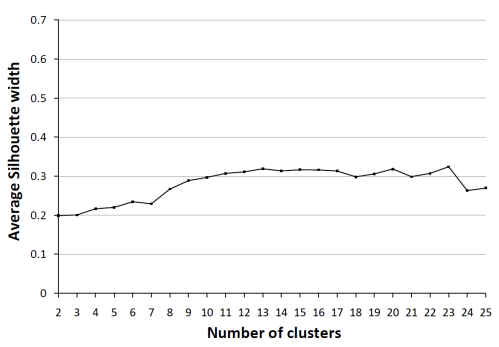
The use of an equal frequency discretization technique (without any training set) demonstrates the unsupervised learning abilities of FMGE in scenarios where no separate training set is available.

We have employed the well known and popular k-means clustering paradigm with Euclidean distance and random non-deterministic initialization, for our graph clustering experimentation.

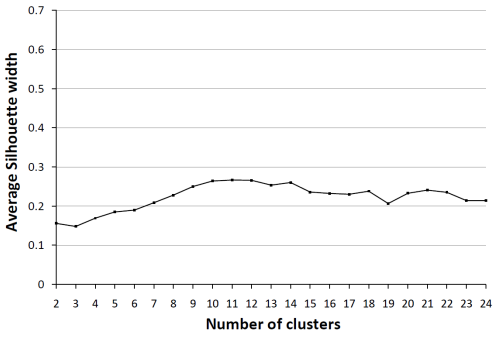
Figure 5.1 presents the Silhouette metric [Kaufman and Rousseeuw, 1990] for datasets in IAM graph database repository, for analyzing the quality of clustering. The Silhouette metric is a standard cluster fitness validation metric. It measures the standardized difference between separation of clusters and the average spread of clusters. The average Silhouette width over all clusters is a value in the range [-1, 1] - the closer this value is to 1 the better is the cluster quality.

The curves in Figure 5.1 presents the Silhouette metric for 2 to 25 clusters. The number of classes in the graph datasets are given in Table 5.1. For example, the mutagenicity graphs originally have 2 classes and the Silhouette metric for mutagenicity graphs for 2 classes in Figure 5.1 is 0.58. Figure 5.1 shows that the Silhouette metric for the actual number of clusters in each of the datasets, is always near or superior to 0.2. The latter is a generally used threshold for Silhouette metric, for compact, better separable and good structure clusters.

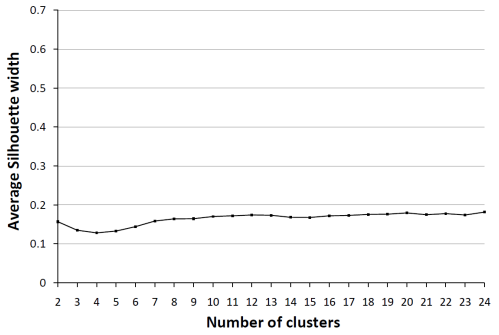
These results demonstrate the graph clustering abilities of FMGE and highlight its unsupervised learning capabilities i.e. FMGE does not require any labeled training set.



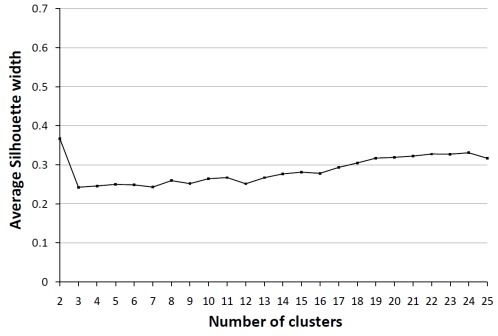
(a) Letter LOW



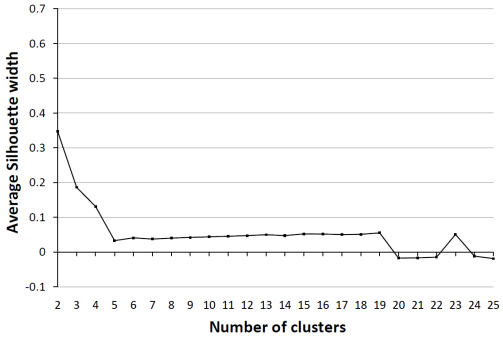
(b) Letter MED



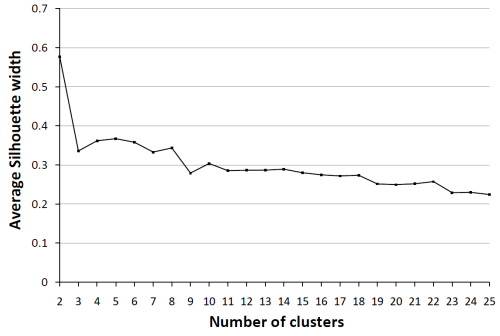
(c) Letter HIGH



(d) GREC



(e) Fingerprint



(f) Mutagenicity

Figure 5.1: Number of clusters versus average Silhouette width for k-means clustering, for IAM graph database repository.

Apart from the Silhouette metric we have also evaluated the quality of clustering achieved by FMGE by the percentage of correctly clustered graph instances. Table 5.5 presents the graph clustering experimentation results for the actual number of classes in various datasets of IAM graph database repository.

Table 5.5: Quality of k-means clustering for IAM graph database repository.

Dataset	**FMGE feature vector space**
	correctly clustered graphs (%)
Letter LOW	89
Letter MED	60
Letter HIGH	41
GREC	82
Fingerprint	57
Mutagenicity	82

For each of the six graph datasets, we have used the actual number of classes in dataset (see Table 5.1) as the number of clusters to be found by k-means clustering. The presented results show the quality of k-means clustering in the FSMFV feature space. It has been calculated as the ratio of correctly clustered graphs to the total number of graphs in a given dataset. For example, a 89% clustering quality for letter LOW graph dataset means that 89% of the graphs are correctly clustered.

These results demonstrate that the proposed methodology of first embedding graphs into feature vectors and then applying a clustering technique has the significant potential to turn an impossible operation in original graph space into a realizable operation with an acceptable accuracy in the resulting FSMFV feature vector space.

The percentage of correctly clustered graphs for letter LOW graphs, GREC graphs and mutagenicity graphs is very good ($\geq$82%). However, for letter MED graphs and fingerprint graphs its not so good. And for letter HIGH graphs the resulting clustering quality is bad. The low clustering quality for letter HIGH graphs is because of the fact that the high level of distortions applied to graphs in this dataset entirely changes the topology of the graphs. This effects the FSMFV of the graphs in this dataset and results into too much overlap between the clusters.

Apart from this another reason, as we have already discussed for graph classification experimentation results, is that there are not enough attributes on the nodes and edges of graphs. FMGE extracts all the structural and elementary level information from graphs by exploiting the node and edge level attributes. The less number of attributes on nodes and edges (and lack of meaningful and discriminatory attributes) is one of the main reasons for low quality of clustering for these datasets.

If more meaningful and discriminatory attributes are attached to the nodes and edges of graphs, the quality of clustering achieved by FMGE will certainly improve. The latter is because of the fact that the attributes are critical to provide the structural and elementary level details to FMGE.

GEPR graph database: A second set of clustering experimentation was performed on GEPR graphs.

This part of experimentations has been published in [Luqman et al., 2010a] and is reproduced here.

Each of the three graph datasets in this repository are employed independently of others, for graph clustering experimentation. Like the first set of graph clustering experimentation (i.e. IAM graph database repository), during unsupervised learning phase we employed the AIC based equal frequency discretization technique for obtaining an initial set of crisp bins. These bins were then arranged in fuzzy overlapping trapezoidal intervals for embedding the details of graphs into feature vectors.

The scripts from GEPR graph database repository were employed to evaluate the performance index for the clustering of the resulting feature vectors. For evaluating the quality of clustering these scripts employ the clustering validation index to evaluate the separation between classes, as proposed by [Foggia and Vento, 2010]. This cluster validation index is based on original C index. For computing it first the Euclidean distances d_{ij} between each pair of graphs (feature vectors) is computed. Given the distances, the C index is defined as follows: first the set S_w of the distances d_{ij} such that g_i and g_j lie in the same class is computed; M is the cardinality of S_w. Then, the sets S_{min} and S_{max} are computed taking respectively the M shortest distances and the M largest distances among all the possible values of d_{ij} . Finally, the index is computed as:

$$C = \frac{sum(S_w) - sum(S_{min})}{sum(S_{max}) - sum(S_{min})}$$

The smaller the value, the better is the separation of the classes - the index value is in the interval [0, 1], reaching 0 in the ideal case in which all the inter-class distances are smaller than all the intra-class distances [Foggia and Vento, 2010].

Table 5.6 provides the cluster validation index for graph datasets of three image databases in GEPR graph repository.

Table 5.6: Performance indexes for GEPR graphs.

Dataset	**Performance index**
ALOI	0.379
COIL	0.377
ODBK	0.355

The clustering results for graphs in GEPR graph repository are adequate and are not very good. The GEPR graphs have attributes only on nodes. The 2 node attributes encode the size and the average color of the image area (represented by RGB values) represented by a node. These are numeric attributes and were encoded by fuzzy overlapping trapezoidal intervals.

The lack of attributes on edges and less number of attributes on nodes is the main reason for average clustering quality. As we have detailed in the discussion on experimentation for IAM graph database, all the information extracted by FMGE is directly or indirectly related to attributes of nodes and edges. If there are not enough attributes on nodes and edges of graphs, FMGE is not able to embed the graphs into discriminant feature vectors. For graphs extracted from object images a very important attribute that can be extracted from images is the relations between neighboring parts of the object. In a region adjacency graph this could be represented by the relationship between adjacent regions of the image.

The addition of more attributes on nodes and edges of the graphs will enable FMGE to extract and eventually embed more information on graphs. This will result into more discriminant feature vectors which will result into higher quality clustering.

5.5 Graph retrieval and subgraph spotting: an application to content spotting in graphic document image repositories

Subgraph spotting is a very interesting research problem for various application domains where the use of a relational data structure is mandatory. The Query By Example (QBE) based content retrieval systems for graphic document image repositories is a classical application domain of subgraph spotting.

The graph retrieval experimentations have been performed on SESYD graphs (Table 5.3), presenting an application of the proposed framework to the problems of query by example (QBE) and focused retrieval for graphic document image repositories. The graphs have been extracted from two document image repositories comprised of electronic diagrams and architectural floor plans.

The extraction of the graphs is detailed in Appendix B. A document image is represented by a graph in our system. To facilitate the explanation we have used the two terms interchangeable while discussing the experimental results.

An initial version of this work has been published in [Luqman et al., 2010b]. A complete version of the work is published in [Luqman et al., 2011b] and is reproduced here.

In last few years, content based information retrieval (CBIR) systems for graphic document image repositories, have emerged as an important application domain. This has resulted into a gradual shift of attention from the hard problems of symbol recognition and localization to the relatively softer problem of content spotting (a.k.a. symbol spotting). This is a direct outcome of growing size of document image repositories and the increasing demand from users to have an efficient browsing mechanism for graphic content. The format of these documents restricts the use of classical keyword based indexing mechanisms. Thus a very interesting research problem is to investigate into mechanisms of automatically indexing the content of graphic document images; in order to offer to users the ease of query by example (QBE) and granularity of focused retrieval. Graphs have remained a very popular choice of graphics recognition research community [Lladós et al., 2002, Tombre et al., 2006] and keeping in view the wide use of graph based representations for graphic document images, the problem of content spotting in graphic document image repositories, in fact, becomes the problem of subgraph spotting.

In our experimentation, the electronic diagrams and architectural floor plans graph repositories were treated independently of each other for learning and querying phases of the system.

The unsupervised learning phase was performed off-line for automatically indexing the graphs in each of these repositories. This was followed by posing 1000 queries (during an

on-line phase) to each of the repositories for evaluating the performance of the system.

During the automatic indexing phase of graph repositories, a total of 516714 2-node subgraphs (i.e. cliques of order 2) were extracted for electronic diagrams and 305824 2-node subgraphs (i.e. cliques of order 2) for architectural floor plans. These cliques were clustered into 455 classes for electronic diagrams and 211 classes for architectural floor plans. The indexing of electronic diagrams graph repository took $\sim$17 hours on a standard PC with 2GB of RAM and the subgraph spotting was performed in real-time.

For evaluating the performance of our method, we have employed the standard precision and recall measure. Figure 5.2 presents the precision and recall plot for the central tendency of the queries of respective attributed graph datasets. The curves are obtained by retrieving the graphs until the system achieves 100% recall.

It is important to highlight here, that the results in Figure 5.2 shows the retrieval performance of the system. This means that a graph (a document image) in result is considered a good retrieval, if it contains the query graph (query graphic content). The method highlights the spotted subgraphs in the result graphs (zones in the retrieved document images) but we have not taken them into account for computing the precision recall curve. Our this choice is mainly because of the reason that there is no widely accepted (standard) mechanism for focused retrieval results.

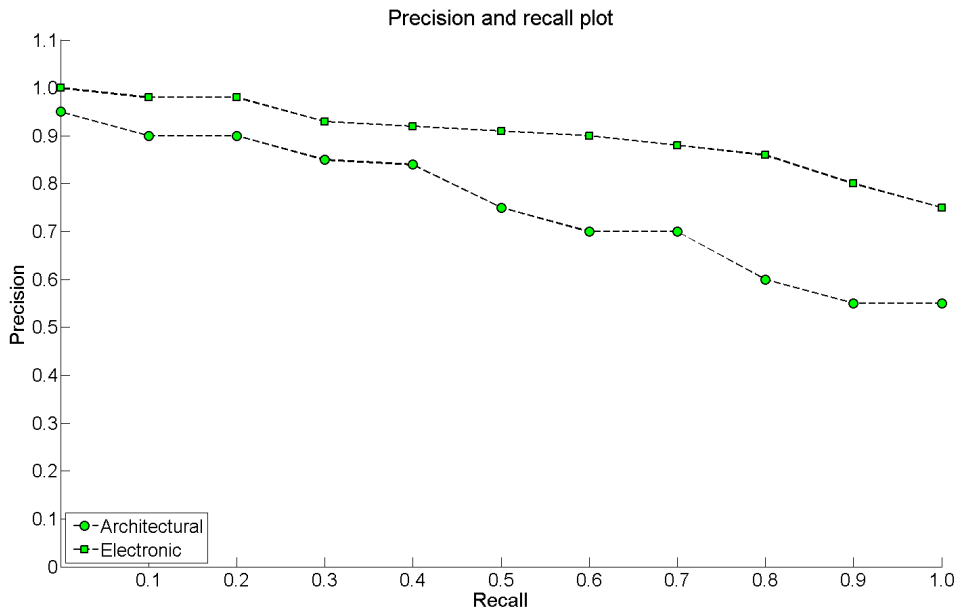


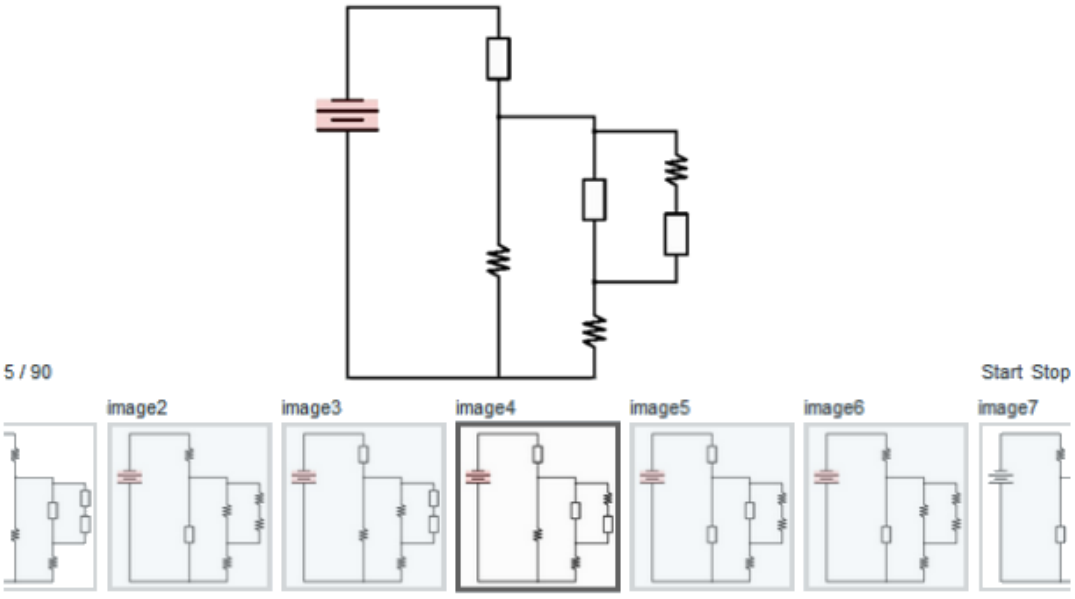
Figure 5.2: Precision and recall plot for graph retrieval from SESYD graph database.

The focused retrieval performance of the system was evaluated manually for a subset of queries. The results are not very good but are encouraging. Some snapshots of the focused retrieved results for query images are presented in Figure 5.3 and Figure 5.4. As show in Figure 5.3, if the document image contains only one instance of the query symbol, the focused retrieval results are very good. The system retrieves only relevant document images and correctly highlights the interesting zones (corresponding the to query image) in the retrieved document images.

Figure 5.4 shows an example for the scenario where the system confuses the interesting zones in the retrieved document images. The case when a retrieved document image contains multiple instances of the query image is a good example of latter. The system wrongly highlights the zones in the vicinity of the interesting zones (corresponding to the query image).

This behavior of the system is mainly influenced by the size of the subgraph for spotting the query graph in a retrieved graph. To recall the reader: this is controlled by the parameter w for computing *score* of zones in retrieved graphs. The calculation of *score* has already been discussed in detail in Section 4.3 of the dissertation.

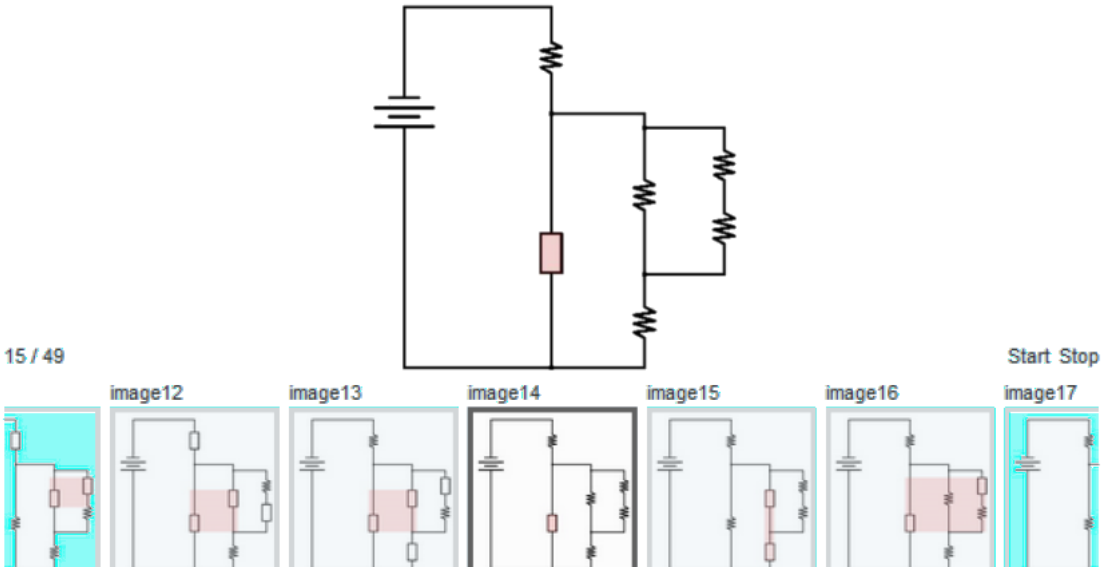
(a) Query image.



(b) Documents retrieved.

Figure 5.3: A snapshot of retrieved results for a query image (single instance of query symbol).

(a) Query image.



(b) Documents retrieved.

Figure 5.4: A snapshot of retrieved results for a query image (multiple instances of query symbol).

The graph retrieval experimentation results demonstrate that the proposed method of graph embedding (FMGE) is capable of automatically indexing graph repositories and is capable of maintaining a high precision rate for sufficiently large graph repositories.

The performance of the system strongly depends upon the graph extraction phase. In our case, we have used a graph extraction method, which is based on a prior vectorization phase [Qureshi et al., 2007]. The employed vectorization technique approximates the circles and arcs by a set of straight lines. A small distortion in the shape of underlying content results into a change in topology of the graph and it effects the graph embedding phase of the system. Because of the latter and the fact that architectural floor plans are comprised of many symbols with arcs and circles, the system has a low performance for this dataset.

The number of clusters and the size of the resulting index is also very important and influences the performance of the system. We have used an agglomerative clustering and used an automatic cut-off for determining the number of clusters for graphs in a repository. This makes the system independent of manually setting the number of clusters. Too large number of classes means that the system indexes too many classes of the cliques of order 2. On the other hand, too small number of clusters means that the system merges the classes of the cliques of order 2. The quality of the learned classifier, in both cases, will be affected. And during the graph spotting phase the system will confuse to determine the class of the cliques of order 2 in query graph. The number of clusters, size of the index and the quality of the learned classifier are inter linked and depend on each other.

However, generally (for the given datasets) the system is able to maintain a high precision value for the series of different recall values. This provides a very interesting solution for the automatic indexing of graph repositories for retrieval and (specially for) browsing of content in graph representation of graphic document image repositories.

5.6 Application of FMGE to graphics recognition

This section demonstrates a real application of FMGE to the problem of graphics recognition. We have achieved graphic symbol recognition by a framework operating in representation, description, classifier learning and classification phases. The representation phase represents the images of graphic entities by attributed graphs. The description phase employs FMGE based graph embedding for encoding the graphs into numeric feature vectors. The classifier learning and classification phases employ Bayesian network classifier to achieve graphics recognition.

Several initial prototypes of FMGE have been used for graphic symbol recognition. These works have been published in [Luqman et al., 2009b], [Luqman et al., 2009a], [Luqman et al., 2010c], [Luqman et al., 2010d] and [Brouard et al., 2010].

Graphics recognition deals with graphic entities in document images and is a subfield of document image analysis. These graphic entities could correspond to symbols, mathematical formulas, musical scores, silhouettes, logos etc., depending on the application domain. [Llados and Sanchez, 2003] has very correctly pointed out that the documents from electronics, engineering, music, architecture and various other fields use domain-dependent graphic notations which are based on particular alphabets of symbols. These industries have a rich heritage of hand-drawn documents and because of high demands of application domains, overtime symbol recognition is becoming core goal of automatic image analysis and understanding systems. Hand-drawn based user interfaces, backward conversion from raster images to CAD, content based retrieval from graphic document databases and browsing of graphic documents are some of the typical applications of symbol recognition. Detailed discussion on the application domains of symbol recognition has been provided by research surveys [Chhabra, 1998, Lladós et al., 2002, Cordella and Vento, 2000, Tombre et al., 2006].

5.6.1 Representation phase

This important and basic phase concerns the representation of images of graphic symbols by an Attributed Relational Graph (ARG), as proposed by [Qureshi et al., 2007], and is summarized in Figure 5.5.

The topological and geometric details about structure of symbol are extracted and are represented by an ARG. In first step, the symbol is vectorized and is represented by a set of primitives. These are given by labels 1, 2, 3, 4 in the vectorization part of Figure 5.5). In next step, the graph is constructed by using the aforementioned primitives as the nodes and the topological relations between them as the edges. Nodes are assigned the *'relative length' (normalized between 0 and 1)* and *'primitive-type' (Vector for filled regions of shape and Quadrilateral for thin regions)* as attributes; whereas edges of the graph have *'connection-type' (L, X, T, P, S)* and *'relative angle' (normalized between 0 and 90)* as attributes.

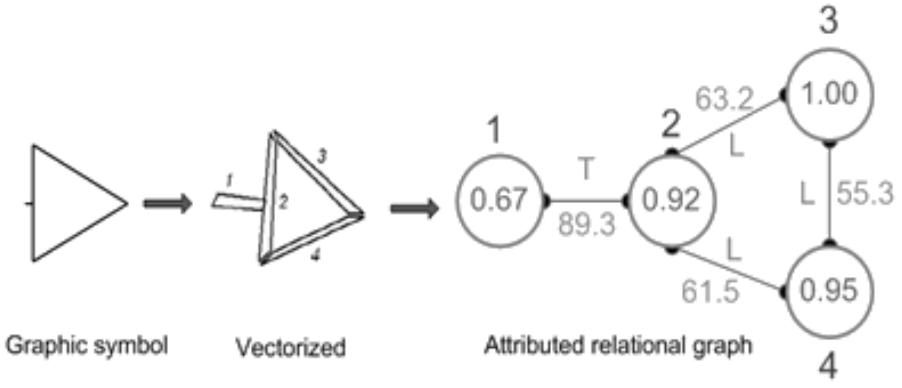


Figure 5.5: Representing a graphic symbol image by an attributed graph.

5.6.2 Description phase (FMGE)

This phase concerns the description of attributed graph representations of graphic symbols by feature vectors. The is achieved through Fuzzy Multilevel Graph Embedding (FMGE). FMGE extracts the graph level details, subgraph homogeneity details and elementary level details from graph representations of symbols in learning set and adapts its parameters to underlying graphs, during the off-line unsupervised learning phase. It embeds the attributed graph representations of the symbols into Fuzzy Structural Multilevel Feature Vectors (FSMFV), during the on-line graph embedding phase. The use of fuzzy logic enables FMGE to increase the robustness of the graph embedding and provides resistance against the irregularities and uncertainties introduced in shape of symbol as result of degradations and deformations.

The features in the signature for attributed graphs of graphic symbols are adapted to the underlying graph representation and are given in Figure 5.6. As show in the Figure 5.6, three intervals are used for encoding numeric information in graphs i.e. for density of connections at nodes (node degree), distribution of relative length of primitives (the relative length attribute of nodes) and distribution of relative angle of connections (the relative angle attribute of edges). We have not used the resemblance attributes for embedding these graphs.

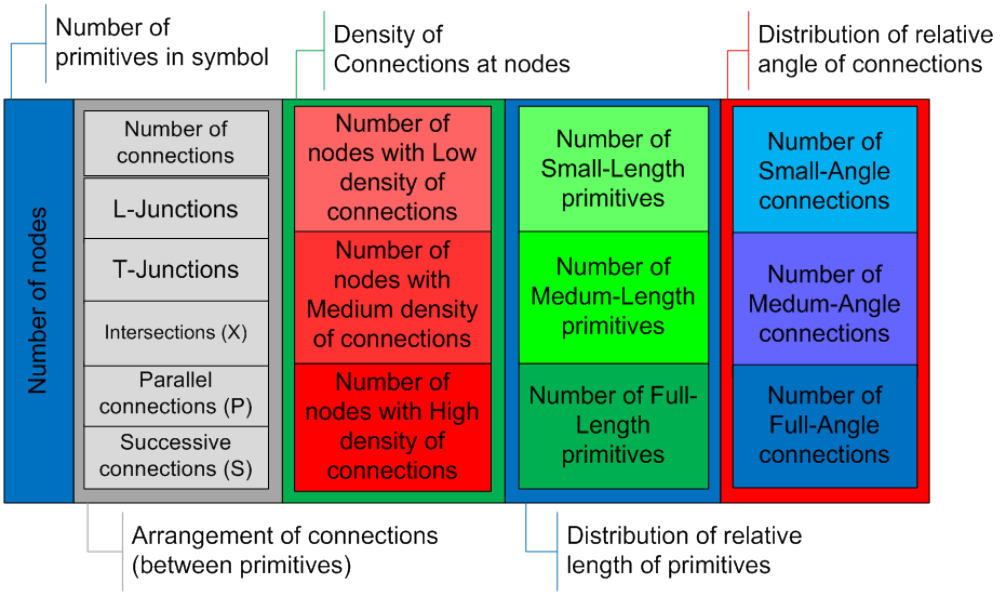


Figure 5.6: FMGE embedding of an attributed graph of a graphic symbol.

5.6.3 Classifier learning phase

We have used a Bayesian network classifier for modeling the joint probability distribution of the FSMFVs of the attributed graph representations of the graphic symbols in learning set. Our choice of using a Bayesian network classifier is primarily motivated by the fact that the probabilistic reasoning of Bayesian networks enables the framework to handle the uncertainties in the feature vector embedding of the attributed graphs (of symbols). As lots of information is lost while mapping from high dimensional continuous graph space to discrete feature vector space.

The feature vectors are first discretized [Leray and Francois, 2004]. We discretize each feature of the feature vector separately and independently of others. The class labels are chosen intelligently in order to avoid the need of any discretization for them.

The Bayesian network is learned in two steps.

First we learn the structure of the network by genetic algorithms proposed by Delaplace et al. [Delaplace et al., 2006]. These are evolutionary algorithms, but in our case they have provided stable results *(for a given dataset multiple invocations always returned identical network structures)*. Each feature in feature vector becomes a node of network. The goal of structure learning stage is to find the best network structure from underlying data which contains all possible dependency relationships between all feature pairs. The structure of the learned network depicts the dependency relationships between different features in signature.

The second step is learning of parameters of network; which are conditional probability distributions $\Pr(node_i|parents_i)$ associated to nodes of the network and which quantify the dependency relationships between nodes. The network parameters are obtained by maximum likelihood estimation (MLE); which is a robust parameter estimation technique and assigns the most likely parameter values to best describe a given distribution of data. We avoid null probabilities by using Dirichlet priors with MLE. The learned Bayesian network encodes joint probability distribution of the symbol signatures.

5.6.4 Classification phase (graphic symbol recognition)

For recognizing a query symbol we use Bayesian probabilistic inference on the encoded joint probability distribution. This is achieved by using junction tree inference engine which is the most popular exact inference engine for Bayesian probabilistic inference and is implemented in [Leray and Francois, 2004]. The inference engine propagates the evidence (feature vector of query symbol) in network and computes posterior probability for each symbol class. Equation 5.1 gives Bayes rule for the system. It states that posterior probability or probability of a symbol class c_i given a query signature 'evidence e' is

computed from likelihood (probability of e given c_i), prior probability of c_i and marginal likelihood (prior probability of e). The marginal likelihood is to normalize the posterior probability; it ensures that the probabilities fall between 0 and 1.

$$\Pr(c_i|e) = \frac{\Pr(e, c_i)}{\Pr(e)} = \frac{\Pr(e|c_i) \times \Pr(c_i)}{\Pr(e)} \tag{5.1}$$

where,

$e = f1, f2, f3, ..., f16$ and
$\Pr(e) = \sum_{i=1}^{k} \Pr(e, c_i) = \sum_{i=1}^{k} \Pr(e|c_i) \times \Pr(c_i)$

The posterior probabilities are computed for all 'k' symbol classes in learning set and the query symbol is then assigned to class which maximizes the posterior probability i.e. which has highest posterior probability for the given query symbol.

5.6.5 Symbols with vectorial and binary noise

We have performed this set of symbol recognition experimentations on synthetically generated 2D symbols of models collected from database of GREC2005
[Dosch and Valveny, 2006]. In order to get a true picture of the scalability performance of the system on this database, we have experimented with 20, 50, 75, 100, 125 & 150 symbol classes. The learning and test sets are generated based on the deformations and degradations of GREC2005. For each class the perfect symbol i.e. the model, along with its 36 rotated and 12 scaled examples was used for learning as generally Bayesian network learning algorithms perform better on datasets with large number of examples. The system has been tested for its scalability on clean symbols (rotated and scaled), various levels of vectorial deformations and for binary degradations of GREC symbol recognition contest (Figure 5.7 and Figure 5.8). Each test dataset is composed of 10 query symbols for each class.

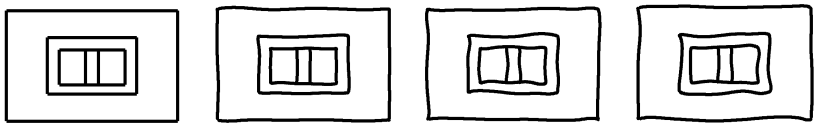


Figure 5.7: Model symbol with deformations, used for simulating hand-drawn symbols.

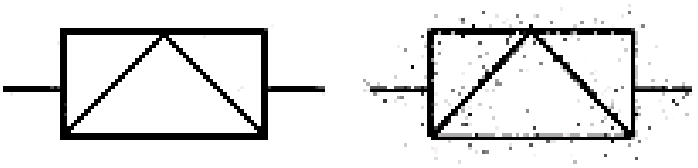


Figure 5.8: Model symbol with degraded example, used to simulate photocopying / printing / scanning.

Table 5.7: Results of symbol recognition experiments for vectorial and binary noise.

Number of classes (models)		**20**	**50**	**75**	**100**	**125**	**150**
Clean symbols (rotated & scaled)		100%	100%	100%	100%	100%	99%
Hand-drawn deformation	Level-1	99%	96%	93%	92%	90%	89%
	Level-2	98%	95%	92%	90%	89%	87%
	Level-3	95%	77%	73%	70%	69%	67%
Binary degrade		98%	96%	93%	92%	89%	89%

Table 5.7 summarizes the experimental results. A 100% recognition rate for clean symbols illustrates the invariance of our method to rotation and scaling. Our system outperforms all GREC participants (available results from GREC2003 [Valveny and Dosch, 2004] and GREC2005 [Dosch and Valveny, 2006] competetions) in scalability tests and is comparable to contest participants for low levels of deformation and degradations. The recognition rates decrease with level of deformation and drop drastically for high binary degradations. This is an expected behaviour and is a result of the irregularities produced in symbol signature; which is a direct outcome of the noise sensitivity of vectorization step, as also pointed out by [Lladós et al., 2002]. We used only clean symbols for learning and (thus) the recognition rates truely illustrate the robustness of our system against vectorial and binary noise.

5.6.6 Symbols with contextual noise

This set of experimentations was performed on a synthetically generated corpus, of symbols cropped from complete documents [Delalandre et al., 2010]. These experiments focus on evaluating the robustness of the proposed system against context noise i.e. the structural noise introduced in symbols when they are cropped from documents. This type of noise gets very important when we are dealing with symbols in context in complete documents. We have performed these experiments on two subsets of symbols: consisting of 16 models from floor plans and 21 models from electronic diagrams. The models are derived from GREC2005 database [Dosch and Valveny, 2006] and are given in Figure 5.10 and Figure 5.11. For each class the perfect symbol (model), along with its 36 rotated and 12 scaled examples was used for learning. The examples of models, for learning and test sets were generated synthetically [Delalandre et al., 2010]. The test set contains symbols with different levels of context-noise (Figure 5.9) in order to simulate the cropping of symbols from documents. Test symbols were randomly rotated and scaled and multiple query symbols were included for each class.

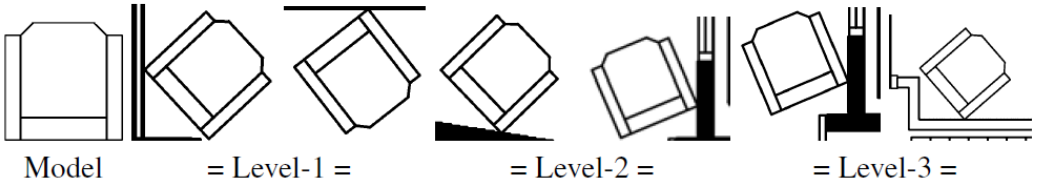


Figure 5.9: An arm chair with 2 examples of each different level of contextual noise.

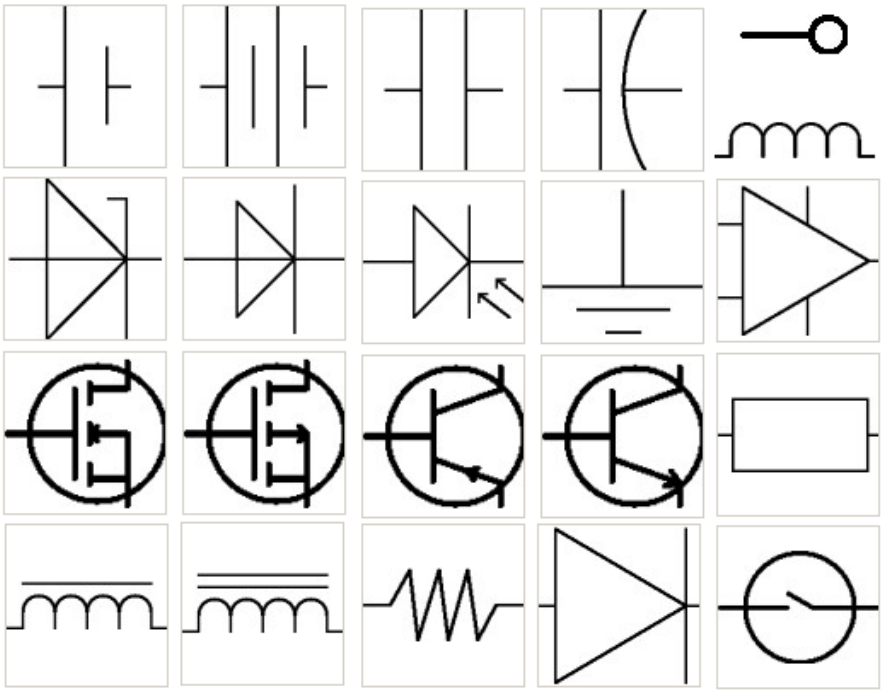


Figure 5.10: Model symbols from electronic drawings.

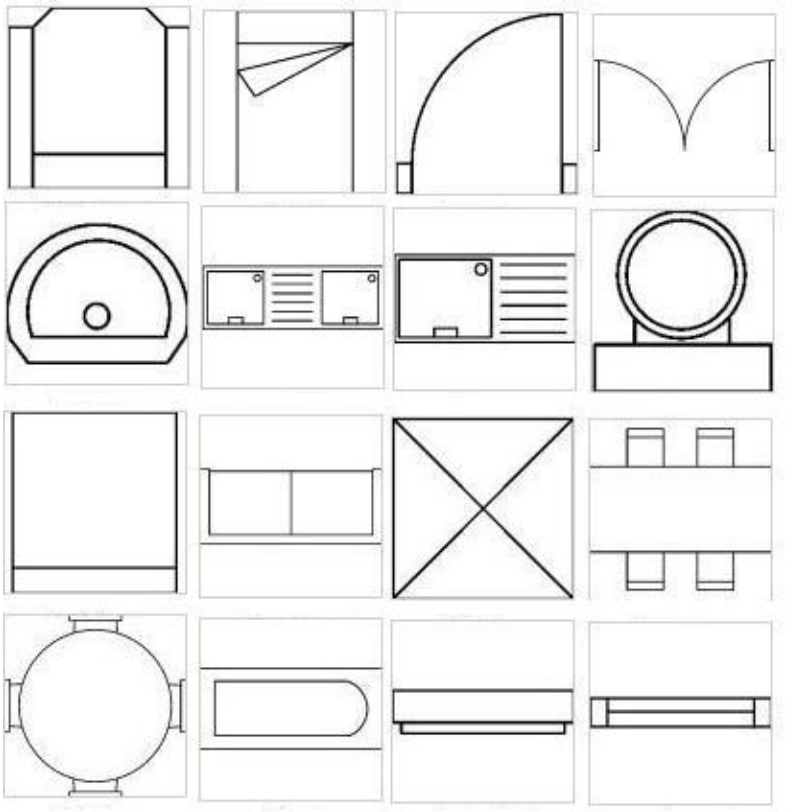


Figure 5.11: Model symbols from floor plans.

Table 5.8 summarizes the results of experiments for context noise. We have not used any sophisticated de-noising or pretreatment and our method derives its ability to resist against context noise, directly from underlying vectorization technique, the fuzzy approach used for computing structural signature and the capabilities of Bayesian networks to cope with uncertainties. The models for electronic diagrams contain symbols consisting of complex arrangement of lines and arcs, which affects the features in structural signature as the employed vectorization technique is not able to cope with arcs and circles; as is depicted by the recognition rates for these symbols. But keeping in view the fact that we have used only clean symbols for learning and noisy symbols for testing, we believe that the results show the ability of our signature to exploit the sufficient structural details of symbols and it could be used to discriminate and recognize symbols with context noise.

Table 5.8: Results of symbol recognition experiments for context noise.

	Noise	**Model symbol (classes)**	**Query symbol (each class)**	Recognition rate (match with topmost result)	Recognition rate (a match in top-3 results)
Floor plans	Level-1	16	100	84%	95 %
	Level-2	16	100	79%	90 %
	Level-3	16	100	76%	87 %
Average recognition rate				80%	91%
Electronic diagrams	Level-1	21	100	69%	89%
	Level-2	21	100	66%	88%
	Level-3	21	100	61%	85%
Average recognition rate				65%	87%

Structural methods are the strongest methods for graphics representation and statistical classifiers provide efficient recognition techniques. By designing a mechanism to convert a structural representation to feature vector, the whole range of statistical tools (classifiers) are opened for that structural representation.

The application of FMGE to graphic symbol recognition does not use any sophisticated de-noising or pretreatment and it drives its power to resist against deformations and degradations, directly from representation, description, learning and classification phases. FMGE addresses the issue of sensitivity of structural representations to noise and deformations by introducing fuzzy overlapping trapezoidal intervals for computing structural signature.

Promising experimental results confirm that the proposed methodology of combining structural representation and statistical classifier is useful for graphics recognition.

5.6.7 Complexity of FMGE

The graph embedding phase of one attributed graph by FMGE requires only a fraction of a second and can be performed in real-time on standard PC.

The unsupervised learning phase of FMGE can be computational intensive depending on the size of dataset and the graphs. However, this phase is performed off-line and has linear time complexity. This is illustrated by Figure 5.12, which presents the time requirements of the unsupervised learning phase of FMGE for synthetically generated graphs from IAM GREC dataset. All parameters except the number of attributes are kept constant.

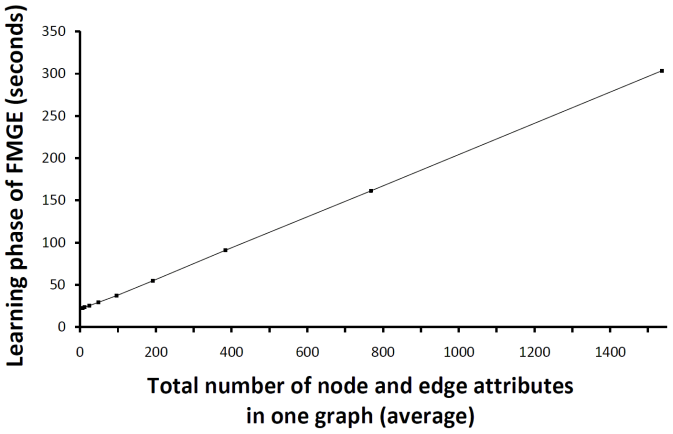


Figure 5.12: Time complexity of unsupervised learning phase of FMGE.

5.7 Conclusion

In this chapter we have presented the experimental evaluations of FMGE for the problems of graph clustering and graph classification. We have also presented the experimental evaluation of the frame work for automatic indexing of graph repositories for graph retrieval and subgraph spotting. Along with the experimental evaluations we have presented the application of FMGE to the real problems of recognition, indexing and retrieval of graphic document image repositories.

The experimental evaluations have confirmed that FMGE and the subsequent classification and clustering in real vector spaces is applicable to different graph classification and graph clustering problems. The NP-complete operations in graph space can be solved in polynomial time (P) in FMGE feature vector space. Also, In certain cases FMGE outperforms the classical techniques for graph classification and graph clustering, in terms of recognition rate and quality of clustering respectively.

However, a limitation of FMGE is that it is dependent on the attributes of the graph nodes and edges, for extracting discriminatory information. The features for the structural information, also, use the node and edge attributes for extracting subgraph homogeneity. The direct consequence of the dependence on the node and edge attributes in graph, is that the performance of FMGE is affected if the graphs do not have many attributes on nodes and edges.

On the other side, less dependent on the topology of the graph is useful in case of noisy data. Generally, the noise in underlying data has a direct impact on the topology of the graph. Less dependence on the topology helps FMGE in robustness against the deformations in graph.

The framework for automatic indexing of graph repositories proposes a very general solution, applicable to a wide range of domains where the use of graphs is mandatory. Experimental results for the two repositories of electronic diagrams and architecture floor plans are very encouraging.

The application of FMGE to graphics recognition, demonstrates that the graph based structural representations for graphic document images could be adapted modern computational tools.

Chapter 6

Discussion and Conclusions

In this chapter we present a discussion on the presented work. We highlight the possible improvements which should be further studied for improving the proposed method of explicit graph embedding. We conclude this dissertation and point out some important possible lines of future research.

6.1 Discussion about FMGE

6.1.1 Parameters

The Fuzzy Multilevel Graph Embedding (FMGE) method presented in this thesis has proposed a framework for extracting three levels of details from the attributed graphs by performing multilevel analysis of graphs. The three levels of details include graph level details, structural level details and the elementary level details. It can process graphs with symbolic as well as numeric attributes on both nodes and edges. The size of the graphs is not an issue, FMGE can embed small as well as big graphs.

The framework permits the embedding of these details of graphs into numeric feature vectors, by employing fuzzy histograms for numeric part of the information and crisp histogram for symbolic part of the information. The framework employs fuzzy overlapping trapezoidal intervals for smooth transition between the intervals for constructing the fuzzy

histogram, in order to minimize the information loss while mapping from continuous graph space to discrete feature vector space.

The size of the resulting feature vectors is mainly based on the histograms (of frequencies) for encoding the information extracted by multilevel analysis of graphs.

The presented framework is quite modular in nature. The possible customizations of its various modules are discussed below:

6.1.1.1 Discretization technique

For embedding numeric attributes, we have outlined to use two basic discretization techniques i.e. equally spaced discretization and AIC based equal frequency discretization. However FMGE is fully capable of employing sophisticated state-of-the-art discretization methods. A survey of popular discretization techniques is presented by Liu et al. [Liu et al., 2002].

6.1.1.2 Fuzzy membership function

Our proposed framework employs trapezoidal membership function from fuzzy logic but FMGE is fully capable of utilizing any of the available membership functions. Other popular choices for fuzzy membership function include triangular and Gaussian arrangements [Ishibuchi and Yamamoto, 2003]. In light of domain knowledge, appropriate choices could be made for discretization technique and fuzzy membership function.

6.1.1.3 Number of fuzzy intervals for numeric attributes

An important parameter of FMGE's unsupervised learning phase is the number of fuzzy intervals to be associated to each attribute. The unsupervised learning phase of FMGE learns s_i fuzzy intervals for attribute i in input collection of graphs. The attribute i refers to structural and elementary level numeric information in input graphs (i.e. node degree, numeric node attributes and numeric edge attributes in input graphs). The parameter s_i for attribute i is independent of other attributes. It is also important to highlight here that this parameter is not necessarily same for all attributes. This parameter can be learned and tested on a validation dataset; if one is available. This is demonstrated by our experimentation on graph classification. If no validation dataset is available the number of fuzzy intervals s_i has to be specified manually. A third possibility is that if the training set can be considered to be a true representative of the test set, an equal frequency discretization technique could be used for automatically finding the appropriate number of fuzzy intervals. This is demonstrated by our graph clustering experimentation.

As a general rule of thumb, using 3 fuzzy intervals (i.e. small, medium and large) for the attributes is a safe choice. However, a more intelligent choice could be made in the light of domain knowledge.

6.1.1.4 Resemblance attributes

For embedding edge attributes resemblance for nodes (structural level information), we have outlined the use of a measure of central tendency i.e. mean and the spread of the attribute's resemblance i.e. the standard deviation on the attribute resemblance of all couple of edges for a node.

6.1.1.5 Classifier

We have used a simple non-parametric classifier for graph classification experimentation and a basic clustering technique for graph clustering experimentation, for demonstration purposes. However, all sophisticated classifiers and clustering methods are fully applicable to the resulting feature vectors of FMGE.

We have illustrated the application of FMGE in different domains but have mainly focused on the graphic document images, because the graphic document images need to generate sophisticated graph representations.

Another important contribution of this work is the proposed method to spot subgraphs in huge repositories of attributed graphs. This is achieved by automatically constructing the index of a graph repository by employing FMGE together with clustering and classification tools on the cliques of order 2 in the graphs in the repository.

6.1.2 Complexity

The unsupervised learning phase of FMGE can be computational expensive, depending on the size of the graphs and that of the graph dataset. This is dependent on the number of attributes of nodes and edges in the graph. However, the complexity of this phase is linear to the number of attributes in graph.

The graph embedding phase of FMGE is fast. It is performed in less that a fraction of a second on a standard PC.

6.2 Conclusions

We have presented a method of explicit graph embedding and a framework for automatic indexing of graph repositories, with an aim to bridge the gap between structural and statistical approaches of pattern recognition. Our work proposes a straightforward, simple and computational efficient solution for facilitating the use of graph based powerful representations together with learning and computational strengths of state-of-the-art machine learning, classification and clustering.

The proposed graph embedding method exploits multilevel analysis of graph for embedding it into a feature vector. The feature vector contains graph level features (graph order and graph size), along-with structural level features (node degree, node attributes resemblance for edges and edge attributes resemblance for nodes) and elementary level features (node attributes and edge attributes). We have minimized the information loss, while mapping from continuous graph space to discrete vector space, by employing fuzzy overlapping trapezoidal intervals. These intervals are learned during an unsupervised learning phase. The intervals are employed during graph embedding phase for constructing fuzzy interval encoded histograms for structural and elementary level features. This achieves an eventual embedding of a graph into a feature vector.

The experimental results are encouraging and demonstrates the applicability of our method to graph clustering and classification problems. The method could be deployed in an unsupervised fashion for graph clustering problem even if no separate learning set is available as it has built-in capability to implicitly learn from the input collection of graphs. Its unsupervised learning capabilities also allows it to generalize to unseen graphs i.e. to be deployed in a supervised fashion where it learns on a graph dataset and embeds unseen graphs. The proposed method is applicable to a wide range of domains for solving the problems of graph clustering and graph classification.

However, our method is too much dependent on the attributes of graph for extracting discriminant information from graph and it lacks information on the topology of graph.

The framework for automatic indexing of attributed graph repositories employs explicit graph embedding and gives a very general solution for graph retrieval and subgraph spotting. The automatic indexing of attributed graph repositories doest not require a labeled training set and thus has the capability of less expensive and fast deployment to various domains where the use of a relational data structure is mandatory.

We have obtained promising experimental results, which clearly demonstrate that Fuzzy Multilevel Graph Embedding (FMGE) enables the structural representations to benefit from the computational efficient statistical models and tools, for the problems of graph classification, graph clustering, graph retrieval and subgraph spotting.

6.3 Future challenges

The current research findings are encouraging and one of the future lines of work is to take this work forward for improving the quality of embedding achieved by FMGE.

An important direction of future research in this regard is to study the feature vector of FMGE in further details. We have reported some very preliminary results in [Luqman et al., 2011a]. The application of dimensionality reduction, using principal component analysis (PCA), independent component analysis (ICA) and their kernel versions, will be very interesting. An interesting reference to proceed in this direction is [Bunke and Riesen, 2011b].

Linked to the dimensionality reduction of feature vector of FMGE, an interesting direction of future research is to use feature selection methods for selecting meaningful features and reducing the size of the feature vector. A reference in direction is [Bunke and Riesen, 2011a].

For extracting more topological information from graphs and to include it in the FMGE, graph paths and Morgan index [Morgan, 1965] are interesting pointers for future research.

The scenarios where learning set is available and FMGE adapts its parameter to the graphs in the learning set, the detection of outliers for cleaning the learning set is important. This is another important challenge for future research.

To take the subgraph spotting work forward, future research work can improve the quality of indexing of the graph repository by exploring cliques of higher order (≥ 3) and to build a multi-resolution index of a graph repository. This will index the graphs on the basis of substructures of different sizes, which will improve the quality of subgraph spotting by removing false positives as we move down to the resolution defined by lower order cliques.

A very interesting future lines of research, for addressing the current main limitation of graph embedding methods in general, is to try to get the surjective matching of the nodes of two attributed graphs that have been compared in feature vector space. This has been illustrated for two example graphs in Figure 6.1.

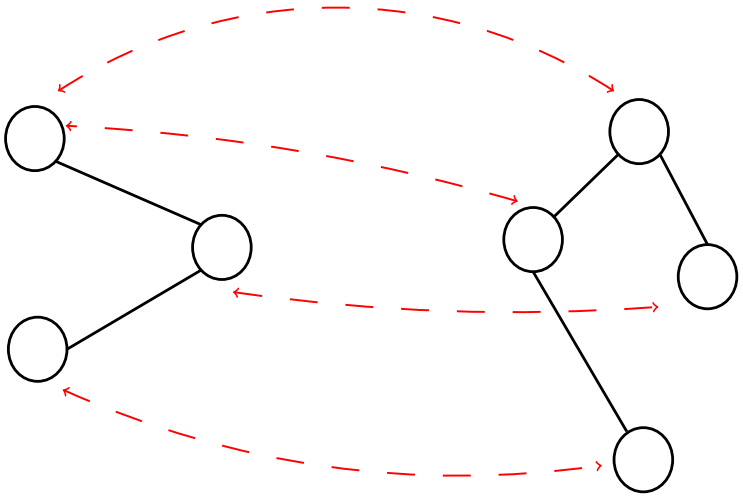


Figure 6.1: Bijective match of nodes of two graphs.

Appendix

Appendix A

Graph databases

A.1 IAM graph database repository

Six datasets from IAM graph database repository, proposed by [Riesen and Bunke, 2010c], have been employed for our experimentation. These graph datasets are publicly available[1]. The IAM graph database repository contains graphs from the field of document image analysis and graphics recognition. The datasets in IAM graph database repository are detailed in subsequent subsections. The example images are reproduced from [Riesen, 2010].

A.1.1 Letter graphs

The letter graph dataset is comprised of graphs extracted from drawings of 15 capital letters of Roman alphabet that consists of straight lines only viz. A, E, F, H, I, K, L, M, N, T, V, W, X, Y and Z. For each class, a prototype line drawing is manually constructed. An illustration of the prototype line drawings is given in Figure A.1. These prototype drawings are then converted into prototype graphs by representing lines by undirected edges and ending points of lines by nodes. Each node is labeled with a two-dimensional attribute giving its position relative to a reference coordinate system. Edges are unlabeled. In order to test classifiers under different conditions, distortions are applied on the prototype graphs with three different levels of strength. This results in three different versions of this database, including graphs with a low, medium and high level of distortion. These three graph dataset consist of a training set, a validation set, and a test set of size 750 each. The graphs are uniformly distributed over the 15 classes [Bunke and Riesen, 2011b]. Figure A.2, A.3 and A.4 illustrate ten graph instances for each distortion level representing the letter A. Note that letters that undergo distortions of medium and high strength are

[1] `http://www.greyc.ensicaen.fr/iapr-tc15/links.html`

difficult to be recognized, even for a human observer [Riesen, 2010].

Figure A.1: Prototypes of letters A to Z.

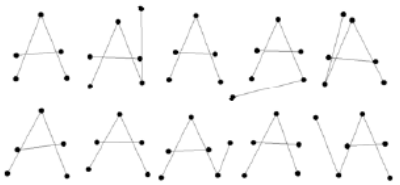


Figure A.2: Instances of letter A at distortion level *low*.

Figure A.3: Instances of letter A at distortion level *medium*.

Figure A.4: Instances of letter A at distortion level *high*.

A.1.2 GREC graphs

Automatic conversion of line drawings from paper to electronic form requires the recognition of geometric primitives like lines, arcs, circles etc. in scanned documents [Riesen, 2010]. The GREC graph dataset is comprised of graphs representing 22 symbols from architectural and electronic drawings. The prototype images of each class are presented in Figure A.5. The images occur at five different distortion levels (Figure A.6). Depending on the distortion level, either erosion, dilation, or other morphological operations are applied. The result is thinned to obtain lines of one pixel width. Finally, graphs are extracted from the resulting denoised images by tracing the lines from end to end and detecting intersections as well as corners. Ending points, corners, intersections and circles are represented by nodes and labeled with a two-dimensional attribute giving their position. The nodes are connected by undirected edges which are labeled as line or arc. An additional attribute specifies the angle with respect to the horizontal direction or the diameter in case of arcs. For an adequately sized set, the five graphs per distortion level are individually distorted 30 times to obtain a data set containing 3300 graphs uniformly distributed over the 22 classes. These distortions consists of translations and scalings of the graphs in a certain range, and random deletions and insertions of both nodes and edges. The resulting set is split into a training and a validation set of size 836 each, and a test set of size 1628 [Bunke and Riesen, 2011b].

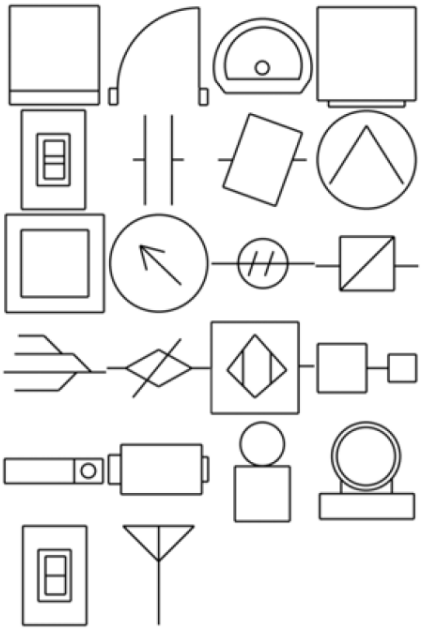


Figure A.5: The prototype images of the 22 GREC classes.

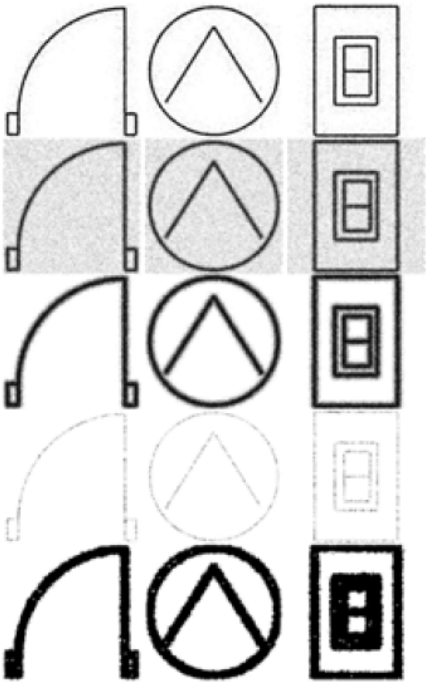


Figure A.6: The five distortion levels (bottom to top) applied to three sample images.

A.1.3 Fingerprint graphs

The Fingerprint data set stems from the emerging field of biometric person authentication. Biometric person authentication refers to the task of automatically recognizing the identity of a person from his or her physiological or behavioral characteristics. Fingerprint images are particularly interesting as a biometric measurement since each person possess this biometric, persons are believed to have unique fingerprints, and fingerprints do not change over time. Moreover, fingerprint images are rather easy to be captured by sensors [Riesen, 2010]. Fingerprints are converted into graphs by filtering the images and extracting regions that are relevant. In order to obtain graphs from fingerprint images, the relevant regions are binarized and a noise removal and thinning procedure is applied. This results in a skeletonized representation of the extracted regions. Ending points and bifurcation points of the skeletonized regions are represented by nodes. Additional nodes are inserted in regular intervals between ending points and bifurcation points. Finally, undirected edges are inserted to link nodes that are directly connected through a ridge in the skeleton. Each node is labeled with a two-dimensional attribute giving its position. The edges are attributed with an angle denoting the orientation of the edge with respect to the horizontal direction. The fingerprint dataset consists of a training set of size 500, a validation set of size 300, and a test set of size 2000. Thus, there are 2800 fingerprint images totally; out of the four classes arch, left, right, and whorl from the Galton-Henry classification system [Riesen and Bunke, 2010c]. The example fingerprint images for these four classes are presented in Figure A.7, Figure A.8, Figure A.9 and Figure A.10 respectively.

A.1.4 Mutagenicity graphs

Mutagenicity is the ability of a chemical compound to cause mutations in DNA and is therefore one of the numerous adverse properties of a compound that hampers its potential to become a marketable drug. Mutagenic compounds pose a toxic risk to humans and screening of drug candidates for mutagenicity is a regulatory requirement for drug approval [Riesen, 2010]. The molecules are converted into attributed graphs in a straightforward manner by representing atoms as nodes and the covalent bonds as edges. Nodes are labeled with the number of the corresponding chemical symbol and edges by the valence of the linkage. The mutagenicity data set is divided into two classes mutagen and nonmutagen. We use a training set of size 1500, a validation set of size 500, and a test set of size 2337. Thus, there are 4337 elements totally (2401 mutagen elements and 1936 nonmutagen elements) [Riesen and Bunke, 2010c].

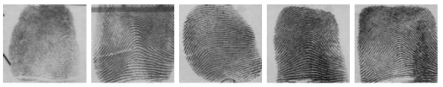


Figure A.7: Fingerprint examples from the Galton-Henry class *Arch*.

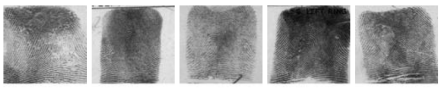


Figure A.8: Fingerprint examples from the Galton-Henry class *Left loop*.

Figure A.9: Fingerprint examples from the Galton-Henry class *Right loop*.

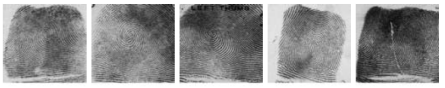


Figure A.10: Fingerprint examples from the Galton-Henry class *Whorl*.

A.2 GEPR graphs

The GEPR graph database consists of graphs extracted from three large publicly available image databases viz. the Amsterdam Library of Object Images (ALOI), the Columbia Object Image Library (COIL) and the Object Databank by Carnegie-Mellon University (ODBK). The Amsterdam Library of Object Images is a collection of images of 1000 small objects. Each object has been acquired several times changing the orientation and the illumination, for a total of 110250 images. Figure A.11 presents some example images. GEPR graph database is composed of 50 objects from ALOI and 72 views for each object (i.e. 3600 images). The Columbia Object Image Library is a collection of images of 100 small objects, each acquired from 72 different orientations. Figure A.12 presents some example images from COIL. GEPR graph database contains 50 objects with 72 views each, from COIL (i.e. 3600 images). The Object Data Bank has been obtained from a collection of 209 3D object models, that have been rendered with photo-realistic quality using 14 different view points. Figure A.13 presents some examples of the pictures from ODBK. The GEPR graph database contains 208 objects and 12 of the 14 view points for each, from ODBK (i.e. 2496 images). The images of each database have been divided into a first set that has been distributed in order to tune the algorithms, and a second set for the performance evaluation [Foggia and Vento, 2010].

Each image has been smoothed using a Gaussian filter, and then it has been segmented using a Pyramidal segmentation algorithm. Finally, from the segmentation the Region Adjacency Graph (RAG) has been constructed as the image graph-based representation. The nodes of the graph have as attributes the relative size and the average color components of the corresponding regions, while the edges of the graph have no attributes [Foggia and Vento, 2010]. The example images of ALOI, COIL and ODBK are reproduced from [Foggia and Vento, 2010].

Figure A.11: Some examples of the images in the ALOI database.

Figure A.12: Some examples of the images in the COIL database.

Figure A.13: Some examples of the images in the ODBK database.

Appendix B

Graphs representation of graphic document images

A new graph repository is constructed by extracting graphs from images of architectural floor plans and electronic diagrams in SESYD image dataset [Delalandre et al., 2010], for experimentation of the framework for automatic indexing of graph repositories for graph retrieval and subgraph spotting. The corresponding graph repository is made publicly available[1] for academia and scientific community for research purposes.

The SESYD graph database consists of graph extracted from line drawing document images in SESYD dataset [Delalandre et al., 2010]. The SESYD dataset contains 100 synthetically generated line drawing document images of architectural floor plans and electronic diagrams each, for 10 different levels of degradations. The architectural floor plans and the electronic diagrams are composed of 16 and 21 unique symbol models respectively. For architectural floor plans a total of 200 document images were selected. These document images are comprised of 4216 symbols which come from 16 different classes of symbols. An example architectural floor plan is presented in Figure B.2. On the other hand, 800 electronic diagram images were selected for constructing our graph dataset. The electronic diagrams are comprised of a total of 9600 symbols which belong to 21 different classes. Figure B.3 presents an example electronic diagram image. The query images simulate the contextual noise. This type of noise occurs in cropped regions of graphic document images. An interesting application of the latter is selecting a region in a graphic document image (on a modern tactile interface), for querying a graphic document content based image retrieval (CBIR) system. The query images are synthetically generated with three different levels of noise; simulating different levels of contextual noise. Figure B.4 presents example query images.

The document images in SESYD dataset were represented by attributed graphs by

[1] `http://www.rfai.li.univ-tours.fr/PagesPerso/mmluqman/public/SESYD_graphs.zip`

using the work of Qureshi et al. [Qureshi et al., 2007]. The graph extraction phase is illustrated in Figure B.1 for a part of graphic document image. The topological and geometric details about structure of graphic content are extracted and are represented by an attributed relational graph (ARG). In first step, the graphic content is vectorized and is represented by a set of primitives (labels 1, 2, 3, 4 in Figure B.1). In next step, these primitives become nodes and topological relations between them become arcs in ARG. Nodes have *relative length* (normalized between 0 and 1) and *primitive-type* (Vector for filled regions of shape and Quadrilateral for thin regions) as attributes; whereas arcs of the graph have *connection-type* (L, X, T, P, S) and *relative angle* (normalized between 0° and 90°) as attributes.

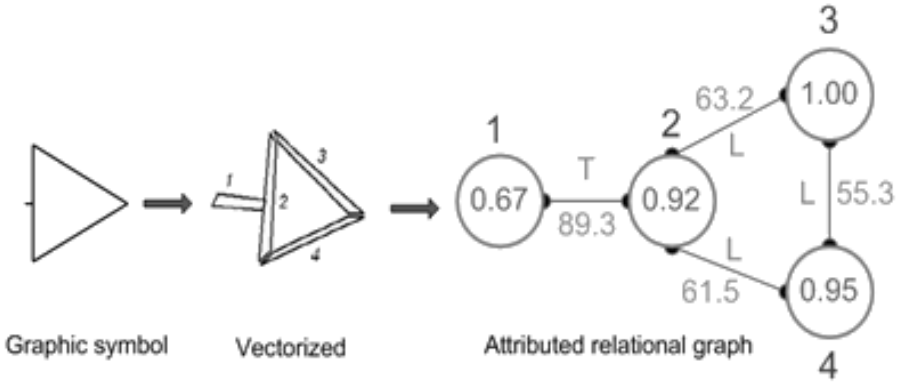


Figure B.1: Representing graphic content by an attributed graph.

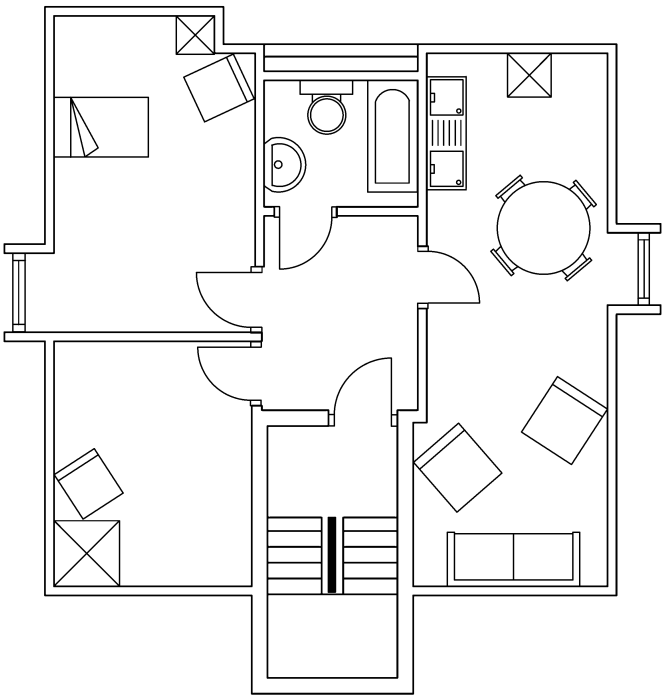


Figure B.2: An example architectural floor plan image from SESYD dataset.

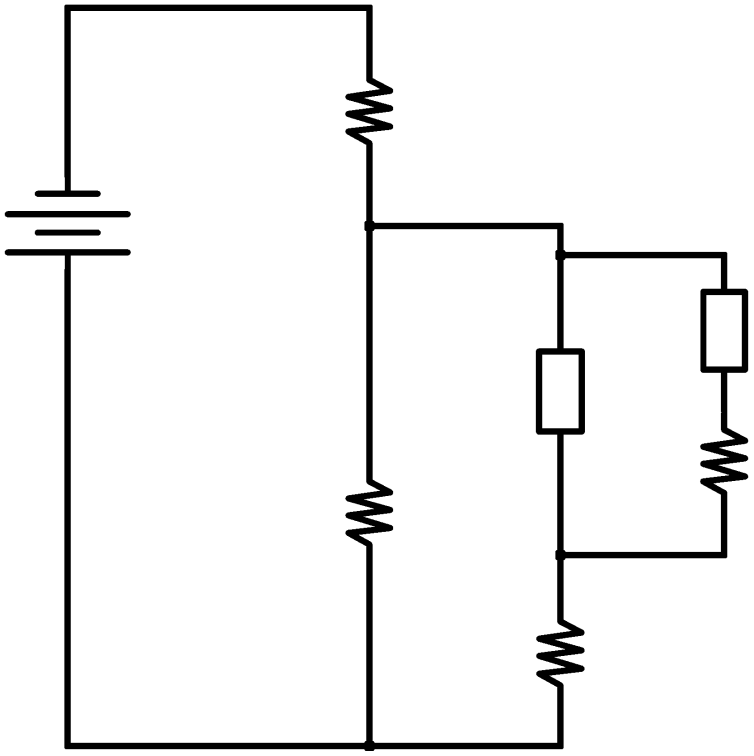


Figure B.3: An example electronic diagram image from SESYD dataset.

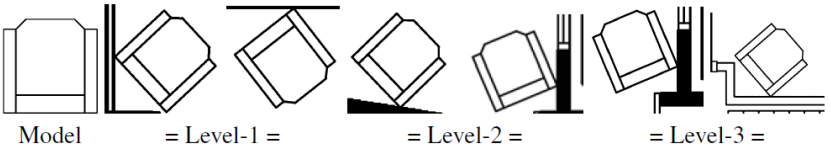


Figure B.4: An arm chair with 2 examples of each different level of contextual noise.

Bibliography

[A. and Zadeh, 2008] A., L. and Zadeh (2008). Is there a need for fuzzy logic? *Information Sciences*, 178(13):2751–2779.

[Borges, 1996] Borges, P. S. d. S. (1996). *A model of strategy games based on the paradigm of the Iterated Prisoner's Dilemma employing Fuzzy Sets*. PhD thesis, Universidade Federal de Santa Catarina.

[Brouard et al., 2010] Brouard, T., Delaplace, A., Luqman, M. M., Cardot, H., and Ramel, J.-Y. (2010). Design of evolutionary methods applied to the learning of Bayesian network structures. In *Bayesian Network*, pages 13–38.

[Brunner and Brunnett, 2004] Brunner, D. and Brunnett, G. (2004). Mesh Segmentation Using the Object Skeleton Graph. In *International Conference on Computer Graphics and Imaging*, pages 48–55.

[Bunke, 1997] Bunke, H. (1997). On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters.*

[Bunke, 1999] Bunke, H. (1999). Error correcting graph matching : On the influence of the underlying cost function. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[Bunke et al., 2002] Bunke, H., Foggia, P., Guidobaldi, C., Sansone, C., and M., V. (2002). A comparison of algorithms for maximum common subgraph on randomly connected graphs. In *International Workshops on Structural, Syntactic, and Statistical Pattern Recognition*.

[Bunke et al., 2001] Bunke, H., Gunter, S., and Jiang, X. (2001). Towards bridging the gap between statistical and structural pattern recognition: Two new concepts in graph matching. In *International Conference on Advances in Pattern Recognition*, pages 1–11. Springer.

[Bunke et al., 2005] Bunke, H., Irniger, C., and Neuhaus, M. (2005). Graph Matching - Challenges and Potential Solutions. In *International Conference on Image Analysis and Processing*, pages 1–10.

[Bunke and Riesen, 2011a] Bunke, H. and Riesen, K. (2011a). Improving vector space embedding of graphs through feature selection algorithms. *Pattern Recognition*, 44(9):1928–1940.

[Bunke and Riesen, 2011b] Bunke, H. and Riesen, K. (2011b). Recent advances in graph-based pattern recognition with applications in document analysis. *Pattern Recognition*, 44(5):1057–1067.

[Bunke and Shearer, 1998] Bunke, H. and Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*.

[Byun, 2003] Byun, H. (2003). A survey on pattern recognition applications of support vector machines. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(3):459–486.

[Caelli and Kosinov, 2004] Caelli, T. and Kosinov, S. (2004). Inexact graph matching using eigen-subspace projection clustering. *International Journal of Pattern Recognition and Artificial Intelligence*.

[Carcassoni and Hancock, 2001] Carcassoni, M. and Hancock, E. R. (2001). Weighted graph-matching using modal clusters. In *9th International Conference on Computer Analysis of Images and Patterns*.

[Chen et al., 2007] Chen, T., Yang, Q., and Tang, X. (2007). Directed graph embedding. In *International Joint Conference on Artificial Intelligence*, pages 2707–2712.

[Chhabra, 1998] Chhabra, A. (1998). Graphic symbol recognition: An overview. *Graphics Recognition Algorithms and Systems*, pages 68–79.

[Chung, 1997] Chung, F. R. K. (1997). *Spectral Graph Theory*. American Mathematical Society.

[Colot et al., 1994] Colot, O., Courtellemont, P., and El-Matouat, A. (1994). Information criteria and abrupt changes in probability laws. In *Signal Processing VII: Theories and Applications*, pages 1855–1858.

[Conte et al., 2004] Conte, D., Foggia, P., Sansone, C., and Vento, M. (2004). Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298.

[Cordella and Vento, 2000] Cordella, L. and Vento, M. (2000). Symbol recognition in documents: a collection of techniques? *International Journal on Document Analysis and Recognition*, 3(2):73–88.

[Corneil and Gotlie, 1970] Corneil, D. and Gotlie, C. (1970). An Efficient Algorithm for Graph Isomorphism. *Journal of the Association for Computing Machinery*, 17:51–64.

[De Sa, 2001] De Sa, J. (2001). *Pattern recognition: concepts, methods, and applications*. Springer Verlag.

[Delalandre et al., 2010] Delalandre, M., Valveny, E., Pridmore, T., and Karatzas, D. (2010). Generation of synthetic documents for performance evaluation of symbol recognition & spotting systems. *International Journal on Document Analysis and Recognition*, pages 1–21.

[Delaplace et al., 2006] Delaplace, A., Brouard, T., and Cardot, H. (2006). Two evolutionary methods for learning Bayesian network structures. *2006 International Conference on Computational Intelligence and Security*, pages 137–142.

[Dosch and Valveny, 2006] Dosch, P. and Valveny, E. (2006). Report on the Second Symbol Recognition Contest. In Liu, W. and Lladós, J., editors, *Graphics Recognition. Ten Years Review and Future Perspectives*, volume 3926 of *Lecture Notes in Computer Science*, pages 381–397. Springer Berlin / Heidelberg.

[Duda et al., 2000] Duda, R., Hart, P., and Stork, D. (2000). *Pattern classification*, volume 2. Wiley Interscience.

[Eshera and Fu, 1984] Eshera, M. and Fu, K. (1984). A graph distance measure for image analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 14:394–408.

[Ferrer et al., 2008] Ferrer, M., Valveny, E., Serratosa, F., Riesen, K., and Bunke, H. (2008). An approximate algorithm for median graph computation using graph embedding. In *International Conference on Pattern Recognition*, pages 1–4. Ieee.

[Ferrer et al., 2010] Ferrer, M., Valveny, E., Serratosa, F., Riesen, K., and Bunke, H. (2010). Generalized median graph computation by means of graph embedding in vector spaces. *Pattern Recognition*, 43(4):1642–1655.

[Fischler and Elschlager, 1973] Fischler, M. and Elschlager, R. (1973). The representation and matching of pictorial structures. *IEEE Transactions on Computers*.

[Foggia and Vento, 2010] Foggia, P. and Vento, M. (2010). Graph Embedding for Pattern Recognition. In Ünay, D., Çataltepe, Z., and Aksoy, S., editors, *Recognizing Patterns in Signals, Speech, Images and Videos*, volume 6388 of *Lecture Notes in Computer Science*, pages 75–82. Springer.

[Franco et al., 2003] Franco, P., Ogier, J.-M., Loonis, P., and Mullot, R. (2003). A Topological Measure for Image Object Recognition. In *Lecture Notes in Computer Science, volume 3088*, pages 279–290.

[Friedman and Kandel, 1999] Friedman, M. and Kandel, A. (1999). *Introduction to pattern recognition*. World Scientific.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[Gibert et al., 2011a] Gibert, J., Valveny, E., and Bunke, H. (2011a). Dimensionality Reduction for Graph of Words Embedding. In *Graph-Based Representations in Pattern Recognition*, pages 22–31.

[Gibert et al., 2011b] Gibert, J., Valveny, E., and Bunke, H. (2011b). Dimensionality Reduction for Graph of Words Embedding. In *LNCS 6658*, pages 22–31.

[Gibert et al., 2011c] Gibert, J., Valveny, E., and Bunke, H. (2011c). Vocabulary Selection for Graph of Words Embedding. In *5th Iberian Conference on Pattern Recognition and Image Analysis*, pages 216–223. LNCS. Berlin: Springer, 6669 edition.

[Godsil and Royle, 2011] Godsil, C. and Royle, G. (2011). *Algebraic Graph Theory*. Springer.

[Gori et al., 2005] Gori, M., Maggini, M., and Sarti, L. (2005). Exact and approximate graph matching using random walks. *IEEE transactions on pattern analysis and machine intelligence*, 27.

[Harchaoui, 2007] Harchaoui, Z. (2007). Image classification with segmentation graph kernels. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Hart et al., 1968] Hart, P., Nilsson, N., and Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems, Man and Cybernetics*.

[Inokuchi et al., 2000] Inokuchi, A., Washio, T., and Motoda, H. (2000). An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. *Lecture Notes in Computer Science*, 1910:13–23.

[Ishibuchi and Yamamoto, 2003] Ishibuchi, H. and Yamamoto, T. (2003). Deriving fuzzy discretization from interval discretization. In *International Conference on Fuzzy Systems*, pages 749–754. IEEE.

[Jain et al., 1999] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering : a review. *ACM Computing Surveys*.

[Jiang and Bunke, 1999] Jiang, X. and Bunke, H. (1999). Optimal Quadratic-Time Isomorphism of Ordered Graphs. *Pattern Recognition*, 32:1273–1283.

[Kaufman and Rousseeuw, 1990] Kaufman, L. and Rousseeuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*, volume 39 of *Wiley Series in Probability and Mathematical Statistics*. John Wiley & Sons.

[Klir and Yuan, 1995] Klir, G. and Yuan, B. (1995). *Fuzzy sets and fuzzy logic*. Prentice-Hall, Englewood Cliffs.

[Köbler et al., 1993] Köbler, J., Schöning, U., and Torán, J. (1993). *The graph isomorphism problem : its structural complexity*. Birkhauser Verlag.

[Kosinov and Caelli, 2002] Kosinov, S. and Caelli, T. (2002). Inexact multisubgraph matching using graph eigenspace and clustering models. In *SSPR/SPR*, pages 133–142.

[Kramer and Raedt, 2001] Kramer, S. and Raedt, L. (2001). Feature construction with version spaces for biochemical application. In *18th International Conference on Machine Learning*, pages 258–265.

[Kuncheva, 2004] Kuncheva, L. (2004). *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley.

[Lee and Madabhushi, 2010] Lee, G. and Madabhushi, A. (2010). Semi-Supervised Graph Embedding Scheme with Active Learning (SSGEAL): Classifying High Dimensional Biomedical Data. In *Pattern Recognition in Bioinformatics*, volume 6282 of *Lecture Notes in Computer Science*, pages 207–218. Springer.

[Leray and Francois, 2004] Leray, P. and Francois, O. (2004). BNT Structure Learning Package : Documentation and Experiments. *Structure*, (November).

[Levenshtein, 1966] Levenshtein, V. I. (1966). *Binary codes capable of correcting deletions, insertions and reversals*. Soviet Physics Doklady.

[Liu, 2010] Liu, B. (2010). Uncertainty Theory. In *Uncertainty Theory*, volume 300 of *Studies in Computational Intelligence*, pages 1–79. Springer Berlin / Heidelberg.

[Liu et al., 2002] Liu, H., Hussain, F., Tan, C., and Dash, M. (2002). Discretization: An enabling technique. *Data Mining and Knowledge*, pages 393–423.

[Llados and Sanchez, 2003] Llados, J. and Sanchez, G. (2003). Symbol recognition using graphs. In *International Conference on Image Processing*, volume 2, pages 49–52. IEEE.

[Lladós et al., 2002] Lladós, J., Valveny, E., Sánchez, G., and Martí, E. (2002). Symbol Recognition: Current Advances and Perspectives. In *Graphics Recognition Algorithms and Applications*, volume 2390, pages 104–128.

[Lopresti and Wilfong, 2003] Lopresti, D. and Wilfong, G. (2003). A fast technique for comparing graph representations with applications to performance evaluation. *International Journal of Document Analysis and Recognition*.

[Luo et al., 2003] Luo, B., Wilson, R., and Hancock, E. (2003). Spectral embedding of graphs. *Pattern Recognition*, 36(10):2213–2230.

[Luqman et al., 2010a] Luqman, M., Lladós, J., Ramel, J.-Y., and Brouard, T. (2010a). A Fuzzy-Interval Based Approach for Explicit Graph Embedding. In *Recognizing Patterns in Signals, Speech, Images and Videos*, volume 6388, pages 93–98.

[Luqman et al., 2009a] Luqman, M. M., Brouard, T., and Ramel, J.-Y. (2009a). Graphic Symbol Recognition using Graph Based Signature and Bayesian Network Classifier. In *Tenth International Conference on Document Analysis and Recognition (ICDAR)*, pages 1325–1329, Barcelona. IEEE Computer Society.

[Luqman et al., 2010b] Luqman, M. M., Brouard, T., Ramel, J.-y., and Llados, J. (2010b). A Content Spotting System For Line Drawing Graphic Document Images. In *20th International Conference on Pattern Recognition*, volume 20, pages 3420–3423.

[Luqman et al., 2010c] Luqman, M. M., Brouard, T., Ramel, J.-Y., and Llados, J. (2010c). Vers une approche floue d encapsulation de graphes : application à la reconnaissance de symboles. In *Colloque International Francophone sur l'Ecrit et le Document*, pages 169–184.

[Luqman et al., 2012] Luqman, M. M., Brouard, T., Ramel, J.-Y., and Llados, J. (2012). Recherche de sous-graphes par encapsulation floue des cliques dordre 2: Application à la localisation de contenu dans les images de documents graphiques. In *Colloque International Francophone sur l'Ecrit et le Document*, page accepted.

[Luqman et al., 2009b] Luqman, M. M., Delalandre, M., Brouard, T., Ramel, J.-Y., and Lladós, J. (2009b). Employing fuzzy intervals and loop-based methodology for designing structural signature : an application to symbol recognition. In *GREC*, pages 22–31.

[Luqman et al., 2010d] Luqman, M. M., Delalandre, M., Brouard, T., Ramel, J.-Y., and Lladós, J. (2010d). Fuzzy Intervals for Designing Structural Signature: An Application to Graphic Symbol Recognition. In Ogier, J.-M., Liu, W., and Lladós, J., editors, *Graphics Recognition. Achievements, Challenges, and Evolution*, volume 6020, pages 12–24. Springer Berlin / Heidelberg.

[Luqman et al., 2011a] Luqman, M. M., Lladós, J., Ramel, J.-Y., and Brouard, T. (2011a). Dimensionality Reduction for Fuzzy-Interval Based Explicit Graph Embedding. In *GREC*, pages 117–120.

[Luqman et al., 2011b] Luqman, M. M., Ramel, J.-Y., Llados, J., and Brouard, T. (2011b). Subgraph Spotting through Explicit Graph Embedding : An Application to Content Spotting in Graphic Document Images. In *Eleventh International Conference on Document Analysis and Recognition (ICDAR)*, pages 870–874.

[McGregor, 1982] McGregor, J. J. (1982). Backtrack search algorithms and the maximal common subgraph problem. *Software Practice and Experience*.

[Messmer, 1995] Messmer, B. (1995). *Efficient Graph Matching Algorithms*. PhD thesis, University of Bern, Switzerland.

[Messmer and Bunke, 1998] Messmer, B. and Bunke, H. (1998). Error-Correcting Graph Isomorphism using Decision Trees. *Int. Journal of Pattern Recognition and Art. Intelligence*, 12:721–742.

[Morgan, 1965] Morgan, H. L. (1965). The Generation of a Unique Machine Description for Chemical Structures - A Technique Developed at Chemical Abstracts Service. *Journal of Chemical Documentation*, 5(2):107–113.

[Neuhaus and Bunke, 2004] Neuhaus, M. and Bunke, H. (2004). A probabilistic approach to learning costs for graph edit distance. In *17th International Conference on Pattern Recognition*.

[Neuhaus and Bunke, 2007] Neuhaus, M. and Bunke, H. (2007). Automatic learning of cost functions for graph edit distance. *Information Sciences*.

[Neuhaus, M. et Bunke, 2006] Neuhaus, M. et Bunke, H. (2006). Edit distance-based kernel functions for structural pattern classification. *Pattern Recognition*.

[Okada et al., 2005] Okada, Y., Sahara, T., Ohgiya, S., and Nagashima, T. (2005). Detection of cluster boundary in microarray data by reference to mips functional catalogue database. In *International Conference on Genome Informatics*, pages 2–3.

[Papadopoulos, A. N. et Manolopoulos, 1999] Papadopoulos, A. N. et Manolopoulos, Y. (1999). Structure-based similarity search with graph histograms. In *International Workshop on Database and Expert Systems Applications*.

[Pavlidis, 1972] Pavlidis, T. (1972). Representation of Figures by Labeled Graphs. *Pattern Recognition*, 4:5–17.

[Pekalska and Duin, 2005] Pekalska, E. and Duin, R. P. W. (2005). *The Dissimilarity Representation for Pattern Recognition : Foundations And Applications*. World Scientific Publishing.

[Pelillo, 1999] Pelillo, M. (1999). Replicator equations, maximal cliques, and graph isomorphism. *Neural Computation*.

[Qureshi, 2008] Qureshi, R. J. (2008). *Reconnaissance de formes et symboles graphiques complexes dans les images de documents*. PhD thesis, Université François-Rabelais de Tours, France.

[Qureshi et al., 2007] Qureshi, R. J., Ramel, J.-Y., Cardot, H., and Mukherji, P. (2007). Combination of Symbolic and Statistical Features for Symbols Recognition. In *International Conference on Signal Processing, Communications and Networking*, pages 477–482.

[Radzikowska and Kerre, 2002] Radzikowska, A. M. and Kerre, E. E. (2002). A comparative study of fuzzy rough sets. *Fuzzy Sets and Systems*, 126(2):137–155.

[Ramel, 1992] Ramel, J.-Y. (1992). *Lecture Automatique de Partitions Musicales*. PhD thesis, INSA Lyon, France.

[Read and Corneil, 1977] Read, R. C. and Corneil, D. G. (1977). The graph isomorphism disease. *Journal of Graph Theory*.

[Reingold et al., 1997] Reingold, E., Nievergelt, J., and Deo, N. (1997). *Combinatorial algorithms : theory and practice*. Prentice-Hall.

[Riesen, 2010] Riesen, K. (2010). *Classification and clustering of Vector Space Embedded Graphs*. PhD thesis, University of Bern, Switzerland.

[Riesen and Bunke, 2009] Riesen, K. and Bunke, H. (2009). Graph classification based on vector space embedding. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(6):1053–1081.

[Riesen and Bunke, 2010a] Riesen, K. and Bunke, H. (2010a). *Graph Classification and Clustering Based on Vector Space Embedding*. World Scienti c.

[Riesen and Bunke, 2010b] Riesen, K. and Bunke, H. (2010b). *Graph Classification And Clustering Based On Vector Space Embedding*. World Scientific Publishing Co., Inc.

[Riesen and Bunke, 2010c] Riesen, K. and Bunke, H. (2010c). IAM graph database repository for graph based pattern recognition and machine learning. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 287–297. Springer.

[Riesen et al., 2007] Riesen, K., Neuhaus, M., and Bunke, H. (2007). Graph embedding in vector spaces by means of prototype selection. In *International Conference on Graph-based Representations in Pattern Recognition*, pages 383–393. Springer-Verlag.

[Robles-Kelly and Hancock, 2007] Robles-Kelly, A. and Hancock, E. (2007). A Riemannian approach to graph embedding. *Pattern Recognition*, 40(3):1042–1056.

[Roth et al., 2003] Roth, V., Laub, J., Kawanabe, M., and Buhmann, J. (2003). Optimal cluster preserving embedding of nonmetric proximity data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1540–1551.

[Selkow, 1977] Selkow, S. (1977). The tree-to-tree editing problem. *Information Processing Letters*.

[Shaw and Jebara, 2009] Shaw, B. and Jebara, T. (2009). Structure preserving embedding. *International Conference on Machine Learning*, pages 1–8.

[Shawe-Taylor and Cristianini, 2004] Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge University Press.

[Shokoufandeh et al., 2005] Shokoufandeh, A., Macrini, D., Dickinson, S., Siddiqi, K., and Zucker, S. (2005). Indexing hierarchical structures using graph spectra. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1125–1140.

[Sidère, 2012] Sidère, N. (2012). *Contribution aux méthodes de reconnaissance structurelle de formes : approche à base de projections de graphes*. PhD thesis, Université François-Rabelais de Tours, France.

[Sidère et al., 2008] Sidère, N., Héroux, P., and Ramel, J.-Y. (2008). A Vectorial Representation for the Indexation of Structural Informations, in Structural, Syntactic, and Statistical Pattern Recognition. In *Lecture Notes in Computer Science, 5342*, pages 45–54.

[Sidère et al., 2009a] Sidère, N., Héroux, P., and Ramel, J.-Y. (2009a). Embedding labeled graphs into occurence matrix. In *8th International Workshop on Graphics Recognition*, pages 44–50.

[Sidère et al., 2009b] Sidère, N., Héroux, P., and Ramel, J.-Y. (2009b). Vector Representation of Graphs: Application to the Classification of Symbols and Letters. *International Conference on Document Analysis and Recognition*, pages 681–685.

[Solnon, 2010] Solnon, C. (2010). All different-based filtering for subgraph isomorphism. *Artificial Intelligence*.

[Sonbaty and Ismail, 1998] Sonbaty, Y.-E. and Ismail, M. (1998). A New Algorithm for Subgraph Optimal Isomorphism. *Pattern Recognition*, 31(2):205–218.

[Tai, 1979] Tai, K.-C. (1979). The tree-to-tree correction problem. *Journal of the Association for Computing Machinery*.

[Tombre et al., 2006] Tombre, K., Tabbone, S., and Dosch, P. (2006). Musings on symbol recognition. In *Graphics Recognition. Ten Years Review and Future Perspectives*, pages 23–34.

[Tsai and Fu, 1979] Tsai, W. and Fu, K. (1979). Error-correcting isomorphisms of attributed relational graphs for pattern analysis. *IEEE Transaction on Systems, Man and Cybernetics*.

[Ullman, 1976] Ullman, J. R. (1976). An Algorithm for Subgraph Isomorphism. *Journal of the Association for Computing Machinery*, 23(1):31–42.

[Umeyama, 1988] Umeyama, S. (1988). An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Valveny and Dosch, 2004] Valveny, E. and Dosch, P. (2004). Symbol recognition contest: A synthesis. *Graphics Recognition*, pages 368–385.

[Wiener, 1947] Wiener, H. (1947). Structural determination of paraffin boiling points. *Journal of the American Chemical Society*.

[Wilson et al., 2005] Wilson, R. C., Hancock, E. R., and Luo, B. (2005). Pattern vectors from algebraic graph theory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1112–24.

[Winston, 1970] Winston, P. (1970). Learning structural descriptions from examples. Technical report, Massachusetts Institute of Technology.

List of publications

The following publications are a direct consequence of the research carried out during the elaboration of this thesis. They give a clear idea of the progression that has been achieved over the span of thesis lifetime.

Journal paper

Muhammad Muzzamil Luqman, Jean-Yves Ramel, Josep Lladós and Thierry Brouard. (2012). Fuzzy Multilevel Graph Embedding. under review, submitted December 2011. *Pattern Recognition*.

Book chapter

Thierry Brouard, Alain Delaplace, Muhammad Muzzamil Luqman, Hubert Cardot, and Jean-Yves Ramel. (2010). Design of evolutionary methods applied to the learning of Bayesian network structures. *in Bayesian Network*, ISBN: 978-953-307-124-4, Sciyo, pp. 13-38.

International conference contributions

Muhammad Muzzamil Luqman, Jean-Yves Ramel, Josep Lladós and Thierry Brouard. (2011). Subgraph Spotting through Explicit Graph Embedding: An Application to Content Spotting in Graphic Document Images. *in Eleventh International Conference on Document Analysis and Recognition (ICDAR)*, volume 11, IEEE Computer Society, pp. 870-874.

Muhammad Muzzamil Luqman, Thierry Brouard, Jean-Yves Ramel and Josep Lladós. (2010). A Content Spotting System For Line Drawing Graphic Document Images. *in Twentieth International Conference on Pattern Recognition (ICPR)*, volume 20, IEEE Computer Society, pp. 3420-3423.

Muhammad Muzzamil Luqman, Thierry Brouard and Jean-Yves Ramel. (2009). Graphic Symbol Recognition using Graph Based Signature and Bayesian Network Classifier. *in Tenth International Conference on Document Analysis and Recognition (ICDAR)*, volume 10, IEEE Computer Society, pp. 1325-1329.

Selected papers for post-workshop LNCS publication

Muhammad Muzzamil Luqman, Josep Lladós, Jean-Yves Ramel and Thierry Brouard. (2010). A Fuzzy-Interval Based Approach For Explicit Graph Embedding. *in Lecture Notes in Computer Science, Recognizing Patterns in Signals, Speech, Images, and Videos*, volume 6388, pp. 93-98.

Muhammad Muzzamil Luqman, Mathieu Delalandre, Thierry Brouard, Jean-Yves Ramel and Josep Lladós. (2010). Fuzzy Intervals for Designing Structural Signature: An Application to Graphic Symbol Recognition. *in Lecture Notes in Computer Science, Graphics Recognition. Achievements, Challenges, and Evolution*, volume 6020, pp. 12-24.

International workshop contributions

Muhammad Muzzamil Luqman, Josep Lladós, Jean-Yves Ramel and Thierry Brouard. (2011). Dimensionality Reduction for Fuzzy-Interval Based Explicit Graph Embedding. *in Ninth IAPR International Workshop on Graphics RECognition (GREC)*, volume 9, pp. 117-120.

Muhammad Muzzamil Luqman, Mathieu Delalandre, Thierry Brouard, Jean-Yves Ramel and Josep Lladós. (2009). Employing fuzzy intervals and loop-based methodology for designing structural signature: an application to symbol recognition. *in Eighth IAPR International Workshop on Graphics RECognition (GREC)*, volume 8, p. 22-31.

Francophone conference contributions

Muhammad Muzzamil Luqman, Thierry Brouard, Jean-Yves Ramel and Josep Lladós. (2012). Recherche de sous-graphes par encapsulation floue des cliques d'ordre 2: Application á la localisation de contenu dans les images de documents graphiques. *in Colloque International Francophone sur l'Ecrit et le Document*, to appear (accepted).

Muhammad Muzzamil Luqman, Thierry Brouard, Jean-Yves Ramel and Josep Lladós. (2010). Vers une approche floue d'encapsulation de graphes: application á la reconnaissance de symboles. *in Colloque International Francophone sur l'Ecrit et le Document*, pp. 169-184.

Abstract :

This thesis addresses the problem of lack of efficient computational tools for graph based structural pattern recognition approaches and proposes to exploit computational strength of statistical pattern recognition. It has two fold contributions. The first contribution is a new method of explicit graph embedding. The proposed graph embedding method exploits multilevel analysis of graph for extracting graph level information, structural level information and elementary level information from graphs. It embeds this information into a numeric feature vector. The method employs fuzzy overlapping trapezoidal intervals for addressing the noise sensitivity of graph representations and for minimizing the information loss while mapping from continuous graph space to discrete vector space. The method has unsupervised learning abilities and is capable of automatically adapting its parameters to underlying graph dataset. The second contribution is a framework for automatic indexing of graph repositories for graph retrieval and subgraph spotting. This framework exploits explicit graph embedding for representing the cliques of order 2 by numeric feature vectors, together with classification and clustering tools for automatically indexing a graph repository. It does not require a labeled learning set and can be easily deployed to a range of application domains, offering ease of query by example (QBE) and granularity of focused retrieval.

Keywords :

Pattern recognition, graph clustering, graph classification, graph embedding, subgraph spotting, fuzzy logic, graphics recognition.

Résumé :

Cette thése aborde le probléme du manque de performance des outils exploitant des représentations á base de graphes en reconnaissance des formes. Nous proposons de contribuer aux nouvelles méthodes proposant de tirer partie, á la fois, de la richesse des méthodes structurelles et de la rapidité des méthodes de reconnaissance de formes statistiques. Deux principales contributions sont présentées dans ce manuscrit. La premiére correspond á la proposition d'une nouvelle méthode de projection explicite de graphes procédant par analyse multi-facettes des graphes. Cette méthode effectue une caractérisation des graphes suivant différents niveaux qui correspondent, selon nous, aux point-clés des représentations á base de graphes. Il s'agit de capturer l'information portée par un graphe au niveau global, au niveau structure et au niveau local ou élémentaire. Ces informations capturées sont encapsulées dans un vecteur de caractéristiques numériques employant des histogrammes flous. La méthode proposée utilise, de plus, un mécanisme d'apprentissage non supervisée pour adapter automatiquement ses paramètres en fonction de la base de graphes á traiter sans nécessiter de phase d'apprentissage préalable. La deuxième contribution correspond á la mise en place d'une architecture pour l'indexation de masses de graphes afin de permettre, par la suite, la recherche de sous-graphes présents dans cette base. Cette architecture utilise la méthode précédente de projection explicite de graphes appliquée sur toutes les cliques d'ordre 2 pouvantêtre extraites des graphes présents dans la base á indexer afin de pouvoir les classifier. Un partitionnement des cliques permet de constituer l'index qui sert de base á la description des graphes et donc á leur indexation en ne nécessitant aucune base d'apprentissage pré-étiquetées. La méthode proposée est applicable á de nombreux domaines, apportant la souplesse d'un système de requête par l'exemple et la granularité des techniques d'extraction ciblée (focused retrieval).

Mots clés :

Reconnaissance des formes, partitionnement de graphes, classification de graphes, projection de graphes, repérage de sous-graphes, logique floue, reconnaissance de graphiques.