Chapter 1

Introduction

1.1 Graph Matching

The graph is one of the most prominent mathematical structure in computer science. It is defined as a set of nodes and edges, where each edge is a connection between a pair of nodes. Due to its inherent ability to demonstrate the structural representation between objects, it is used in a wide range of applications in science and engineering.

For example, in biological applications, graphs are used as biological networks where a node represents biological units like cells, protein, neuron, etc., and an edge represents the connection between them. In chemical applications graphs are used as a molecular graph, where nodes represent atoms or molecules, and edge denotes valence or bond between chemical units. In computer science, graphs are ubiquitous and are used in almost every area of computing. For example, in the operating system, graphs are used to characterize resource allocation graph in which process and resources are designated by nodes. An edge from resource to process denotes that resource is allocated to process, whereas an edge from process to resource indicates that the process has requested for the corresponding resource. In computer network graphs are used to find the shortest path for routing the data packets over the communication network. In software engineering control flow graph is used to find the complexity of a program and, to envisage the dependency and association between the different components of a software project. In this thesis, we use graphs for matching of structured data.

Graph matching is the process of computing the similarity between two graphs. It is a generalization of tree matching, which itself is a generalization of string matching. While the string is a linear structure, implying that at each successive step only one element like a symbol can be appended to an existing element; on the other hand trees and graphs can be connected to more than one data element. This increase in the representation power from strings to trees and trees to graphs also leads to the increased computational complexity of matching algorithms. While polynomial time algorithms for string and tree matching are known but no polynomial time algorithm is known for graph matching.

Depending on the nature of matching, graph matching is broadly classified into exact and inexact matching. Exact graph matching requires a strict one-to-one correspondence between vertices and edges of two graphs. It is like graph isomorphism problem, where a bijection is required between the vertices of one graph to another one such that for every edge in the first graph there exists a corresponding edge in the second graph which connects the same set of nodes. Exact graph matching even though theoretically important, may not be useful in real-world applications, where input data often get modified due to the presence of noise and error in storage and transmission process. To overcome the effect of noise and distortion on input data, inexact graph matching is used, which finds a similarity score between two graphs. Inexact graph matching is also known as error-tolerant graph matching as it can accommodate some tolerance to noise or error that may have incurred during the processing of data.

A major application of graph matching is in structural pattern recognition. Structural pattern recognition utilizes the underlying structure of the object to perform the various pattern recognition tasks. The ability to identify patterns is one of the fundamental characteristics of human beings and up to certain extents whole of living beings. Pattern recognition consists of analyzing and classifying patterns based on their characteristics and features. Every person deals with a large number of pattern recognition problems in day to day life. Examples include recognition of a relative from a group of persons, identifying a particular book from a book self, identification of streets and friend's home, etc. Because of the complex cognition system of the human brain, the task of recognizing different object seems to be intuitive and very simple, but the same tasks using a machine or computer can be very complicated. Pattern recognition system develops algorithms to perform such tasks automatically using a machine.

Depending on the nature of the underlying problem, pattern recognition can be categorized into statistical and structural pattern recognition. Statistical pattern recognition uses feature vectors to represent different patterns. Use of feature vectors allows standard mathematical techniques, which apply to vector space, are also applicable in statistical pattern recognition domain. Therefore the use of feature vectors leads to many efficient algorithms in statistical pattern recognition. The limitation of the use of feature vectors of fixed dimension is that it can not be used for pattern having structures like strings, trees and graphs. In such situations, structural pattern recognition offers an alternative to statistical pattern recognition by recognizing patterns using graph-based representations instead of feature vectors. The major benefit of using graph is that its representational ability is higher than that of feature vectors. Here we can observe a classic trade-off between using vector and graph. While the efficiency of mathematical operations using vector are high, but their representational power is low, on the other hand, representational power of graphs are high, but the efficiency of various mathematical operations is low. Due to unavailability of efficient polynomial time solutions to graph matching problem, various approximation and suboptimal algorithms have been proposed recently.

1.2 Contribution

In this section, a brief description of the contribution of this thesis is given.

Graph matching using the concept of homeomorphism is introduced. Path contraction is used to remove the nodes with degree two from all simple paths of input graphs in which every node except first and last have degree two. Since path contraction is reverse of subdivision operation, the resulting graphs are homeomorphic and topologically equivalent. We use this concept to define homeomorphic graph edit distance.

Error-tolerant graph matching using node contraction is presented, in which input graphs are transformed by contracting smaller degree nodes. Node contraction is the process of deleting a node and its linked edges provided that the node is not a cut vertex. This approach is used to present the notion of extended graph edit distance. The proposed framework can be used as a trade-off between time versus accuracy considerations. Experimental

results show that the algorithm attains efficiency without altering the topology of the graphs drastically.

A class of graph matching algorithms is proposed, which reduces the graph size by removing the less relevant nodes or edges using some measure of centrality. It reduces the search space by contracting the nodes with least centrality values. Four different centrality measures: degree, betweenness, eigenvector and PageRank are used for comparison. Node contraction can be considered as a special case of this approach, where centrality measure is degree centrality. Depending on the structure and properties of various graph dataset, we can choose the appropriate centrality measure to reduce the size of the graphs. The proposed algorithm is implemented on the letter and molecules dataset and results show that different centrality criteria can be selected to save computation time during graph matching.

A novel approach to measure graph similarity between geometric graphs is introduced. Vertex distance between two geometric graphs is defined using the coordinate positions of the vertices. Edge distance between two geometric graphs is defined using the angular orientation, length and position of edges and the resulting distance is shown to be a metric. Finally, graph distance is defined using the linear combination of vertex distance and edge distance. The resulting graph distance notion is shown to be a metric over the set of all geometric graphs.

Exact and error-tolerant geometric graph matching is described using the notion of the proposed geometric graph similarity metric. A geometric graph isomorphism algorithm is presented to check the isomorphism between two geometric graphs. Prior to checking geometric graph isomorphism, this uses graph alignment algorithm to perform the geometric transformation on a graph so that its reference coordinates are aligned to the input graph. Error-tolerant graph matching using the geometric graph similarity is presented. The input graph size is made equal by appending additional vertices and edges of appropriate values. Weighting parameters are used to combine the vertex distance and the three components of edge distance to compute the final graph distance metric. Proposed graph matching algorithm is both error-tolerant as well as computed in polynomial time. Experimental evaluation shows that the proposed geometric graph matching framework is promising to graph dataset having two-dimensional coordinates for each vertex.

1.3 Organization of the Thesis

This thesis is organized as follows. The present chapter provides an introduction to graph matching and a brief overview of the proposed work.

Chapter 2, provides a survey of exact and error-tolerant graph matching. It also describes the definitions and basic concepts used in exact and inexact graph matching.

Chapter 3, introduces graph matching using various extensions to graph edit distance. It describes the homeomorphic graph edit distance, which uses path contraction as a preprocessing step. It also discusses the error-tolerant graph matching using node contraction and presents the extended graph edit distance.

Chapter 4, presents graph matching utilizing node centrality information to ignore the less relevant nodes and introduces r-centrality graph edit distance. Here r is the fraction of nodes to be contracted from the graph based on given centrality criteria. It uses degree, betweenness, eigenvector and PageRank centrality measures for computing the centrality of nodes in the graphs.

Chapter 5, introduces an intuitive approach to measure the similarity between two geometric graphs. It defines vertex distance using the position of vertices of the two graphs; then it defines edge distance using the alignment, length and position features of edges. Finally, it combines both vertex and edge distance to compute graph distance. It also presents algorithms for exact and error-tolerant graph matching using the proposed geometric graph similarity framework.

Finally, chapter 6, provides the concluding remarks.