Chapter 2

Graph Matching: A Survey

2.1 Introduction

Graph matching is the process of computing the similarity between two graphs. It is broadly classified into exact and inexact graph matching. While exact graph matching has more theoretical implications in computer science, on the other side inexact graph matching has more practical implications in computer science and pattern recognition. For example, an optimal solution to graph isomorphism problem which is an exact graph matching problem will lead to resolution of its complexity class, which is currently neither known to be in class P, nor in NP-complete. On the other hand, the subgraph isomorphism problem is known to be in NP-complete and therefore, no efficient polynomial time algorithm is available.

In this chapter, a review of various exact, approximate and error-tolerant graph matching techniques is provided. Conte *et al.* [1] describe an extensive survey of different exact and inexact graph matching techniques used in pattern recognition. Foggia *et al.* [2] provide a more recent survey of graph matching and learning techniques used in pattern recognition. Bunke [3] presents a formal framework and algorithm for graph matching. Gao *et al.* [4] describe a survey of the various algorithm for graph edit distance. Livi and Rizzi [5] provide a review of graph matching problem focusing methodological and algorithmic results.

This chapter is organized as follows. Section 2.2, presents basic definitions and concepts used in exact and error-tolerant graph matching. Section 2.3, describes a survey of exact graph matching techniques. Section 2.4, presents a review of error-tolerant graph matching methods. Finally, section 2.5 describes the summary.

2.2 Basic Concepts and Definitions

In this section, we review the basic concepts and definitions used in exact and inexact graph matching [6, 7, 8, 9].

Definition 2.2.1 (Graph)**.** A graph G is defined as $G = (V, E, \mu, \nu)$, where

- V is the set of vertices or nodes,
- E is the set of edges or links,
- $\mu : V \rightarrow L_V$ is a function that assigns a node label set $l_v \in L_V$ to each vertex $v \in V$,
- $\nu : E \rightarrow L_E$ is a function that assigns an edge label set $l_e \in L_E$ to each edge in E,
- L_V and L_E are node label set and edge label set respectively.

When the nodes or edges of a graph have labels, then the graph is called a labeled graph. If $L_V = L_E = \emptyset$ then G is called the unlabeled graph. Figure 2.1 shows examples of labeled and unlabeled graphs. $|G|$ denotes the number of nodes in a graph G.

Definition 2.2.2 (Subgraph)**.** Let $G_1 = (V_1, E_1, \mu_1, \nu_1)$ and $G_2 = (V_2, E_2, \mu_2, \nu_2)$ be two graphs. Graph G_1 is said to be a subgraph of graph G_2, if

- $V_1 \subseteq V_2$,
- $E_1 \subseteq E_2$,
- For every node $u \in G_1$, we have $\mu_1(u) = \mu_2(u)$,
- similarly, for every edge $e \in G_1$, we have $\nu_1(e) = \nu_2(e)$.

Definition 2.2.3 (Graph Isomorphism)**.** Let $G_1 = (V_1, E_1, \mu_1, \nu_1)$ and $G_2 = (V_2, E_2, \mu_2, \nu_2)$ be two graphs. A graph isomorphism between G_1 and G_2 is a function $f : V_1 \rightarrow V_2$ such that

- $\mu_1(u) = \mu_2(f(u)), \forall u \in V_1$,

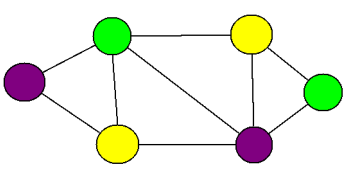

FIGURE 2.1-A: Undirected labeled

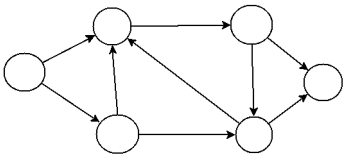

FIGURE 2.1-B: Directed unlabeled

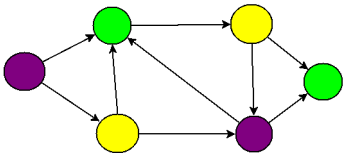

FIGURE 2.1-C: Directed labeled

FIGURE 2.1: Graph Examples

- $\forall e_1 = (u,v) \in E_1, \exists e_2 = (f(u), f(v)) \in E_2$ such that $\nu_1(e_1) = \nu_2(e_2)$,
- $\forall e_2 = (u,v) \in E_2, \exists e_1 = (f^{-1}(u), f^{-1}(v)) \in E_1$ such that $\nu_1(e_1) = \nu_2(e_2)$,

In other words, graph isomorphism between two graphs G_1 and G_2 is a bijection between every vertex $u \in G_1$ to a unique vertex $v \in G_2$, such that their labels and connecting edges are preserved.

Definition 2.2.4 (Subgraph Isomorphism)**.** Let G_1 and G_2 be two graphs. A mapping $f : V_1 \to V_2$ from G_1 to G_2 is called as subgraph isomorphism if there is a graph isomorphism between G_1 and a subgraph of G_2.

Definition 2.2.5 (Maximum Common Subgraph)**.** Let G_1 and G_2 be two graphs. A graph G is said to be a common subgraph of G_1 and G_2, when G is subgraph isomorphic to both G_1 and G_2. G is called as Maximum Common Subgraph (mcs) of G_1 and G_2, when there does not exists any larger size graph than G which is common subgraph to both G_1 and G_2.

Definition 2.2.6 (Minimum Common Supergraph)**.** Let G_1 and G_2 be two graphs. A graph G is said to be a common supergraph of G_1 and G_2, when G_1 and G_2 are subgraph isomorphic to G. G is called as Minimum Common Supergraph (MCS) of G_1 and G_2, when there does not exists any smaller size graph than G which is common supergraph to both G_1 and G_2.

A sequence of edit operations that transform a graph G_1 into G_2 is called an edit path between G_1 and G_2. A common set of edit operations include insertions, deletions, and substitutions of nodes and edges. The notation $\varepsilon \to u$ is used to represent the insertion of a vertex u, $u \to \varepsilon$ represents deletion of vertex u, and $u \to v$ denotes substitution of a vertex u by vertex v. Similarly, $\varepsilon \to e$ is used to represent the insertion of an edge e, $e \to \varepsilon$ represents deletion of the edge e, and $e \to f$ denotes substitution of an edge e by edge f.

Definition 2.2.7 (Graph Edit Distance)**.** The Graph Edit Distance (GED) between two graphs G_1 and G_2 is defined by

$$GED(G_1, G_2) = min_{(e_1,\ldots,e_k)\in\varphi(G_1,G_2)} \sum_{i=1}^{k} c(e_i)$$

Here $\varphi(G_1, G_2)$ indicates the set of edit distance paths converting G_1 to G_2, and $c(e_i)$ is the cost function associated with every edit operation e_i.

The graph edit distance between two graphs is the minimum number of modifications in terms of edit operations needed to transform one graph into another one. Graph edit distance is a generalization of tree edit distance, which in turn is a generalization of string edit distance. While the edit operations in string edit distance consist of insertion, deletion and substitution of alphabets only, the set of edit operations in tree edit distance as well as graph edit distance consists of insertion, deletion and substitution of nodes and edges.

2.3 Exact Graph Matching

2.3.1 Tree Search-Based Techniques

Tree search-based methods are among the first techniques used for graph matching task. In the 1968 paper [10], authors describe A^* search algorithm to find an optimal path between two nodes of a graph having the minimum cost. In order to select the best node to expand the search path, it uses an evaluation function $f(n)$ which is the sum of two component $g(n)$ and $h(n)$. Here $g(n)$ is the actual cost of an optimal search path from source node s to n and $h(n)$ is the actual cost of an optimal search path from n to a target node t. Authors show that by using a suitable choice of $f(n)$ the A^* search algorithm always find an optimal path from a source node to a target node. Further details and variants of A^* search algorithm is provided in the book [11].

Ullmann [12] in 1976, describes an algorithm for subgraph isomorphism which can also be used to find graph isomorphism as well as monomorphism. The author proposed a refinement procedure to inferentially eliminate the successor nodes which are not consistent

with the previous matching. Ullmann substantially updates the above paper in [13] by computing subgraph isomorphism using binary constraint satisfaction. Specifically, the author proposed a new bit-vector algorithm for binary constraint satisfaction, which depends more on the search and less on domain editing. In their 1980 paper [14], authors proposed graph monomorphism algorithm using netgraph computed from the Cartesian product of the vertices of two graphs to prune the search space.

Cordella *et al.* [15] in 1998, describe the VF algorithm, which performs graph matching using state-space representation. In this technique, each state represents a partial solution to the graph matching. The transformation from one state to other denotes inclusion of a pair of matched nodes. They use a set of criteria to prune the states of partial matching, which does not lead to required graph matching. Using experiments, they show that the VF algorithm performs significantly better than Ullmann's algorithm [16]. In [17] the same authors proposed the VF2 algorithm, which is a modified version of the VF algorithm. VF2 is particularly suitable for large graphs as it reduces memory requirement to linear with respect to the number of vertices in the graph. The formulation of the graph isomorphism problem within constraint satisfaction framework is provided in [18]. The authors also proposed a new algorithm to take advantage of the problem structure to enhance the look-ahead procedure. A backtrack search algorithm for maximal common subgraph problem is described in [19].

Bunke *et al.* [20] in 2002, describe algorithms for maximum common subgraph detection and their performance evaluation. In [21] author describes enumeration of all connected maximal common subgraph between two graphs by transforming the maximal common subgraph problem into the clique problem.

2.3.2 Algebraic Graph Theory Based Techniques

One of the important algorithm for exact graph matching and graph isomorphism proposed by McKay, known as nauty, is based on group theory [22]. This paper describes an efficient algorithm for computing the generators for the automorphism group of the graphs. The automorphism group can be used to find canonical labeling of the input graphs. The canonical labeling determines the set of isomorphic graphs by comparing the adjacency matrices of canonical form for corresponding graphs. One drawback of the above algorithm

is that the computation of canonical labeling can be exponential in the worst case. The authors in the paper [23] have shown that the the nauty program may take more computation time than some other algorithms like VF2.

Darga *et al.* [24] in 2004, introduced a new symmetry detection tool, saucy, which is an implementation of the automorphism group using a sparse data structure. They demonstrate saucy to be more efficient than nauty for several practical graph dataset. In [25], authors improved the saucy program and described a symmetry discovery algorithm by exploiting the sparsity present in input as well as output. McKay and Piperno [26] in 2014, improved the nauty program and introduced a novel technique called Traces that is more efficient than most of the other existing graph isomorphism tools.

2.3.3 Special Type of Graphs

Although subgraph isomorphism is NP-complete, and graph isomorphism problem is neither known to be P nor NP-complete, polynomial time algorithms for the special class of graphs are known. Aho *et al.* [27] describe an algorithm for tree isomorphism in polynomial time. Hopcroft and Wong describe a linear time algorithm for finding isomorphism between two planar graphs [28]. This algorithm improved the solution of planar graph isomorphism from $O(n.logn)$ [29] to linear time. The proposed algorithm can be modified to partition a collection of planar graphs into equivalence class of isomorphic planar graphs in linear time. A polynomial time algorithm of graph isomorphism problem for bounded valence graph is given by Luks [30]. The author has performed a polynomial-time reduction from the graph isomorphism problem of bounded valence to the color automorphism problem of groups having the composition factor of bounded order. The color automorphism problem for a group consists of evaluating generators for the subgroup of the group that stabilizes the color classes, and this problem is solvable in polynomial time. Dickinson *et al.* [31] in 2004, presented a method to find graph isomorphism for a class of graphs with unique node labels in polynomial time. They have shown that subgraph isomorphism, maximum common subgraph and graph edit distance can be computed in quadratic time. The 2007 paper by Kuramochi and Karypis [32] has proposed an algorithm to compute graph isomorphism between geometric graphs in polynomial time.

2.3.4 Other Techniques

Messmer and Bunke [33] proposed a new technique to graph and subgraph isomorphism detection from a sample graph to a large dataset of model graphs. They use a preprocessing step to create a decision tree for model graphs, which can be used to detect subgraph isomorphism from the candidate input graph to model graphs in polynomial time. The drawback of this approach is that the preprocessing step take exponential computation time and decision tree construction take exponential space with respect to the number of nodes in the graphs. Same authors in [34], proposed a new technique for finding subgraph isomorphism between an input graph and a large database of pre-processed graphs. In this method, subgraphs which occur frequently are represented only once to save the execution time to find them in the input graph.

Irniger and Bunke [35] presented a decision tree structures for graph database filtering. In this paper, they represent graph utilizing feature vectors, and then they build a decision tree to index the graph database using these feature vectors. Gori *et al.* [36] in 2005, describe a general framework for exact as well as approximate graph matching using a random walk. This paper presents a polynomial time algorithm for graph isomorphism for a set of graphs that do not easily reduce to other graphs. Using experiments, the authors have shown that their proposed algorithm is remarkably more efficient than other existing techniques.

Dahm *et al.* [37] in 2012, describe an improvement for subgraph isomorphism detection by eliminating the nodes using their topological features. For large graphs, this technique removes the nodes from the second graph which are not compatible with the nodes from the first graph based on the topological features of its nodes, before testing them for subgraph isomorphism.

2.4 Error-Tolerant Graph Matching

One of the significant limitations of exact graph matching is its strict constraints imposed for matching. Because of its stringent conditions, exact matching is often not suitable for real-world applications. For example, exact matching can only identify whether two graphs are exactly similar or not; but it can not tell how much they are similar. In other words,

exact matching does not explore similarity space between exactly similar graphs, which are isomorphic, and dissimilar graphs, which are non-isomorphic.

In many real-world applications, input graph data may get modified due to the presence of noise during the storage and acquisition process. The graph data may also get deformed during the processing step of extracting graphs from objects. In such circumstances, exact matching cannot be used to find a matching or similarity between two graphs. Due to the above reasons, error-tolerant graph matching is used to perform general matching between two graphs apart from checking graph and subgraph isomorphism. In this sense, error-tolerant matching is a generalization of exact matching, which can tell how much two graphs are similar when they are not exactly similar.

Error-tolerant matching is also known as an inexact matching because it finds an approximate matching between two graphs in a reasonable time when finding the optimal solution is often computationally expansive and intractable. Therefore error-tolerant matching is useful even when the possibility of distortion in input data is not present, as it returns an approximate but efficient matching when finding the best solution is not feasible.

Error-tolerant matching is also known as error-correcting graph matching because it can be used to correct errors in the form of deformation, that may have been occurred due to the presence of noise or distortions during the processing of graph. Many error-tolerant graph matching techniques are based on different kind of errors with their associated cost, and the task is to find a matching with minimum error. For example, a class of techniques for error-tolerant matching is based on graph edit operations. The standard edit operations on a graph can be insertion, deletion and substitution of nodes and edges, with each edit operation has its associated cost. Then the graph edit cost can be defined as the minimum cost of a set of edit operations that transform one graph to the other one.

In the remainder of this section, we survey the important techniques of error-tolerant graph matching proposed in the literature.

2.4.1 Tree Search-Based Techniques

Tsai and Fu [38] in 1979 proposed an error-correcting isomorphism of attributed relational graph for matching deformed patterns using an ordered search tree based algorithm. In this paper, the authors use a combination of structural technique by representing patterns as a relational graph, and statistical technique by using deformation probabilities for error-tolerant matching. Same authors in their 1983 paper [39], extended their work to error-correcting subgraph isomorphism by formulating it as a state-space tree search problem. They also described an ordered search algorithm for computing optimal error-tolerant subgraph isomorphism.

The 1980 paper [14] by Ghahraman *et al.* describes an algorithm for optimal graph monomorphism problem by specifying graph morphisms using subgraphs of Cartesian product graph, and utilizing a branch and bound algorithm. Shapiro and Haralick [40] in 1981, proposed relational homomorphism between two structural description using efficient tree-search techniques. The 1983 paper by Sanfeliu and Fu [41] proposed a distance measure for a non-hierarchical attributed relational graph using the recognition of nodes. It finds the minimum number of modification needed to transform one graph into another. To determine the cost of recognition of nodes, the main features of vertices are expressed by a different cost function, which are used to find the similarity between the nodes.

Eshera and Fu [42] in 1984, proposed a new method for computing global distance measure between attributed graphs for image analysis. It works by decomposition of attributed relational graphs into a basic attributed relational graph, which is a graph in the form of one level tree. Then, state space representation is generated from these basic attributed relational graphs to find an optimal matching between the sets of basic attributed relational graphs.

The 1996 paper [43] by Cordella *et al.* describes an efficient algorithm for the error-tolerant matching of ARG graphs using a defined set of syntactic and semantic transformations. This paper defines transformation contextually with reference to a prototype, which is applicable when the sample graph matches the prototype. The same authors [44] in 1998, extend their work to perform matching using state space representation to exhibit a partial matching between two graphs. Serratosa *et al.* [45] in 1999, proposed an efficient algorithm

to find a suboptimal similarity measure between Function Described Graph (FDG) and ARG using tree search, where FDG itself is depicted as the ensemble of ARG's.

Berretti *et al.* [46] in 2000, describe an efficient solution for indexing and matching of ARG's using heuristic based on A^* search. The 2001 paper [47] by the same authors presents an efficient solution for subgraph error-correcting isomorphism problem using a look-ahead technique for computing object distances. Gregory and Kittler [48] in 2002, propose color image retrieval using graph search techniques utilizing matching through graph edit operations and optimal search methods.

2.4.2 Relaxation Labeling

A novel class of error-tolerant graph matching techniques is based on relaxation labeling. The idea is to assign each node of the first graph a label from a set of labels so that it matches a node of the other graph. For each node of the one graph, there is a vector of probabilities based on a normal probability distribution for assigning it to the nodes of the other graph. During the initial step, these probabilities are computed using node and edge attributes and other information about graphs. It is then updated in successive iterative steps until the desired labeling is achieved. Fischler and Elschlager [49] in 1973, proposed relaxation labeling by providing a combined descriptive scheme and decision metric. Authors also described an algorithm using an approach similar to dynamic programming to reduce a huge amount of computational time. Rosenfeld *et al.* [50] in 1976 described scene labeling by extending discrete relaxation to probabilistic relaxation and performing ambiguity reduction among the objects in a scene using iterated relaxation operations.

Relaxation labeling technique has been further improved and extended in several works. The 1989 paper [51] by Kittler and Hancock try to resolve the issue of combining evidence in probabilistic relaxation in a systematic way by providing a specification that does not need the use of a support function. Above authors [52] in 1990, describe discrete relaxation to perform the maximum a posteriori probability estimation of globally consistent label assignment. Christmas *et al.* [53] in 1995, proposed probabilistic relaxation for graph matching using features extracted from 2D images and expressed the matching problem in the Bayesian framework for label assignment. Wilson and Hancock [54] in 1997, proposed an error-tolerant matching using Bayesian consistency measure by extending the

probabilistic framework. The 1999 paper [55] by Huet and Hancock extends the above work by describing a Bayesian matching for large image libraries using edge consistency as well as node attribute similarity. Myers *et al.* [56] in 2000, proposed Bayesian graph edit distance for performing error-tolerant matching of corrupted graphs. Torsello and Hancock [57] in 2003, used relaxation labeling to perform approximate tree edit distance having uniform edit cost.

2.4.3 Spectral Methods

A class of graph matching techniques is based on the spectral method of algebraic graph theory. The spectral method is based on the fact that adjacency matrices of graphs remain unchanged during node rearrangement and therefore for similar graphs, adjacency matrices will have the same eigendecomposition. Computing eigenvalue and eigenvector are well-known problems, which can be solved efficiently in polynomial time. Therefore many graph matching techniques use the spectral techniques by evaluating eigenvalues and eigenvector of matrices.

Umeyama [58] in 1988, proposed the matching between two weighted graphs, having weight for each edge, using eigendecomposition of the adjacency matrices. The author shows that near optimal matching can be achieved If the graphs are close to each other. One of the restriction of the proposed method is that the input graphs should be of the same size. The paper [59] by Carcassoni and Hancock in 2001, proposed an eigendecomposition based technique to weighted graph matching using a hierarchical approach. They use a probabilistic approach to restrict the individual matching using the matching between the pairwise clusters. In 2004, Caelli and Kosinov [60] presented graph eigen-space based method for error-tolerant graph matching using eigen-subspace projection and vertex clustering techniques for graphs having an equal and different number of nodes. Robles-Kelly and Hancock [61] in 2005, proposed graph spectral seriation technique to transform the adjacency matrix of a graph into a string sequence and then apply efficient string matching methods to it. In 2005, Shokoufandeh *et al.* [62] described a framework for indexing hierarchical image structures using the spectral characterization of a directed acyclic graph. Cour *et al.* [63] in 2007, proposed balanced graph matching using spectral relaxation technique for an approximate solution to graph matching problem. The authors also present a normalization technique to enhance the

accuracy of the existing graph matching methods. Duchenne *et al.* [64] in 2011, proposed a tensor-based algorithm for hypergraph matching by maximizing the multilinear objective function through the generalization of spectral techniques.

2.4.4 Artificial Neural Networks

A category of graph matching methods using artificial neural network has also been proposed. Suganthan *et al.* [65] in 1995, applied the Hopfield neural network to homomorphic graph matching by optimizing an input energy criterion. The 1997 paper [66] by Sperduti and Starita proposed supervised neural network for graph classification using the concept of generalized recursive structure. In 1998, Frasconi *et al.* [67] presented a general framework for processing of structural information by unifying artificial neural network and belief network models, where the connection between data variables are represented by directed acyclic graphs. Barsi [68] in 2003, described the generalization of self-organizing feature map by replacing the regular neuron grid by the undirected graph. Jain and Wysotzki [69] in 2005, proposed a neural network based technique to error-tolerant graph matching problem by reducing the search space using neural refinement procedure and subsequently performing energy minimization process. The 2009 paper [70], by Scarselli *et al.*, presented a graph neural network model by extending the existing neural network models to support the data represented in the structural domain.

2.4.5 Graph Edit Distance

Graph edit distance is one of the most flexible and widely used technique for error-tolerant graph matching. Fu and Bhargava [71] in 1973, proposed a method to represent a pattern by a tree instead of strings. Authors describe a tree system consisting of grammar, transformation and mapping on tree and tree automata; and apply it to the structural pattern recognition. The 1983 paper [41] by Sanfeliu and Fu presented a distance measure for finding the minimum number of alterations needed to convert one graph to another one using insertion, deletion, and substitution of nodes and edges. Bunke and Allerman [72] in 1983 described an error-tolerant graph matching of attributed graphs for structural pattern

recognition using state space search with heuristic information. Authors define the cost function of edit operations so that the graph distance satisfies the properties of metrics under particular conditions.

Graph edit distance applies to a wide range of applications, as specific edit cost functions can be defined for different applications. One of the limitations of graph edit distance is that it is computationally expensive, and its worst-case complexity is exponential with the number of nodes in input graphs as Zeng *et al.* [73] demonstrated graph edit distance to be an NP-hard problem. To overcome this issue, a large number of approximate and suboptimal algorithms for error-tolerant graph matching has been proposed. In 2004, Neuhaus and Bunke [74] presented an approximate and error-tolerant graph matching for planar graphs by successively matching subgraphs to locally optimize structural similarity until it obtains a complete edit path. To find the candidate path the authors use the concept of the neighborhood of a node, to match the subgraph constructed from the neighborhood of the nodes of the first graph to subgraphs created from the neighborhood of the various nodes of the other graph. Neuhaus *et al.* [75] in 2006, presented techniques for suboptimal computation of graph edit distance using the fact that A^* search-based graph edit distance traverse a large area of search space which may not be pertinent to particular classification problems. Authors proposed two variants of A^*, namely A^*-beamsearch and A^*-pathlength. A^*-beamsearch process only a fixed number of nodes called beam-width at any given level of the search tree rather than exploring all the successor nodes of the search tree. A^*-pathlength uses an extra weight factor to select the longer partial edit path over the short ones. Riesen and Bunke [76] in 2009, proposed suboptimal graph edit distance computation using bipartite graph matching, which considers only local instead of global edge structure in the course of the optimization process. Their method uses bipartite optimization process to map vertices and its local structure of the first graph to vertices and its local structure of the second graph. Sole-Ribalta *et al.* [77] in 2012, described the various properties and applications of the graph edit distance cost. Ferrer *et al.* [78] in 2015, further enhance the distance accuracy of the above bipartite graph matching framework by exploiting the order of the assignment computed by the approximation algorithm. In 2015, Riesen *et al.* [79] described the estimation of exact graph edit distance using the regression analysis on the lower and upper bounds of bipartite approximation. Riesen and Bunke [80] in 2015 used various search strategies like iterative search, greedy search, and beam search, etc., to improve the approximation of bipartite graph edit distance computation. Riesen [8] in

2015, describes the various algorithm for structural pattern recognition using graph edit distance. Dwivedi and Singh [81] in 2017, proposed homeomorphic graph edit distance, which performs path contraction on the two inputs graphs before computing graph edit distance. Same authors in 2018, extend the above approach to describe error-tolerant graph matching using node contraction [82].

2.4.6 Special Type of Graphs

Error-tolerant graph matching algorithms for special kind of graphs have also been proposed. Shasha *et al.* [83] in 1994, described efficient algorithms and heuristics, leading to approximate solutions of tree edit distance. In 1997 author [84] describes an error-tolerant tree matching algorithm using branch and bound technique. Rocha and Pavlidis in [85] presented efficient error-correcting homomorphism for planar graphs. Valiente [86] in 2001, described a tree distance which can be computed efficiently in polynomial time. Dwivedi and Singh [87] in 2019, presented an efficient error-tolerant graph matching algorithm for geometric graphs. Other inexact graph matching for a particular class of graphs is described in [88], [89].

2.4.7 Graph Kernels and Embeddings

Kernel methods enable us to apply statistical pattern recognition to graph domain. Different type of graph kernels used in graph matching is described by Neuhaus and Bunke [7] and Gartner [90]. Neuhaus *et al.* [91] described various kernels for error-tolerant graph classification. Kashima *et al.* [92] in 2003, proposed marginalized kernels between labeled graphs based on an infinite dimensional feature space. In [93], the authors present graph kernels based on the shortest path, which is positive definite and computed in polynomial time. Lafferty and Lebanon [94] in 2005, described a class of kernel known as diffusion kernel for statistical learning which utilizes the geometric structures of statistical models. In [95], the author presents convolution kernels for discrete structures which generalizes the class of radial basis kernels. Gazere *et al.* [96] in 2012, describe two new graph kernels for regression and classification problem. The first kernel, which is Laplacian kernel, is based on the notion of edit distance, whereas the second kernel, known as treelet kernel is

based on subtrees enumeration. Strug [97] in 2011, described a kernel for the hierarchical graph using a combination of tree and graph kernel. In the 2013 paper, Bai and Hancock [98] proposed graph kernels based on Jensen-Shannon kernel, which is non-extensive information theoretic kernel. Riesen and Bunke [99] in 2010, describe graph embedding and various classification and clustering techniques for graphs based on vector space embedding.

2.4.8 Other Methods

Now, we briefly review other techniques for error-tolerant graph matching proposed in literature. In [100], the authors describe the error-tolerant graph matching using an expectation maximization algorithm. Gold and Rangarajan [101] in 1996, proposed graduated assignment algorithm for graph matching. Liu *et al.* [102] proposed a modified genetic algorithm for solving weighted graph matching problem. Khoo and Suganthan [103] in 2001, describe a genetic algorithm based optimization for multiple relational graph mapping. Wang *et al.* [104] present an approximate graph matching algorithm for labeled graphs. Some other techniques are described in [105, 106, 107, 108].

2.5 Summary

This chapter presented basic concepts and definition used in graph matching. It also described a survey of various graph matching techniques and algorithms. We discussed methods for both exact as well as error-tolerant graph matching techniques. A brief survey of graph edit distance is also presented. Different type of graph matching techniques may be suitable for different applications depending on their requirement. For example, exact graph matching is suited for applications which require rigorous matching such as graph isomorphism. On the other hand, for the applications which are affected by the presence of noise or distortion inexact matching may be perfect. We also described exact and inexact graph matching techniques for special types of graphs.

In the survey on graph matching, one issue we found is that the graph matching techniques are computationally much expansive. We have addressed this issue in chapters 3, 4 by

proposing several approximate solutions. In chapter 3, we present an approach to graph matching using node contraction, which reduces the search space based on the degree of nodes. Chapter 4, describes graph matching derived from a given fraction of nodes of both graphs using some centrality measure. It can be used to perform matching under a time constraint.

Another issue we found in graph matching is the lack of an adequate similarity measure. Towards this direction, we proposed a similarity measure in chapter 5, to find the similarity between two geometric graphs. This framework requires that every node of the graph dataset should have its associated coordinate point. Given any graph, it can be embedded in a two-dimensional plane to get the coordinates of its vertices. We applied this similarity framework to perform exact and error-tolerant graph matching.