Recent Developments in Graph Matching

Horst Bunke
Department of Computer Science
University of Bern
Neubrückstr. 10, CH-3012 Bern, Switzerland
bunke@iam.unibe.ch

Abstract

Graphs are a powerful and versatile tool useful in various subfields of science and engineering. In many applications, for example, in pattern recognition and computer vision, it is required to measure the similarity of objects. When graphs are used for the representation of structured objects, then the problem of measuring object similarity turns into the problem of computing the similarity of graphs, which is also known as graph matching. In this paper, similarity measures on graphs and related algorithms will be reviewed. Also theoretical work showing various relations between different similarity measures will be discussed. Other topics to be addressed include graph clustering and efficient indexing of large databases of graphs.

1 Introduction

Graphs are a powerful data structure useful for object representation in structural pattern recognition. Typically, the parts of a complex object are represented by nodes, and relations between the parts by edges. These relations can be of various nature, for example, geometric, spatial, temporal, conceptual, a.s.o. Labels and attributes for the nodes and edges are used to incorporate further information in a graph representation. For example, labels or attributes often represent quantities such as the length of a line segment, the size or the color of a region, the angle enclosed between two straight lines, or the spatial distance between two points. In addition to a high representational power, graphs have a number of interesting invariance properties. For instance, if a graph is translated, rotated or transformed into its mirror image, it is still the same graph in the mathematical sense. Due to these properties, graphs have become a very popular representation formalism.

In many applications in pattern recognition and related areas, object similarity is a key issue. Given a database of known objects and some input to be classified, the task is to retrieve one or several objects from the database that are similar to the input. If graphs are used for object representation, this problem turns into determining the similarity of graphs, which is generally referred to as *graph matching*.

Early approaches to graph matching were restricted to finding graph or subgraph isomorphisms between two graphs [55]. A graph isomorphism is a bijective mapping between the nodes of two graphs that have the same number of nodes, identical labels, and an identical edge structure. Similarly, a subgraph isomorphism between two graphs g_1 and g_2 is an isomorphism between g_1 and a subgraph of g_2. Graph and subgraph isomorphism are useful to find out if two objects are the same or if one object is present in a group of several objects, up to the invariance properties inherent to the underlying graph representation. However, both graph and subgraph isomorphism are limited in their applicability because real world objects are usually affected by noise such that the graph representations of identical objects may not exactly match. Therefore, it is necessary to integrate some degree of error tolerance into the matching process.

One way to cope with errors and distortions is to use the maximum common subgraph of two graphs as a similarity measure [24, 29]. The maximum common subgraph g of two graphs g_1 and g_2 is a subgraph of both g_1 and g_2 and has, among all those subgraphs, the maximum number of nodes. Clearly, the more similar g_1 and g_2 are, the larger is their maximum common subgraph. A powerful alternative to maximum common subgraph is error tolerant matching using graph edit distance [6, 54]. Graph edit distance is an extension of the well-known concept of string edit distance [57] to the domain of graphs. In graph edit distance computation one introduces a set of graph edit operations. The purpose of these edit operations is to correct the errors that may have corrupted the graphs under consideration. In its most general form, a graph *edit operation* is either a deletion, insertion, or substitution (i.e. label change). Edit operations can be applied to nodes as well as to edges. The edit distance of two graphs, g and g', is defined as the shortest

Proceedings of the International Conference on Pattern Recognition (ICPR'00)
1051-4651/00 $10.00 @ 2000 IEEE

sequence of edit operations that transform g into g'. Obviously, the shorter this sequence is the more similar are the two graphs. Thus edit distance is suitable to measure the similarity of graphs. The shortest sequence of edit operations that transform a graph g_1 into another graph g_2 is not only a measure of the similarity of g_1 and g_2, but also implies a mapping from the nodes of g_1 to the nodes of g_2. This mapping corresponds to a transformation that corrects all distortions in the graphs with minimum effort. In practical applications, some edit operations may have more importance than others. Hence, very often a *cost* is assigned to each individual edit operation. Typically the more likely an edit operation is to occur the smaller is its cost. An assignment of costs to the individual edit operations is often called a *cost function*. Given a set of edit operations together with their costs, graph edit distance computation in its most general form means to find a sequence of edit operations that transform, with minimum cost, one of the given graphs into the other.

Numerous applications of graph matching have been reported in the literature. They include case-based reasoning [2, 39], machine learning [11, 17, 31], planning [44], semantic networks [13], conceptual graph [28], monitoring of computer networks [52]. Furthermore it was used in the context of visual languages and programming by graph transformations [41, 43]. Numerous applications from the areas of pattern recognition and machine vision have been reported. They include recognition of graphical symbols [19, 23], character recognition [26, 42], shape analysis [9, 25, 38], three-dimensional object recognition [64] and video indexing [50, 51].

In this paper we review recent developments in the area of graph matching. Basic concepts and theoretical foundations are presented in Section 2. Then in Section 3 an overview of graph matching algorithms is given. Recent work in graph clustering is described in Section 4. The particular problem of efficiently accessing a large database of graphs is addressed in Section 5. Finally, a discussion and conclusions are presented in Section 6.

2 Fundamental Concepts and Theoretical Advancements

The graphs typically found in pattern recognition and computer vision are characterized by a finite number of nodes and a finite number of directed edges. Each node and each edge usually has one or more (symbolic) labels and/or (numerical) attributes associated with it. These nodes and attributes come from a finite or an infinite domain. Typically, all nodes and all edges typically have the same attributes and labels, respectively. The most important concepts in graph matching are graph isomorphism, subgraph isomorphism, maximum common subgraph detection, and error-tolerant graph matching using graph edit distance. For formal definitions see [4].

In some recent papers, relationships between graph edit distance and other, well established concepts from graph theory were studied. Although the results of these studies may not be directly applicable to practical problems, they are certainly useful to gain a better theoretical understanding of graph matching. In [3] it was shown that maximum common subgraph and graph edit distance computation are equivalent to each other under a special class of cost functions. This class of cost functions is characterized by the constraint that the cost of deletion and insertion of any item (i.e. any node or edge) is no more than the cost of substituting the same item. Consequently, there won't be any substitution operation in a minimum cost sequence of edit operations, and the only edit operations actually applied are insertions and deletions. Under this class of cost functions the maximum common subgraph g of two graphs g and g' and their edit distance are related to each other through the simple equation

$$\delta(g, g') = 1 - \frac{|mcs(g, g')|}{max(|g|, |g'|)} \qquad (1)$$

In this equation $mcs(g, g')$ denotes the maximum common subgraph of g and g' and $|g|$ stands for the number of nodes of g. This similarity measure is a metric. Thus it may be useful for applications where properties such as reflexivity or triangular inequality are desired.

An in-depth study of the influence of the underlying cost function on graph edit distance computation was presented in [5]. The main result of this study is that, for any cost function, there exist infinitely many other, equivalent cost functions that lead to the same optimal sequence of edit operations for transforming two given graphs into each other. Moreover, given the edit distance $d(g, g')$ under one particular cost function, the edit distance $d'(g, g')$ under any other cost function from the same equivalence class is just a linear function of $d(g, g')$. From the practical point of view, this result tells us that any particular graph matching algorithm designed for a special cost function can be used for infinitely many other cost functions as well, i.e., all other cost functions from the same equivalence class.

A novel concept, the minimum common supergraph of two graphs, was recently introduced [7]. A supergraph g of two graphs, g' and g'', is a graph that contains both g' and g'' as subgraphs. The minimum common supergraph of g' and g'' is a graph that is a supergraph of both g' and g'' and has, among all those supergraphs, the minimum number of nodes and edges. It has been shown that the computation of the minimum common supergraph can be solved through computation of the maximum common subgraph. Similarly to eq.(1) there is a relation between the minimum common supergraph of two graphs and their edit distance

[7]. While maximum common subgraph can be regarded a kind of intersection operator on graphs, minimum common supergraph can be interpreted as graph union. This observation may be an interesting starting point for investigating graph operators with algebraic properties.

3 Graph Matching Algorithms

A wide spectrum of algorithms with different characteristics have become available for graph matching. The standard algorithm for graph and subgraph isomorphism detection is the one by Ullman [55]. Maximum common subgraph detection has been addressed in [24, 29, 37]. Classical methods for error-tolerant graph matching can be found in [15, 45, 48, 54, 63]. Most of these algorithms are particular versions of the A* search procedure, i.e., they rely on some kind of tree search incorporating various heuristic lookahead techniques in order to prune the search space.

These methods are guaranteed to find the optimal solution but require exponential time and space due to the NP-completeness of the problem. Suboptimal, or approximative methods, on the other hand, are polynomially bounded in the number of computation steps but may fail to find the optimal solution. For example, in [10, 62] probabilistic relaxation schemes are described. Other approaches are based on neural networks such as the Hopfield network [16], the Kohonen map [65] or the Potts MFT neural net [53]. Also genetic algorithms have been proposed recently [12, 60]. In [58] an approximate method based on maximum flow is introduced and in [18] and [61] graduate assignment and Tabu search are investigated, respectively. However, all of these approximate methods may get tracked in local minima and miss the optimal solution. Approaches to the weighted graph matching problem using Eigenvalues and linear programming, have been proposed in [56] and [1], respectively. As a special case, the matching of trees has been addressed in a series of papers recently [9, 36, 38, 59].

In the remainder of this section we briefly review three optimal graph matching methods that were proposed recently. In [30, 33] a new method is described for matching a graph g against a database of model graphs $g_1, \ldots, g_n$ in order to find the model g_i with the smallest edit distance $d(g, g_i)$ to g. The basic assumption is that the models in the database are not completely dissimilar. Instead, it is supposed that there are graphs ${s_j}'s$ that occur simultaneously as subgraphs in several of the ${g_i}'s$, or multiple times in the same g_i. Under a naive procedure, we will match g sequentially with each of the ${g_i}'s$. However, because of common subgraphs s_j shared by several models g_i, the ${s_j}'s$ will be matched with g multiple times. This clearly implies some redundancy.

In the approach described in [30, 33] the model graphs $g_1, \ldots, g_n$ are preprocessed generating a symbolic data structure, called network of models. This network is a compact representation of the models in the sense that multiple occurrences of the same subgraph s_j are represented only once. Consequently, such subgraphs will be matched only once with the input. Hence the computational effort will be reduced. A further enhancement of the computational efficiency of the method is achieved by a lookahead procedure. This lookahead procedure returns an estimation of the future matching cost. It is precise and can be efficiently computed based on the network. In [30, 35] the same procedure is applied not to graph edit distance computation, but subgraph and graph isomorphism detection.

In [30, 34] an even faster algorithm for graph and subgraph isomorphism detection is described. It is based on an intensive preprocessing step in which a database of model graphs is converted into a decision tree. At run time, the input graph is classified by the decision tree and all model graphs for which there exists a subgraph isomorphism from the input are detected. If we neglect the time needed for preprocessing, the computational complexity of the new subgraph isomorphism algorithm is only quadratic in the number of input graph vertices. In particular, it is independent of the number of model graphs and the number of nodes in any of the model graphs. However, the decision tree that is constructed in the preprocessing step is of exponential size in terms of the number of vertices of the model graphs. The actual implementation described by the authors is able to cope with a single graph in the database of up to 22 nodes, or up to 30 models in the database consisting of up to 11 nodes each.

Recently the decision tree method was extended from exact graph and subgraph isomorphism detection to error-tolerant graph matching [32]. Actually, there are different approaches possible. In one approach, error correction is considered at the time of the creation of the decision tree. That is, for each model graph a set of distorted copies are created and compiled into the decision tree. The number of distorted copies depends on the maximal admissible error. At run time, the decision tree is used to classify the unknown input graph in the same way as in case of exact subgraph isomorphism detection. The time complexity of this procedure at run time is only quadratic in the number of input graph nodes. However, the size of the decision tree is exponential in the number of vertices of the model graphs and in the degree of distortion that is to be considered. Therefore, this approach is limited to (very) small graphs.

In the second approach, the error corrections are considered at run time only. That is, the decision tree for a set of model graphs does not incorporate any information about possible errors. Hence, the decision tree compilation step is identical to the original preprocessing step and, consequently, the size of the decision tree is exponential only

1051-4651/00 $10.00 @ 2000 IEEE

in the size of the model graphs. At run time, a set of distorted copies of the input graph are constructed such that all possible error corrections up to a certain error threshold are considered. Each graph in this set is then classified by the decision tree. The run time complexity of this method is $O(\vartheta n^{2(\vartheta+1)})$ where n is the number of nodes in the input graph and ϑ is a threshold that defines the maximum number of admissible edit operations.

4 Graph Clustering

Clustering is a key concept in pattern recognition. While a large number of clustering algorithms have become available in the domain of statistical pattern recognition, relatively little attention has been payed to the clustering of symbolic structures, such as strings, trees, or graphs [14, 27, 47]. In principle, however, given a suitable similarity (or dissimilarity) measure, for example, edit distance or the measure defined in eq. (1), many of the clustering algorithms originally developed in the context of statistical pattern recognition, can be applied in the symbolic domain.

In this section we review work on a particular problem in graph clustering, namely, the representation of a set of similar graphs through just a single prototype [20]. This problem typically occurs after a set of graphs has been partitioned into clusters. Rather than storing all members of a cluster, only one, or a few, representative elements are being retained.

Assume that we are given a set $G = \{g_1, \cdots, g_n\}$ of graphs and some distance function $d(g_1, g_2)$ to measure the dissimilarity between graphs g_1 and g_2. A straightforward approach to capture the essential information in set G is to find a graph $\bar{g}$ that minimizes the average distance to all graphs in G, i.e.,

$$\bar{g} = arg\ min_g \frac{1}{n} \sum_{i=1}^{n} d(g, g_i) \tag{2}$$

Let's call graph $\bar{g}$ the *generalized median* of G. If we constrain g to be a member of the given set G, then the resultant graph

$$\hat{g} = arg\ min_{g \in G} \frac{1}{n} \sum_{i=1}^{n} d(g, g_i) \tag{3}$$

is called the *set median* of G.

Given set G, the computation of the set median is a straightforward task. It requires just O(n^2) distance computations. (Notice, however, that each of these distance computations will have a high computational complexity, in general.) But the set median is restricted in the sense that it can't really generalize from the given patterns represented by set G. Therefore, generalized median is the more powerful and interesting concept. However, the actual computational procedure for finding a generalized median of a given set of graphs is no longer obvious.

It was theoretically shown that for the particular cost function mentioned in Section 2 and the case where G consists of only two elements, any maximum common subgraph of the two graphs under consideration is a generalized median [8]. Similarly, any minimum common supergraph is a generalized median as well [7]. Further theoretical properties of the generalized median have been derived in [21]. These properties are useful to restrict the search space for generalized median graph computation, which was shown to be exponential in the number of graphs in set G and their size.

A practical procedure for generalized median graph computation using a genetic search algorithm was proposed in [20]. An interesting feature of this algorithm is the chromosome representation. This representation encodes both, a generalized median graph candidate, and the optimal mapping of the nodes of this candidate to the nodes of the given graphs. Hence, the computationally expensive step of computing the optimal mapping for each candidate arising during the genetic search is avoided. Nevertheless, because of the high computational complexity inherent to the problem, the applicability of this procedure is still limited to rather small sets of graphs consisting of a few nodes each.

Concrete application examples of generalized median graph computation involving graphical elements and handprinted isolated characters have been given in [19]. In Fig. 1 another application example is shown. Each of the images in this figure consists of a number of geometric primitives (circle, triangle, square). Certain relations between these geometric primitives are considered (right, below). Using the geometric primitives and the spatial relations, it is straightforward to derive a graph representation of each image. For example, the graph corresponding to the image in the second column in row d) is given in Fig.2.

In Fig. 1, the graphs representing the images in the first two columns in each row correspond to the given graphs in set G, while the images in the third column represent the generalized median, respectively. In this example, the cost function was chosen such that the generalized median is a maximum common subgraph of the two given graphs [8]. In other words, the generalized median graph consists of exactly those geometric primitives and relations that are common to the two given images in the first two columns of each row. For example, in row a) the two given images have nothing in common and the corresponding generalized median is empty. In row b) the geometric primitives circle and square together with their spatial relation right is common to both given images. In the third column in row c), the symbol X denotes a wild card label that can be substituted with low cost by any other node label. In this example, it is

1051-4651/00 $10.00 @ 2000 IEEE

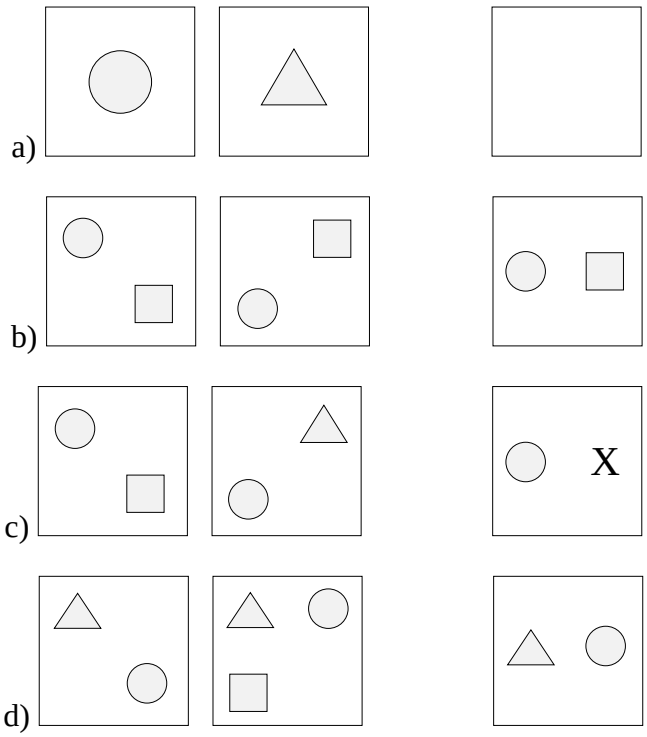

Figure 1. An application of generalized median graph (see text)

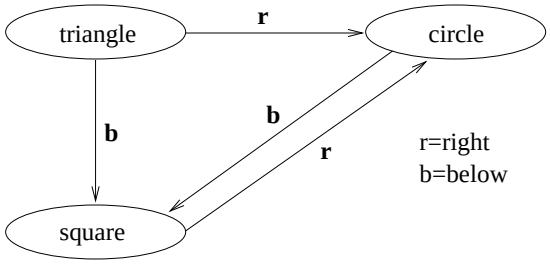

Figure 2. Graph representation of one of the images in Fig. 1

cheaper to maintain a node with the wild card label than to delete this node and the relation *right*.

Is is desirable to make generalized median graph computation applicable to larger graphs and sets of graphs. Once this goal has been reached, a number of other tools from statistical pattern recognition can be adapted to the domain of graphs. Examples include standard deviation and distance measures such as Mahalanobis distance.

5 Filtering Large Databases of Graphs

Graph matching is particularly challenging in presence of large databases. If a database holds many graphs, the sequential comparison of an input graph with each graph in the database becomes infeasible. Consequently, various indexing and preprocessing mechanisms have been proposed to reduce the computational effort [46, 49]. Recently, a new approach to the retrieval of graphs from large databases using machine learning techniques was proposed [22]. For the purpose of simplicity, only the problem of graph isomorphism was addressed.

The main idea of the proposed approach is to use simple features, which can be efficiently extracted from a graph, to reduce the number of possible candidates in the database. Examples of such features are the number of nodes or edges in a graph, the number of nodes or edges with a certain label, the number of edges (with a certain label l) incident to a node (with a certain label l'), a.s.o. Obviously, a necessary condition for a graph in the database being isomorphic to the input graph is that these features have identical values in both graphs. Therefore, it can be expected that certain graphs in the database can be ruled out by a few fast tests using only these simple features. Consequently, the number of candidates that have to undergo an expensive test for isomorphism can be reduced.

A potential problem with this approach is that there may exist a large number of simple features. Hence the question arises which of these features are most suitable to rule out as many candidates from the database as quickly as possible. To find the best feature set, the application of machine learning techniques, in particular decision tree induction by means of the C4.5 algorithm [40] was proposed.

In the experiments reported in [22] only three types of features were used, namely the number of vertices with a given label, the number of vertices with a given number of incoming edges, and the number of vertices with a given number of outgoing edges. Obviously, these features are easy and computationally inexpensive to extract. On the other hand, as it was experimentally demonstrated, they are efficient in ruling out a large number of potential candidate graphs.

Given a database with a number of graphs, the features of each graph are extracted and passed on to C4.5, which builds a decision tree. These steps are done off-line. In the on-line phase, the task is to decide whether there is a graph, which is isomorphic to the input, in the database. To solve this task, the features of the input graph are extracted and used to traverse the decision tree. There are only two possible outcomes of this decision tree traversal procedure. The first outcome is that we don't reach a leave node. In this case it is guaranteed that there is no graph in the database isomorphic to the input. The second outcome is that we do reach a leave node. In this case all graphs associated with that leave node are possibly isomorphic to the input. Hence

1051-4651/00 $10.00 @ 2000 IEEE

each of these graphs is tested, using a conventional algorithm, for example, the one reported in [55]. Obviously, feature extraction and decision tree traversal are inexpensive operations when compared to isomorphism test. Therefore, substantial savings in computation time can be expected if the average number of graphs associated with a leaf node is small, and the depth of the decision tree is bounded.

The efficiency of the proposed algorithm was experimentally evaluated using randomly generated graphs. In one series of experiments, a database of 1000 graphs with n nodes each was generated. The parameter n was continuously increased from 5 to 50. The average number of graphs associated with a leaf node in the decision tree was only 2 in this experiment, independent of the parameter n and the number of node labels, which was varied from 5 to 100. The depth of the tree is shown in Fig. 3. As both the depth of the tree and the average number of graphs associated with a leaf node are small, the proposed approach seems very appropriate for efficient retrieval of graphs from large databases.

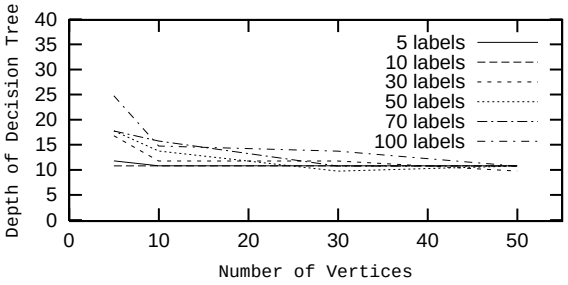

Figure 3. Depth of decision tree as a function of graph size

Future work in this area will address more general retrieval paradigms, i.e., subgraph isomorphism and error-tolerant matching, and more complex graph features.

6 Discussion and Conclusions

In this paper we have reviewed recent developments in graph matching. It can be concluded that graphs are a versatile and flexible representation formalism suitable for a wide range of problems in intelligent information processing, including the areas of pattern recognition and computer vision. A wide spectrum of graph matching algorithms have become available meanwhile. They range from deterministic approaches, suitable for finding optimal solutions to problems involving graphs with a limited number of nodes and edges, to approximate methods that are applicable to large-scale problems.

The graph matching algorithms reviewed in this paper are very general. In fact, there are no problem dependent assumptions included. The nodes and edges of a graph may represent anything, and there are no restrictions on the node and edge labels. The distortion model used in graph edit distance computation includes the deletion, insertion, and substitution of both nodes and edges. Hence it is powerful enough to model any type of error that may be introduced to a graph.

Adapting a graph matching algorithm to a particular task requires the solution of two concrete problems. First, a suitable graph representation of the objects of the problem domain has to be found. Secondly, appropriate error correction, i.e. edit, operations together with their costs have to be defined. For the solution of both problems, domain specific knowledge must be utilized whenever it is meaningful.

There are a number of open problems in graph matching that deserve further attention. An example is the extension of generalized median graph computation to larger graphs and sets of graphs. More generally, the introduction of concepts that are well established in the area of statistical pattern recognition to the structural domain is a great challenge. Particular examples include self organizing feature maps, vector quantization, and automatic parameter learning. Another topic of interest is the extension of the database indexing method described in Section 5 to error-tolerant matching.

Acknowledgement

The author wants to thank Dr. X. Jiang for continuous collaboration and intensive exchange of ideas.

References

[1] H. Almohamed. A linear programming approach for the weighted graph matching problem. *IEEE Trans. PAMI*, 15:522–525, 1993.

[2] K. Börner, E. Pippig, E. Tammer, and C. Coulon. Structural similarity and adaption. In I.Smith and B.Faltings, editors, *Advances in Case-based Reasoning, LNCS 1168*, pages 58–75. Springer, 1996.

[3] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18:689–694, 1997.

[4] H. Bunke. Error-tolerant graph matching: a formal framework and algorithms. In A. Amin, D. Dori, B. Pudil, and H. Freeman, editors, *Advances in Pattern Recognition*, pages 1–14. Springer Verlag, 1998.

[5] H. Bunke. Error correcting graph matching: On the influence of the underlying cost function. *IEEE Trans. PAMI*, 21:917–922, 1999.

[6] H. Bunke and G. Allerman. A metric on graphs for structural pattern recognition. In H.W.Schüssler, editor, *Signal Processing II: Theories and Applications*, pages 257–260. Elsevier Science Publishers B.V. (North-Holland), 1983.

[7] H. Bunke, X. Jiang, and A. Kandel. On the minimum common supergraph of two graphs. *To appear in Computing*, 2000.

[8] H. Bunke and A. Kandel. Mean and maximum common subgraph of two graphs. *Pattern Recognition Letters*, 21:163–168, 2000.

[9] V. Cantoni, L. Cinque, C. Guerra, S. Levialdi, and L. Lombardi. 2-d object recognition by multiscale tree matching. *Pattern Recognition*, 31:1443–1455, 1998.

[10] W. Christmas, J. Kittler, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *IEEE Trans. PAMI*, 8:749–764, 1995.

[11] D. Cook and L. Holder. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, pages 231–255, 1994.

[12] A. Cross, R. Wilson, and E. Hancock. Genetic search for structural matching. In B. Buxton and R. Cipolla, editors, *Computer Vision - ECCV '96, LNCS 1064*, pages 514–525. Springer Verlag, 1996.

[13] H. Ehrig. Introduction to graph grammars with applications to semantic networks. *Computers and Mathematics with Applications*, 23:557–572, 1992.

[14] R. Englert and R. Glanz. Towards the clustering of graphs. In *Proc. 2nd IAPR-TC-15 Workshop on Graph Based Representations*, pages 125–133, 2000.

[15] M. Eshera and K. Fu. A graph distance measure for image analysis. *IEEE Trans. SMC*, 14:398–408, 1984.

[16] J. Feng, M. Laumy, and M. Dhome. Inexact matching using neural networks. In E. Gelsema and L. Kanal, editors, *Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies and Hybrid Systems*, pages 177–184. North-Holland, 1994.

[17] D. Fisher. Knowledge acquisition via incremental conceptual clustering. In J.W.Shavlik and T.G.Dietterich, editors, *Readings in Machine Learning*, pages 267–283. Morgan Kaufmann, 1990.

[18] S. Gold and A. Rangarajan. A graduate assignment algorithm for graph matching. *IEEE Trans. PAMI*, 18:377–388, 1996.

[19] X. Jiang, A. Münger, and H. Bunke. Synthesis of representative symbols by computing generalized median graphs. In *Proc. Int. Workshop on Graphics Recognition GREC '99*, pages 187–194, Jaipur, 1999.

[20] X. Jiang, A. Münger, and H. Bunke. Computing the generalized median of a set of graphs. In *Proc. 2nd IAPR-TC-15 Workshop on Graph Based Representations*, pages 115–124, 2000.

[21] X. Jiang, A. Münger, and H. Bunke. On median graphs: Properties, algorithms, and applications. *Submitted*, 2000.

[22] M. Lazarescu, H. Bunke, and S. Venkatesh. Graph matching: Fast candidate elimination using machine learning techniques. *Submitted*, 2000.

[23] S. Lee, J. Kim, and F. Groen. Translation- rotation- and scale invariant recognition of hand-drawn symbols in schematic diagrams. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 4:1–15, 1990.

[24] G. Levi. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcolo*, 9:341–354, 1972.

[25] T. Lourens. *A biologically plausible model for corner-based object recognition from color images*. PhD thesis, University of Groningen, The Netherlands, 1998.

[26] S. Lu, Y. Ren, and C. Suen. Hierarchical attributed graph representation and recognition of handwritten chinese characters. *Pattern Recognition*, 24:617–632, 1991.

[27] S.-Y. Lu. A tree-to-tree distance and its application to cluster analysis. *IEEE Trans. PAMI*, 1:219–224, 1979.

[28] P. Maher. A similarity measure for conceptual graphs. *Int. Journal of Intelligent Systems*, 8:819–837, 1993.

[29] J. McGregor. Backtrack search algorithms and the maximal common subgraph problem. *Software-Practice and Experience*, 12:23–34, 1982.

[30] B. Messmer. *Efficient graph matching algorithms for preprocessed model graphs*. PhD thesis, University of Bern, Switzerland, 1995.

[31] B. Messmer and H. Bunke. Automatic learning and recognition of graphical symboles in engineering drawings. In K.Tombre and R.Kasturi, editors, *Graphics Recognition, LNCS 1072*, pages 123–134. Springer Verlag, 1996.

[32] B. Messmer and H. Bunke. Error-correcting graph isomorphism using decision trees. *Int. Journal of Pattern Recognition and Art. Intelligence*, 12:721–742, 1998.

[33] B. Messmer and H. Bunke. A new algorithm for error tolerant subgraph isomorphism. *IEEE Trans. PAMI*, 20:493–505, 1998.

[34] B. Messmer and H. Bunke. A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition*, 32:1979–1998, 1999.

[35] B. Messmer and H. Bunke. Efficient subgraph isomorphism detection - a decompostion approach. *To appear in IEEE Trans. on DKE*, 2000.

[36] K. Oflazer. Error-tolerant retrieval of trees. *IEEE Trans. PAMI*, 19:1376–1380, 1997.

[37] M. Pelillo. A unifying framework for relational structure matching. In *Proc. 14th ICPR*, pages 1316–1319, Brisbane, 1998.

[38] M. Pelillo, K. Siddiqi, and S. Zucker. Matching hierarchical structures using associated graphs. *IEEE Trans. PAMI*, 21:1105–1120, 1999.

[39] J. Poole. Similarity in legal case based reasoning as degree of matching in conceptual graphs. In M.Richter, S.Wess, K.-D.Althoff, and F.Maurer, editors, *Preproceedings: First European Workshop on Case-Based Reasoning*, pages 54–58, 1993.

[40] J. R. Quinlan. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[41] J. Rekers and A. Schürr. Defining and parsing visual languages with layered graph grammars. *Journal of Visual Languages and Computing*, 8:27–55, 1997.

[42] J. Rocha and T. Pavlidis. A shape analysis model with applications to a character recognition system. *IEEE Trans. PAMI*, pages 393–404, 1994.

[43] P. Rodgers and P. King. A graph - rewriting visual language for database programming. *Journal of Visual Languages and Computing*, 8:641–674, 1997.

[44] K. Sanders, B. Kettler, and J. Hendler. The case for graph-structured representations. In D.Leake and E.Plaza, editors, *Case-Based Reasoning Research and Development, LNCS 1266*, pages 245–254. Springer, 1997.

[45] A. Sanfeliu and K. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Trans. SMC*, 13:353–363, 1983.

[46] K. Sengupta and K. Boyer. Organizing large structural modelbases. *IEEE Trans. PAMI*, 17:321–332, 1995.

[47] D. Seong, H. Kim, and K. Park. Incremental clustering of attributed graphs. *IEEE Trans. SMC*, 23:1399–1411, 1993.

[48] L. Shapiro and R. Haralick. Structural descriptions and inexact matching. *IEEE Trans. PAMI*, 3:504–519, 1981.

[49] L. Shapiro and R. Haralick. Organization of relational models for scene analysis. *IEEE Trans. PAMI*, 3:595–602, 1982.

[50] K. Shearer. *Indexing and retrieval of video using spatial reasoning techniques*. PhD thesis, Curtin University of Technology, Perth, Australia, 1998.

[51] K. Shearer, H. Bunke, and S. Venkatesh. Video indexing and similarity retrieval by largest common subgraph detection using decision trees. *to appear in Pattern Recognition*, 2000.

[52] P. Shoubridge, M. Krarne, and D. Ray. Detection of abnormal change in dynamic networks. *Proc. of IDC'99, Adelaide*, pages 557–562, 1999.

[53] P. Suganthan, E. Tesh, and D. Mital. Pattern recognition by graph matching using Potts MFT neural networks. *Pattern Recognition*, 28:997–1009, 1995.

[54] W. Tsai and K. Fu. Error-correcting isomorphisms of attributed relational graphs for pattern recognition. *IEEE Trans. SMC*, 9:757–768, 1979.

[55] J. Ullman. An algorithm for subgraph isomorphism. *Journal of the Association for Computing Machinery*, 23(1):31–42, 1976.

[56] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Trans. PAMI*, 10:695–703, 1988.

[57] R. Wagner and M. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.

[58] I. Wang, K. Zhang, and G. Chirn. The approximate graph matching problem. In *Proc. of 12th ICPR*, pages 284–288, Jerusalem, 1994.

[59] J. Wang. An algorithm for finding the largest approximately commen substructure of two trees. *IEEE Trans. PAMI*, 20:889–895, 1998.

[60] Y.-K. Wang, K.-C. Fan, and J.-T. Horng. Genetic-based search for error-correcting graph isomorphism. *IEEE Trans. SMC*, 27(4):588–597, 1997.

[61] M. Williams, R. Wilson, and E. Hancock. Deterministic search for relational graph matching. *Pattern Recognition*, 32:1255–1271, 1999.

[62] R. Wilson and E. Hancock. Graph matching by discrete relaxation. In E. Gelsema and L. Kanal, editors, *Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies and Hybrid Systems*, pages 165–176. North-Holland, 1994.

[63] E. Wong. Three-dimensional object recognition by attributed graphs. In H.Bunke and A.Sanfeliu, editors, *Syntactic and Structural Pattern Recognition- Theory and Applications*, pages 381–414. World Scientific, 1990.

[64] E. Wong. Model matching in robot vision by subgraph isomorphism. *Pattern Recognition*, 25:287–304, 1992.

[65] L. Xu and E. Oja. Improved simulated annealing, Boltzmann machine, and attributed graph matching. In L.Almeida, editor, *LNCS 412*, pages 151–161. Springer Verlag, 1990.

1051-4651/00 $10.00 @ 2000 IEEE