Matching

Efficient Attributed Graph Matching and its Application to Image Analysis

H. Bunke and B.T. Messmer

Institut für Informatik und angewandte Mathematik, University of Bern, Neubrückstr. 10, CH-3012 Bern, Switzerland, bunke@iam.unibe.ch

Abstract. Graphs are a very powerful data structure for many tasks in image analysis. If both known models and unknown objects are represented by graphs, the detection or recognition problem becomes a problem of graph matching. In this paper, we first review different methods for graph matching. Then we introduce a new family of exact and error-tolerant graph matching algorithms that have a number of interesting properties. The algorithms are particularly efficient if there is a large number of model graphs to be matched with an unknown input graph. Moreover, they allow the incremental updating of a database of known models. This property supports the application of graph matching in a machine learning context. As an example, we show a 2-D shape recognition system based on graph matching that is able to learn new shapes.

1 Introduction

Graph structures are a powerful and universal tool with applications in various subfields of science and engineering. In pattern recognition and image analysis, graphs are often used for the representation of structured objects. For example, if the problem is to recognize instances of known objects in an image, then often models, or prototypes, of the known objects are represented by means of graphs and stored in a database. The unkown objects in the input image are extracted by means of suitable preprocessing and segmentation algorithms, and represented by graphs that are analogous to the model graphs. Thus the problem of object recognition is transformed into a graph matching problem.

Generally, the term *graph matching* refers to the process of comparing two (or more) graphs with each other. There are several classes of graph matching problems. In the *graph isomorphism* problem, we want to determine if two given graphs are isomorphic to each other. An isomorphism is a bijective mapping between the nodes of the two graphs such that the structure of the edges is preserved. Informally speaking, two graphs are isomorphic to each other if they are structurally identical. In the *subgraph isomorphism* problem, we are given two graphs g_1 and g_2, and want to find out if g_2 contains a subgraph that is isomorphic to g_1. More generally, a *bidirectional subgraph isomorphism* between g_1 and g_2 means the existence of subgraphs g_1' and g_2' of g_1 and g_2, respectively, such that g_1' and g_2' are isomorphic to each other. Finally, in *error-tolerant graph matching*, we want to establish a graph, subgraph, or bidirectional subgraph

isomorphism that may include some distortions. The admissable distortions are often problem dependent. A general distortion model may include, for example, the deletion, insertion, and substitution of both nodes and edges. These distortions are also called *edit operations*. In order to model the fact that certain distortions are more frequent than others, one can assign a cost to each individual edit operation. Error-tolerant graph matching can be used to calculate a measure of similarity, or dissimilarity, for a given pair of graphs. This measure of similarity is based on the sequence of edit operations that has the minimum cost among all possible sequences that transform one of the given graphs into the other. Thus, approximate graph matching is a generalization of string edit distance computation [25]. For a more comprehensive introduction to graph matching see [1, 21].

It is still an open question whether the graph isomorphism problem is in the complexity class P or NP [7]. In this paper, we will consider only subgraph, bidirectional subgraph and error-tolerant graph matching as these problems are more important with respect to applications in image analysis. All these problems are known to be in NP. This means that all available methods have an exponential time complexity. Consequently, graph matching algorithms that are guaranteed to yield the correct solution are applicable only if the underlying graphs are relatively small. Graph matching for large graphs becomes computationally intractable. The only choice to deal with large graphs are approximate algorithms. These have usually a computational complexity that is lower than exponential, but they are no longer guaranteed to find the correct solution for a given problem.

The best known algorithm for subgraph isomorphism detection is that of Ullman [24]. Is is based on tree search with backtracking. In order to speed up the search, a particular lookahead technique is used, which allows to detect and prune dead ends in the search tree early. Another well-known method for subgraph and bidirectional subgraph isomorphism detection is based on maximal clique detection in a compatibility graph [10]. The algorithms that have been proposed for error-tolerant matching are based on tree search, similar to Ullman's algorithm [2, 5, 19, 23]. As the search space in error-tolerant matching is even larger than in regular subgraph or bidirectional subgraph isomorphism detection, the use of good heuristics together with A^*-like search techniqus [18] becomes indispensable.

Ullman's method, the technique based on maximal clique detection, and tree search based error-tolerant graph matching are optimal algorithms in the sense that they are guaranteed to yield the correct solution to a given problem. In the area of approximate algorithms, on the other hand, methods like simulated annealing [9], neural networks [6], genetic algorithms [11], continous optimization[13] and probabilistic relaxation [12] have been proposed. The idea common to all these methods is to iteratively minimize an objective function that represents the distance of the current solution to the correct solution. The most serious problem of these approaches is that the minimization procedure may either not converge or get trapped in a local minimum.

Numerous image analysis applications of graph matching have been described in the literature. These include character classification [15], schematic diagram and 2-D shape analysis [14], 3-D object recognition [8, 26], stereo vision [10], dynamic scene analysis [3] and muscle tissue classification [20].

2 Efficient subgraph and error-tolerant subgraph matching

In this section, we introduce a new family of optimal algorithms for subgraph and error-tolerant subgraph isomorphism detection. These algorithms have been developed particularly for the case where an input graph g representing some unknown object in an image is to be matched against a database of prototype graphs $p_1, \ldots, p_M$ in order to find each p_i that is a subgraph of g, or – in the case of error-tolerant matching – to find the p_i that is most similar to g. All graphs under consideration may have directed edges and any number of symbolic labels or numeric attributes attached to their nodes and edges. Given g and $p_1, \ldots, p_M$, any of the known algorithms, for example that by Ullman or the method based on maximal clique detection, would sequentially match each p_i against g. In many applications, however, the prototypes p_i will not be completely dissimilar. Instead, there will be graphs s_j that occur as subgraphs simultaneously in several of the p_i's. These s_j's will be matched multiple times against g by any of the known algorithms. This clearly leads to some redundancy.

In the new approach to graph matching described in this paper, the prototype graphs $p_1, \ldots, p_M$ are preprocessed, generating a symbolic data structure, the so-called *network* of prototypes. This network is a compact representation of the prototypes $p_1, \ldots, p_M$ in the sense that a graph s that occurs as a subgraph multiple times within the same or different prototypes is represented only once in the network. Consequently, such a graph will be matched once and only once with the input graph g. Thus the computational effort will be reduced. For the case of error-tolerant subgraph isomorphism detection, the new algorithm can be combined with a very efficient lookahead procedure. For a detailed description of the new method, see [4, 16, 17]. In this paper, we will only briefly sketch the main ideas of the matching algorithm, give an example, and show some results.

The new algorithm follows the divide-and-conquer paradigm. That is, if we want to check if there exists a subgraph isomorphism from one graph G to another graph g, we divide G into two disjoint parts, G_1 and G_2, and check if there exist two subgraph isomorphisms, one from G_1 to g, and another from G_2 to g. If there are two such subgraph isomorphisms and, additionally, the structure of the edges between G_1 and G_2 is preserved in g, we can conclude that there is a subgraph isomorphism from G to g. This observation can be utilized by successively dividing all prototypes p_i - they correspond to G in the description above - into smaller subgraphs until we eventually reach the level of single nodes. The subgraphs resulting from such a recursive division of the prototypes can be arranged in a network, where identical subgraphs resulting from different prototypes are represented only once.

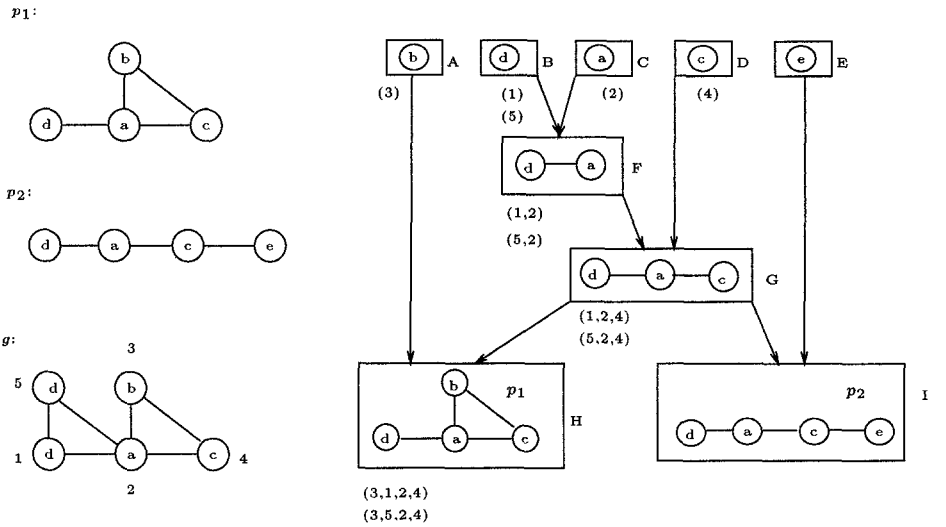

Fig. 1. Network for the graph models p_1 and p_2 and the instances found by the network units after the graph g was processed.

Two prototype graphs, p_1 and p_2, and the network that represents the recursive subdivision of p_1 and p_2 into smaller graphs are shown in Fig. 1. (Note that the network is shown "upside down" with the smallest units, i.e. network nodes representing single nodes of the p_i's, on top and full prototypes at the bottom.) A network like the one shown in Fig. 1 is not uniquely defined, in general. However, this property doesn't influence the matching performance. The generation of a network like the one given in Fig. 1 from a set of prototypes is described in more detail in [16].

A network like the one shown in Fig. 1 can be used for subgraph isomorphism detection in a straightforward manner. Each unit in the network representing some subgraph s that occurs once or multiple times in one or several prototypes has a procedure attached to it that finds all instances of s in the input graph g. Suppose that s has been split into two disjoint parts s_1 and s_2 in the recursive prototype subdivision procedure. Then in order to find all instances of s, we consider all pairs of instances of s_1 and s_2. Every such disjoint pair for which the structure of the edges between s_1 and s_2 is preserved in g is an instance of s. Therefore, in order to find all instances of the prototype graphs in the input, we start with the individual nodes at the top of the network and determine all their instances in g. Then we descend into the network and successively determine all instances of the units at the lower network levels. The procedure terminates once we have reached units at the bottom of the network, which represent the prototypes.

To illustrate the new graph matching procedure, let's consider an example.

Assume we want to determine all occurrences of p_1 and p_2 in the input graph g shown in Fig. 1. Note that the small letters denote node labels, while the number $1, 2, \ldots, 5$ in g are used to uniquely identify the nodes in g. Furthermore, capital letters $A, B, \ldots, I$ are used as identifiers for the network units. Units A through E check for instances of single nodes in the input graph, H and I check for the prototypes p_1 and p_2, respectively, and units F and G check for proper subgraphs of p_1 and p_2 consisting of more than one node. In Fig. 1 the instances that were found by the network units in the input graph g are printed in round brackets below each unit. For example, the nodes 1 and 5 are instances of network unit B in g, the pairs $(1,2)$ and $(5,2)$ are instances of the subgraph represented by network unit F, a.s.o. Finally, two instances of p_1 and no instance of p_2 are found. It can be easily verified that this is the correct result.

It is possible to generalize the algorithm to error-tolerant subgraph isomorphism detection. The static network, an example of which is shown in Fig. 1, as well as the algorithm that compiles a network from a set of prototype graphs remain the same. But the dynamic procedure that finds instances of network units in the input graph has to be extended. In the error-tolerant version, each network unit also accepts distorted subgraphs in the input graph g. Together with each distorted instance of a network unit, the corresponding edit costs are stored. The network nodes are no longer activated strictly from top to bottom, i.e. from smaller to larger units, but in a best-first manner where the subgraph with the smallest edit cost is considered first. It is possible to integrate a lookahead procedure that takes into regard a lower bound estimate of the future cost when selecting the unit with the smallest edit cost. This lookahead procedure can be implemented such that almost no additional computational overhead arises. Thus, we get a very efficient error-tolerant graph matching procedure. For more details of this method see [16, 17].

In a theoretical computational complexity analysis, the best and the worst case behavior of the new family of algorithms were studied and compared to Ullman's method for the case of subgraph isomorphism detection, and A^*-like search techniques for the error-tolerant case [18]. The main result is that for one prototype or completely disjoint prototypes – these are the most unfavorable scenarios for the network based algorithms – the worst case time complexity of both approaches is the same, while in the best case the network-based method is better by a factor of P, where P is the number of nodes in one prototype. Moreover, for the case of $M > 1$ prototypes $p_1, \ldots, p_M$ it can be shown that the complexity of the new method is only sublinearly dependent on M if there are common parts that are shared by different p_i's. In the limit with all p_i's being identical, the complexity is no longer dependent on M. By contrast, the complexity of any traditional, non-network-based matching procedure is always linear in M, no matter how similar or dissimilar the individual p_i's are.

The results of the theoretical complexity analysis could be confirmed in a series of practical experiments where network-based subgraph isomorphism detection was compared to Ullman's algorithm. All graphs in these experiments were randomly generated.

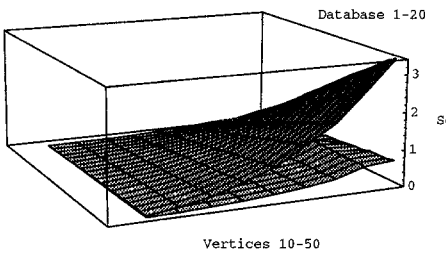

Fig. 2. Computation time for increasing the number of vertices and the number of prototypes in the database (the lower plane denotes the network algorithm, while the upper plane corresponds to the traditional algorithm).

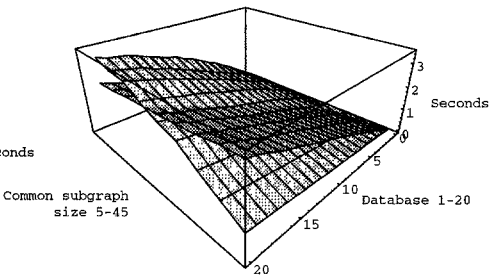

Fig. 3. Computation time for increasing the number of vertices in the common subgraph and the size of the database (the lower plane at the front corner denotes the network algorithm, while the upper plane corresponds to the traditional algorithm).

In the first experiment shown in Fig. 2 we increased the number of vertices in the prototype graphs from 10 to 50 and the number of prototypes in the database from 1 to 20. The labels of the vertices were randomly chosen from a set of 10 different labels. For each prototype graph a corresponding, isomorphic input graph was generated by reordering the adjacency matrix. The lower plane in Fig. 2 denotes the time of the new algorithm while the upper plane denotes the time of the Ullman's algorithm. Fig. 2 confirms that for a growing number of prototypes the time for the new algorithm grows only sublinearly while the traditional algorithm is linearly dependent on the size of the database. Additionally, we observe that for larger prototype graphs the new algorithm performs remarkably better than the traditional algorithm. This can be explained by the fact that for larger graphs, the number of identical subgraphs within a single prototype and among different prototypes also grows. In this first experiment, no common subgraphs of the different prototypes was explicitly defined. However, due to the limited number of labels, common subgraphs evolved naturally.

In the second experiment, the effect of sharing common substructures among prototypes was examined more closely. For this purpose, we generated prototype graphs consisting of 50 vertices and approximately 60 edges and increased the number of prototypes from 1 to 20 and the size of the common subgraph from 5 to 45 vertices. Except for the common subgraph, all prototypes were disjoint. The results of the second experiment are given in Fig 3 where the lower plane at the front corner denotes the times of the new algorithm while the upper plane represents the traditional algorithm. Clearly, the intersection of the two planes indicates that for a small or no common subgraph in the different prototypes the performace of the new algorithm is slightly worse than that of the traditional algorithm. But for an increasing size of the common subgraph, the time needed by the new algorithm decreases and becomes much less than that required by the traditional algorithm.

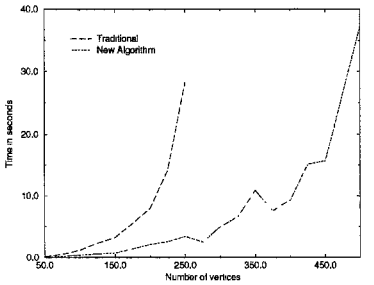

Fig. 4. Computation time for an increasing number of vertices in the prototype graphs.

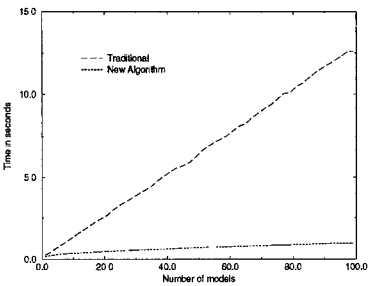

Fig. 5. Computation time for an increasing number of prototypes in the database.

Finally, in the last two experiments we repeated the first experiment for large graphs and a large number of prototypes, respectively. In Fig. 4 we kept the number of prototypes at one and steadily increased the size of the prototype from 50 to 500 vertices. In Fig. 5 the size of the prototypes was fixed at 50 vertices and the number of prototypes was increased from 1 to 100. It can be conluded that under the selected scenario network-based graph matching clearly outperforms the traditional algorithm. Furtermore, even relatively large graphs and databases are computationally tractable using the proposed approach.

3 Recognition and learning of 2-D shapes using network-based graph matching

The graph matching methods introduced in the last section have the property that any network of prototypes can be incrementally built and updated. Given a network that represents prototypes $p_1, \ldots, p_M$, $M \geq 0$, we can successively add new prototypes $p_{M+1}, \ldots, p_{M+L}$ to the network without the need to recompile the whole network from scratch. This property makes network-based graph matching very interesting in the context of machine learning. While research on machine learning in the general artificial intelligence field has produced interesting results [22], machine learning in the domain of computer vision is still in its infancy. In this section, we consider the problem of symbol recognition in engineering diagrams as an example of an application of network-based error-tolerant graph matching. Input data to our system are line drawings that may contain both known and unknown symbols. All known symbols are stored in a database. The output produced by the system is a list of instances of the known symbols found in the diagrams. Furthermore, unknown symbols are identified and added to the database. Thus the system is able to learn new symbols and continuously update its database.

Symbols and drawings are represented by attributed relational graphs. The line segments of a drawing and their spatial relationships are encoded in the

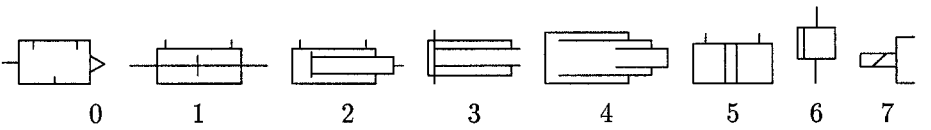

Fig. 6. DIN ISO symbols denoting machine parts.

graph such that the representation is translation, scale and rotation invariant. The process of recognizing a particular symbol in a drawing can then be formulated as the search for an error tolerant subgraph isomorphism between the symbol graph and the drawing graph. As the database of the known symbols usually contains more than one element, the recognition task for a given drawing consists of finding all symbol graphs for which a subgraph isomorphism to the drawing graph with an error less than a certain threshold exists.

The system starts with a number of a priori known symbols in the database. (This number may be zero.) Our proposed learning procedure has to satisfy two objectives. First, all symbols that are unknown, i.e. not contained in the database, should be added to the database in order to guarantee the complete interpretation of the input diagram. Secondly, if the same symbol occurs multiple times in the same input diagram (perhaps with some distortions), only one representative instance of it should be added to the database. This prevents the database from growing unnecessarily large.

In order to satisfy these two objectives, our learning scheme works in two steps. First, after all known symbols have been recognized and removed from the input diagram, all line segments that do not belong to a known symbol are collected and grouped into possible new symbols. A potential new symbol must satisfy certain constraints, such as the presence of a minimal number of line segments or at least one closed loop. These constraints are user-defined and vary from one application to the other. In the second step, a hierarchical clustering procedure is applied to the set of potential new symbols. Initially, the graph distance between each pair of potential symbols is calculated by the error tolerant graph matching method. Then clusters of symbols are formed depending on these distances. Finally, a single symbol is taken as the representative for each cluster and incrementally added to the database of known symbols.

We illustrate the process of recognizing and learning symbols with an example dealing with machine layouts. Notice that our proposed procedure can be very easily adapted to other types of engineering drawings. In Fig. 6 the set of symbols a priori known to the system is given. Based on these symbols, a first interpretation of the drawing in Fig. 7 is attempted. Instances of the symbols 0, 1 and 3 are detected and removed from the drawing; see Fig. 8. The learning scheme is then applied to the remaining symbols in Fig. 8 and the symbols given in Fig. 9 are learned and added to the database of symbols. With these new symbols it is now possible to completely interpret the drawing in Fig. 7.

It is obvious from the symbols displayed in Fig. 6 that this application is well

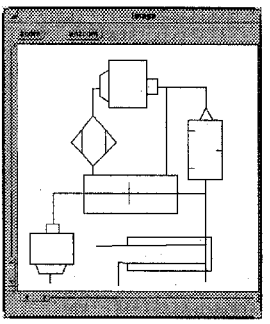

Fig. 7. First drawing, containing known and unknown symbols.

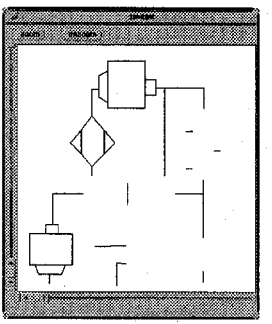

Fig. 8. First drawing after all known symbols have been removed.

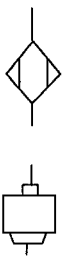

Fig. 9. Two symbols learned from the drawing in Fig. 8.

suited for the new algorithm. There is a large number of common substructures in the machine part symbols such that the detection process can be efficiently done with the network based algorithm. Furthermore, the new algorithm supports the learning scheme by allowing the network structure to be incrementally updated with new symbols.

4 Conclusions

In this paper we described a new method for exact and error-tolerant subgraph isomorphism detection, which is based on a compact network representation of model graphs. With several models in the database, the network represents identical subgraphs of the different model graphs only once, thus reducing the number of computational steps necessary in order to detect exact and error-tolerant subgraph isomorphisms from a model graph to an input graph. In addition to sharing identical graph structures, the network can be combined with a fast lookahead procedure. The efficiency of the new algorithm was analytically shown and practically demonstrated in experiments.

The graph matching algorithm described in this paper is very general and powerful. As a matter of fact, there are no problem dependent assumptions included in the algorithm. Our distortion model consists of the deletion, insertion and substitution of both nodes and edges, which is powerful enough to model any type of error that may be introduced into an input graph. Adapting the matching algorithm to a particular application requires the solution of two concrete tasks. First, a suitable graph representation of the objects in the problem domain has to be developed. Secondly, appropriate costs of the graph edit operations have to be found. It is anticipated that there are many applications where both tasks are not really difficult if a set of sample pattern is given.

One of the most challenging problems in computer vision and image analysis is the automatic learning of object models. The network of prototypes that was

proposed in this paper can be incrementally expanded by new model graphs. This property makes network-based graph matching applicable in the context of machine learning. As an example of object recognition and the automatic learning of models based on graph matching, we have described a prototype system that deals with 2-D objects in engineering drawings. In can be concluded from this example that a similar approach is possible for a number of other machine vision applications.

Acknowledgement

This work is part of project No 5003-34285 funded by the Swiss National Science Foundation under the "Schwerpunktprogramm Informatikforschung".

References

1. H. Bunke. Structural and syntactic pattern recognition. In C.H. Chen, L.F. Pau, and P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision*, pages 163–209. World Scientific Publ. Co. Singapore, 1993.
2. H. Bunke and G. Allerman. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters 1*, 4:245–253, 1983.
3. H. Bunke, T. Glauser, and T.-H. Tran. Efficient matching of dynamically changing graphs. In P. Johansen and O.Olsen, editors, *Theory and Applications of Image Analysis*, pages 110–124. World Scientific Publ., 1992.
4. H. Bunke and B.T. Messmer. Similarity measures for structured representations. In M. M. Richter, S. Wess, K.-D. Althoff, and F. Maurer, editors, *Proceedings EWCBR-93 Lecture Notes on Artifical Intelligence*. Springer Verlag, 1994.
5. M.A. Eshera and K.S. Fu. A graph distance measure for image analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 14(3):398–408, May 1984.
6. J. Feng, M. Laumy, and M. Dhome. Inexact matching using neural networks. In E.S. Gelsema and L.N. Kanal, editors, *Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies and Hybrid Systems*, pages 177–184. North-Holland, 1994.
7. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman and Company, 1979.
8. E. Gmuer and H. Bunke. 3-D object recognition based on subgraph matching in polynomial time. In R. Mohr, T. Pavlidis, and A. Sanfeliu, editors, *Structural Pattern Analysis*, pages 131–147. World Scientific, 1990.
9. L. Herault, R. Horaud, F. Veillon, and J.J. Niez. Symbolic image matching by simulated annealing. In *Proc. British Machine Vision Conference*, pages 319–324. Oxford, 1990.
10. R. Horaud and T. Skordas. Stereo correspondence through feature grouping and maximal cliques. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI*, 11(11):1168–1180, 1989.
11. K.A. De Jong and W. M. Spears. Using genetic algorithms to solve NP-complete problems. In J.D. Schaffer, editor, *Genetic Algorithms*, pages 124–132. Morgan Kaufmann, 1989.
12. J. Kittler, W. J. Christmas, and M. Petrou. Probabilistic relaxation for matching of symbolic structures. In H. Bunke, editor, *Advances in Structural and Syntactic Pattern Recognition*, pages 471–480. World Scientific, 1992.

13. P. Kuner and B. Ueberreiter. Pattern recognition by graph matching - combinatorial versus continous optimization. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(3):527–542, 1988.
14. S.W. Lee, J.H. Kim, and F.C.A. Groen. Translation- rotation- and scale invariant recognition of hand-drawn symbols in schematic diagrams. *International Journal of Pattern Recognition and Artificial Intelligence*, 4(1):1–15, 1990.
15. S.W. Lu, Y. Ren, and C.Y. Suen. Hierarchical attributed graph representation and recognition of handwritten Chinese characters. *Pattern Recognition*, 24:617–632, 1991.
16. B. T. Messmer and H. Bunke. A network based approach to exact and inexact graph matching. Technical Report IAM-93-021, Universität Bern, September 1993.
17. B.T. Messmer and H. Bunke. A new method for efficient error-correcting subgraph isomorphism. In D. Dori and A. Bruckstein, editors, *Syntactic and Structural Pattern Recognition*. World Scientific Publishers, Singapore, to appear in 1995.
18. N.J. Nilsson. *Principles of Artificial Intelligence*. Tioga, Palo Alto, 1980.
19. A. Sanfeliu and K.S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:353–363, 1983.
20. A. Sanfeliu, K.S. Fu, and J.M.S. Prewitt. An application of a graph distance measure to the classification of muscle tissue patterns. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 1(1):17–42, 1987.
21. L. Shapiro. Relational matching. In T.Y. Young, editor, *Handbook of Pattern Recognition and Image Processing: Computer Vision*, pages 475–496. Academic Press, 1993.
22. J.W. Shavlick and T.G. Dietterich, editors. *Readings in Machine Learning*. Morgan Kaufman, San Mateo, 1990.
23. W.H. Tsai and K.S Fu. Error-correcting isomorphisms of attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 9:757–768, 1979.
24. J.R. Ullman. An algorithm for subgraph isomorphism. *Journal of the Association for Computing Machinery*, 23(1):31–42, 1976.
25. R.A. Wagner and M.J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.
26. E. K. 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.