Chapter 10

Graph Matching and Similarity

Horst Bunke and Xiaoyi Jiang

Many graphical interfacing problems relay on graph matching. In this chapter, we explore and illustrate how graph matching can be performed using powerful, "intelligent" algorithms, to improve standard methods.

1. INTRODUCTION

Graphs are a powerful and universal tool widely used in information processing. Numerous methods for graph analysis have been developed. Examples include the detection of Hamiltonian cycles, shortest paths, or vertex coloring (Balakrishnau). In applications such as pattern recognition or information retrieval, object similarity is an important issue. Given a database of objects and a query, the task is to retrieve one or several objects from the database that are similar to the query. If graphs are used for object representation, this problem turns into determining the similarity of graphs, which is generally referred to as *graph matching*.

Standard concepts in graph matching include graph isomorphism, subgraph isomorphism and maximum common subgraph. Two graphs are called *isomorphic* if they have identical structure. More formally, an isomorphism between two graphs g_1 and g_2 is a bijective mapping between the nodes of g_1 and g_2 that preserves the structure of the edges. Graph representations of objects are often invariant under a number of transformations, for example, spatial transformations such as translation, rotation, and scaling. Hence, graph isomorphism is a useful concept to check whether one object is a transformed version of another. There exists a *subgraph isomorphism* between two graphs if one graph contains a subgraph that is

H.-N. Teodorescu et al. (eds.), Intelligent Systems and Interfaces, 281-304.

isomorphic to the other. Subgraph isomorphism is useful to find out if a given object is part of another object or a collection of several objects. The *maximum common subgraph* of two graphs g_1 and g_2 is the largest graph that is a subgraph of both g_1 and g_2. Maximum common subgraph is useful to measure the similarity of two objects. Clearly, the larger the maximum common subgraph of g_1 and g_2 is, the more similar are the two graphs. Algorithms for graph isomorphism, subgraph isomorphism and maximum common subgraph detection have been reported in (Ullman, 1976, Levi, 1972, McGregor, 1982).

A more general method to measure the similarity of two graphs is *graph edit distance*. It is a generalization of string edit distance (Stephen, 1994). In graph edit distance one introduces a set of graph edit operations. These edit operations are used to model distortions that transform an ideal object into a noisy version of itself. Typical graph edit operations include the deletion, insertion and substitution of nodes and edges. Given a set of edit operations, graph edit distance is defined as the minimum number of operations that transform one graph into the other. Often a cost is assigned to each edit operation. The costs are application dependent and used to model the likelihood of the corresponding distortions. Typically, the more likely a certain distortion is to occur, the lower is its cost. If a cost is assigned to each edit operation then the eidt distance of two graphs, g_1 and g_2, is defined as the minimum cost taken over all sequences of edit operations that transform g_1 into g_2 Graph edit distance and related similarity measures have been discussed in (Tsai and Fu, 1979, Shapiro and Haralick, 1981, Sanfeliu and Fu, 1983, Bunke and Allerman, 1983).

A large number of applications of graph matching have been described in the literature. One of the earliest was in the field of chemical structure analysis (Rouvray and Balaban, 1979). More recently, graph matching has been applied to case-based reasoning (Poole, 1993, Boerner, Pippig, Tammer and Coulon, 1996), machine learning (Fisher, 1990, Cook and Holder, 1994, Messmer and Bunke, 1996), planning (Sanders, Kettler and Hendler, 1997), semantic networks (Ehrig, 1992), conceptual graph (Maher, 1993), information retrieval from video databases (Shearer, Bunke, Venkatexh and Kieronska, 1998), monitoring of computer networks (Shonbridge, Kraene and Ray), and in the context of visual languages and programming by graph transformations (Rodgers and King, 1997, Rekers and Schuerr, 1997). Numerous applications from the areas of pattern recognition and machine vision have been reported. They include recognition of graphical symbols (Lee, Kim and Groen, 1990, Jiang, Muenger and Bunke, 1999), character recognition (Lu, Ren and Suen, 1991, Rocha and Pavlidis, 1994), shape analysis (Lourens, 1998), three-dimensional object recognition (Wong, 1992), and others.

In this chapter, we review recent work in the area of graph matching with an emphasis on fundamental concepts and general algorithms. Basic definition and

notations are introduced in Section 2. In Section 3, theoretical foundations are presented showing various relations between the concepts of graph edit distance and maximum common subgraph. Algorithms are discussed in Section 4. Finally, a discussion and conclusions are provided in Section 5. Parts of Sections 2 through 5 originally appeared in (Bunke, 1998).

2. BASIC DEFINITIONS AND NOTATION

In the following, we consider labeled graphs with directed edges. L_V and L_E denote the finite sets of labels for nodes and edges, respectively.

Definition 1: A *graph* is a 4-tuple $g = (V, E, \alpha, \beta)$, where

V is the finite set of vertices

$E \subseteq V \times V$ is the set of edges

$\alpha : V \rightarrow L_V$ is a function assigning labels to the vertices

$\beta : E \rightarrow L_E$ is a function assigning labels to the edges ■

Edge (u, v) originates at node u and terminates at node v. An undirected graph is obtained as a special case if there exists an edge $(u, v) \in E$ for any edge $(u, v) \in E$ with $\beta(u, v) = \beta(v, u)$. Node and edge labels come from the same alphabet, for notational convenience. If it is necessary to explicitly distinguish between node and edge labels, the set L can be split into two disjoint subsets. If $V = \emptyset$ then g is called the *empty graph*. The notation $|g|$ will be used for the number of nodes of graph g.

Definition 2: Let $g = (V, E, \alpha, \beta)$ and $g' = (V', E', \alpha', \beta')$ be graphs; g' is a *subgraph* of g, $g' \subseteq g$, if

$V' \subseteq V$

$\alpha(v) = \alpha'(v)$ for all $v \in V'$

$E' = E \cap (V' \times V')$

$\beta(e) = \beta'(e)$ for all $e \in E'$ ■

From Definition 2 it follows that, given a graph $g = (V, E, \alpha, \beta)$, any subset $V' \subseteq V$ of its vertices uniquely defines a subgraph. This subgraph is called the subgraph that is *induced* by V'.

Definition 3: Let g and g' be graphs. A *graph isomorphism* between g and g' is a bijective mapping $f : V \rightarrow V'$ such that

$$\alpha(v) = \alpha'(f(v)) \text{ for all } v \in V$$

for any edge $e = (u, v) \in E$ there exists an edge $e' = (f(u), f(v)) \in E'$ such that $\beta(e) = \beta'(e')$, and for any edge $e' = (u', v') \in E'$ there exists an edge $e = (f^{-1}(u'), f^{-1}(v')) \in E$ such that $\beta(e) = \beta'(e')$.

If $f : V \rightarrow V'$ is a graph isomorphism between graphs g and g' and g' is a subgraph of another graph g'', i.e., $g' \subseteq g''$, then f is called *subgraph isomorphism* from g to g''. ■

Definition 4: Let $g_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $g_2 = (V_2, E_2, \alpha_2, \beta_2)$ be graphs. A common *subgraph* of g_1 and g_2, $cs(g_1, g_2)$, is a graph $g = (V, E, \alpha, \beta)$ such that there exist subgraph isomorphisms from g to g_1 and from g to g_2. We call g a *maximum common subgraph* of g_1 and g_2, $mcs(g_1, g_2)$, if there exists no other common subgraph of g_1 and g_2 that has more nodes than g. ■

Definition 5: Let $g_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $g_2 = (V_2, E_2, \alpha_2, \beta_2)$ be graphs. An *error-tolerant graph matching* (*etgm*) from g_1 to g_2 is a bijective function $f : \hat{V}_1 \rightarrow \hat{V}_2$, where $\hat{V}_1 \subseteq V_1$ and $\hat{V}_2 \subseteq V_2$. ■

We say that node $u \in \hat{V}_1$ is substituted by node $v \in \hat{V}_2$ if $f(u) = v$. If $\alpha_1(u) = \alpha_2(f(u))$ then the substitution is called an *identical* substitution. Otherwise it is termed a non-*identical* substitution. Furthermore, any node from $V_1 - \hat{V}_1$ is *deleted* from g_1, and any node form $V_2 - \hat{V}_2$ *inserted* in g_2 under f. We will use $\hat{g}_1$ and $\hat{g}_2$ to denote the subgraphs of g_1 and g_2 that are induced by the sets $\hat{V}_1$ and $\hat{V}_2$, respectively.

The mapping f *directly* implies an edit operation on each node in g_1 and g_2. I.e., nodes are substituted, deleted, or inserted, as described above. Additionally, the mapping f *indirectly* implies edit operations on the edges of g_1 and g_2. If $f(u_1) = v_1$ and $f(u_2) = v_2$ and there exist edges $(u_1, u_2) \in E_1$ and $(v_1, v_2) \in E_2$ then edge (u_1, u_2) is substituted by (v_1, v_2) under f. If

$\beta_1((u_1,u_2))=\beta_2((v_1,v_2))$ then the edge substitution is called an *identical* substitution. Otherwise it is termed a *non-identical* substitution. If there exists no edge $(u_1,u_2)\in E_1$, but an edge $(v_1,v_2)\in E_2$, then edge (v_1,v_2) is inserted. Similarly, if $(u_1,u_2)\in E_1$ exists but no edge (v_1,v_2), then (u_1,u_2) is deleted under f. If a node u is deleted from g_1, then any edge incident to u is deleted, too. Similarly, if a node u' is inserted in g_2, then any edge incident to u' is inserted, too. Obviously, any *etgm* f can be understood as a set of edit operations (substitutions, deletions, and insertions of both nodes and edges) that transform a given graph g_1 into another graph g_2.

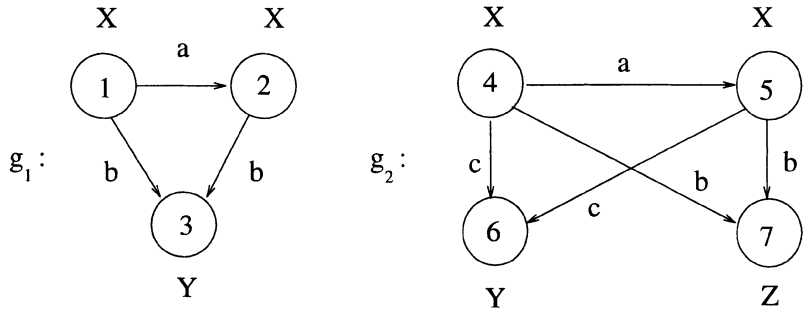

Figure 1. An example of error-tolerant graph matching (see text).

Example 1: A graphical representation of two graphs is given in Figure 1. For those graphs, we have:

$L_V=\{X,Y,Z\}; L_E=\{a,b,c\}.$
$V_1=\{1,2,3\}; V_2=\{4,5,6,7\}.$
$E_1=\{(1,2),(1,3),(2,3)\}; E_2=\{(4,5),(4,6),(4,7),(5,6),(5,7)\}.$
$\alpha_1: 1\mapsto X, 2\mapsto X, 3\mapsto Y;$
$\alpha_2: 4\mapsto X, 5\mapsto X, 6\mapsto Y, 7\mapsto Z.$
$\beta_1: (1,2)\mapsto a, \quad (1,3)\mapsto b, \quad (2,3)\mapsto b;$

Three examples of *etgm* are:

$f_1: 1\mapsto 4, 2\mapsto 5, 3\mapsto 7$ *with* $\hat{V}_1=\{1,2,3\}$ *and* $\hat{V}_2=\{4,5,7\}$
$f_2: 1\mapsto 4, 2\mapsto 5, 3\mapsto 6$ *with* $\hat{V}_1=\{1,2,3\}$ *and* $\hat{V}_2=\{4,5,6\}$
$f_3: 1\mapsto 4, 2\mapsto 5$ *with* $\hat{V}_1=\{1,2\}$ *and* $\hat{V}_2=\{4,5\}$

Under f_1, node 1, 2 and 3 are substituted by nodes 4, 5 and 7, respectively. Consequently, edges (1, 2), (1, 3) and (2, 3) are substituted by (4, 5), (4, 7) and (5, 7), respectively. The substitution of nodes 1 and 2 by 4 and 5 are identical substitutions that involve no label change; there are no label changes involved in the edge substitutions, either. The label *Y* of node 3 is substituted by *Z* of node 7, and node 6 together with its incident edges (4, 6) and (5, 6) are inserted in g_2. There are, of course, many other *etgm*'s from g_1 to g_2. ■

Definition 6: The *cost* of an *etgm* $f : \hat{V}_1 \rightarrow \hat{V}_2$ from a graph $g_1 = (V_1, E_1, \alpha_1, \beta_1)$ to a graph $g_2 = (V_2, E_2, \alpha_2, \beta_2)$ is given by

$$c(f) = \sum_{u \in \hat{V}_1} c_{ns}(u) + \sum_{u \in V_1 - \hat{V}_1} c_{nd}(u) + \sum_{u \in V_2 - \hat{V}_2} c_{ni}(u) + \sum_{e \in E_s} c_{es}(e) + \sum_{e \in E_d} c_{ed}(e) + \sum_{e \in E_i} c_{ei}(e)$$

where

$c_{ns}(u)$ is the cost of substituting node $u \in \hat{V}_1$ *by* $f(u) \in \hat{V}_2$,
$c_{nd}(u)$ is the cost of deleting node $u \in V_1 - \hat{V}_1$ *from* g_1,
$c_{ni}(u)$ is the cost of inserting node $u \in V_2 - \hat{V}_2$ *in* g_2,
$c_{es}(e)$ is the cost of substituting edge *e*,
$c_{ed}(e)$ is the cost of deleting edge *e*,
$c_{ei}(e)$ is the cost of inserting edge *e*,

and E_s, E_d and E_i are the sets of edges that are substituted, deleted, and inserted, respectively. All costs are non-negative real numbers. ■

Notice that the sets E_s, E_d and E_i are implied by the mapping *f*. That is, if edge $e = (u_1, u_2) \in E_1$, $f(u_1) = v_1$, $f(u_2) = v_2$ and $(v_1, v_2) \notin E_2$, then $e \in E_d$. Similarly, if $(u_1, u_2) \notin E_1$, $f(u_1) = v_1$, $f(u_2) = v_2$ and $e = (v_1, v_2) \in E_2$, then $e \in E_i$. Likewise, if $e_1 = (u_1, u_2) \in E_1$, $f(u_1) = v_1$, $f(u_2) = v_2$ and $e_2 = (v_1, v_2) \in E_2$, then $e_1 \in E_s$. Because an edge is deleted (inserted) whenever one or both of its incident nodes are deleted (inserted), we furthermore observe that (1) $e \in E_d$ if $e \in (V_1 \times V_1) - (\hat{V}_1 \times \hat{V}_1)$, and (2) $e \in E_i$ if $e \in (V_2 \times V_2) - (\hat{V}_2 \times \hat{V}_2)$.

A particular set of costs $c_{ns}, c_{nd}, \ldots, c_{ei}$ according to Def. 6 will be called a *cost function*.

Definition 7: Let f be an *etgm* from a graph g_1 to a graph g_2 under a particular cost function. We call f an *optimal ecgm* if there exists no other *ecgm* f' from g_1 to g_2 with $c(f') < c(f)$. ■

The cost of an optimal *etgm* from a graph g_1 to a graph g_2 is also called the *edit distance* of g_1 and g_2, and denoted by $d(g_1, g_2)$. For a given cost function there are usually several optimal *etgm*'s from g_1 to g_2.

In practice, the costs $c_{ns}, \ldots, c_{ei}$ introduced in Def. 6 are used to model the likelihood of errors or distortions that may corrupt ideal graphs of the underlying problem domain. The more likely a certain distortion is to occur, the smaller is its cost. Concrete values for $c_{ns}, c_{nd}, \ldots, c_{ei}$ have to be chosen depending on the particular application.

In the following, it is assumed, for the purpose of simplicity, that the costs $c_{nd}(x), c_{ni}(x)$ and $c_{ns}(x)$, do not depend on node x; neither do $c_{ed}(e), c_{ei}(e)$ and $c_{es}(e)$ depend on edge e. In other words, $c_{nd}(x), c_{ni}(x)$ and $c_{ns}(x)$ will be the same for any node x, and $c_{ed}(e), c_{ei}(e)$ and $c_{es}(e)$ will be the same for any edge e. Hence, the notation $c_{nd}(x) = c_{nd}, c_{ni}(x) = c_{ni}, \ldots, c_{es}(e) = c_{es}$ will be used and a cost function is given by the 6-tuple $C = (c_{nd}, c_{ni}, c_{ns}, c_{ed}, c_{ei}, c_{es})$. Unless otherwise stated, it is assumed that the cost of an identical node or edge substitution is zero, while the cost of any other edit operation is greater than zero.

Example 2: Consider cost function

$$C = (c_{nd}, c_{ni}, c_{ns}, c_{ed}, c_{ei}, c_{es}) = (1,1,1,1,1,1).$$

Then, the *etgm* f_1 given in Example 1 has cost $c(f_1) = 4$ (one node label substitution, one node insertion, and two edge insertions). It can be easily verified that there is no other *etgm* from g_1 to g_2 that has a smaller cost under C. For example, $c(f_2) = 5$ (two edge label substitutions, one node insertion, and two edge insertions) and $c(f_3) = 9$ (one node and two edge deletions, two node and four edge insertions). However, if we change the cost function and consider $C' = (1,1,3,1,1,1)$ then $c(f_1) = 6$, $c(f_2) = 5$, and $c(f_3) = 9$. Thus f_1 is no longer optimal under C'. It can be easily verified that f_2 has in fact the smallest cost among all possible *etgm*'s from g_1 to g_2. If we consider a third cost function $C'' = (1,1,7,1,1,7)$ then $c(f_1) = 10$, $c(f_2) = 17$, and $c(f_3) = 9$. Under this cost function, f_3 is optimal. ■

3. THEORETICAL FOUNDATIONS OF GRAPH MATCHING

As it was pointed out in the last section, the underlying cost function has an important influence on optimal *etgm*. For practical applications, unfortunately, there is no automatic procedure known today to derive a cost function, suitable for a given task, from a set of samples. Typically, cost functions are defined in an *ad hoc* manner, purely guided by heuristics and intuition. In order to avoid the problem of finding a suitable cost function, a new graph distance measure based on the maximum common subgraph of two graphs was proposed in (Bunke and Shearer, 1998). Given two non-empty graphs, g_1 and g_2, their distance is defined as

$$\delta(g_1, g_2) = 1 - \frac{|mcs(g_1, g_2)|}{\max(|g_1|, |g_2|)} \tag{1}$$

In this definition, $mcs(g_1, g_2)$ denotes the maximum common subgraph of g_1 and g_2, and $|g|$ is the number of nodes of graph g. First of all, one notices that no edit operations are involved in eq. (1) and, consequently, no cost function has to be defined to compute $\delta(g_1, g_2)$ for any two given graphs g_1 and g_2. Moreover, it was shown in (Bunke and Shearer, 1998) that the distance in eq. (1) has some interesting properties. In particular it is a metric satisfying

$$0 \leq \delta(g_1, g_2) \leq 1 \tag{2}$$

In this definition, $mcs(g_1, g_2)$ denotes the maximum common subgraph of g_1 and g_2, and $|g|$ is the number of nodes of graph g. First of all, one notices that no edit operations are involved in eq. (1) and, consequently, no cost function has to be defined to compute $\delta(g_1, g_2)$ for any two given graphs g_1 and g_2. Moreover, it was shown in (Bunke and Shearer, 1998) that the distance in eq. (1) has some interesting properties. In particular it is a metric satisfying

$$\delta(g_1, g_2) = 0 \Leftrightarrow g_1 \textit{ and } g_2 \text{ are isomorphic to each other} \tag{3}$$

$$\delta(g_1, g_2) = \delta(g_2, g_1) \tag{4}$$

$$\delta(g_1, g_3) \leq \delta(g_1, g_2) + \delta(g_2, g_3) \tag{5}$$

for any three graphs $g_1; g_2$ and g_3. It is known that the edit distance $d(g_1, g_2)$ introduced in Section 2 is a metric if and only if the underlying cost function

satisfies certain conditions (Bunke and Allerman, 1983). These conditions, however, may be too restrictive or counterintuitive for certain problem domains. But there are applications where metric properties of the underlying distance measure are very much desired. One example is information retrieval from image and video databases (Shearer, 1998). This area relies heavily on browsing to locate required database elements. Thus it is necessary for the distance measure to be well behaved to allow sensible navigation of the database. For example, property (2) makes sure that the range of all possible distances is known in advance, regardless of the particular objects to be compared. By means of property (3) objects have zero distance if and only if they are identical. Eq. (4) implies that the distance from any object A to any object B is the same as from B to A. Finally, because of the triangular inequality (5), we know that no two objects that are dissimilar to each other can be both similar to the same object. The graph distance measure according to eq. (1) is based on the maximum common subgraph of two graphs. Obviously, it can be regarded an alternative to graph edit distance as introduced in Section 2. However, it was recently shown that there is also a direct relation between graph edit distance and maximum common subgraph in the sense that graph edit distance and maximum common subgraph computation are equivalent to each other under a certain cost function (Bunke, 1997). In (Bunke, 1997) the following cost function was considered:

$$
\begin{aligned}
&c_{ns}(x)=\left\{\begin{matrix}0, & \textit{if } \alpha_1(x)=\alpha_2(f(x))\\ \infty, & \textit{otherwise}\end{matrix}\right\}\\
&c_{nd}(x)=1 \textit{ for any } \quad x\in V_1-\hat{V}_1,\\
&c_{ni}(x)=1 \textit{ for any } \quad x\in V_2-\hat{V}_2,\\
&c_{es}(e)=\left\{\begin{matrix}1, & \textit{if } \beta_1((x,y))=\beta_2((f(x),f(y)))\\ \infty, & \textit{otherwise}\end{matrix}\right\}\\
&\qquad \textit{for any } e=(x,y)\in \hat{V}_1\times\hat{V}_1,\\
&c_{ed}(e)=0 \textit{ for any } e=(x,y)\in (V_1\times V_1)-(\hat{V}_1\times\hat{V}_1),\\
&c_{ei}(e)=0 \textit{ for any } e=(x,y)\in (V_2\times V_2)-(\hat{V}_2\times\hat{V}_2)
\end{aligned}
\tag{6}
$$

Under this cost function, any node deletion and insertion has a cost equal to one. Identical node and edge substitutions have zero cost, while substitutions involving different labels have infinity cost. The insertion or deletion of an edge incident to a node that is inserted or deleted, respectively, has no cost. Intuitively speaking, it is assumed that the cost of a node deletion (insertion) includes the cost of deleting (inserting) the incident edges. As for any two graphs $g_1=(V_1,E_1,\alpha_1,\beta_1)$ and $g_2=(V_2,E_2,\alpha_2,\beta_2)$ there is always an *etgm* f with cost $c(f)=|V_1|+|V_2|$

(corresponding to the case where all nodes together with their incident edges are deleted from g_1, and all nodes with their incident edges are inserted in g_2), any edit operation with infinity cost will never need to be considered when looking for an optimal *etgm*. Thus we may think of edit operations with infinity cost as non-admissible. In other words, under the given cost function we can restrict our attention on *etgm*'s involving only insertions, deletions and identical node and edge substitutions, but no non-identical substitutions. For example, for the *etgm* f_3 discussed in Example 1, we have $c(f_3)=3$ under the considered cost function. Obviously both f_1 and f_2 have infinity cost. It was shown in (Bunke, 1997) that under this cost function the following equation holds true for any two graphs g_1 and g_2, and a maximum common subgraph g of g_1 and g_2 (this maximum common subgraph may be empty):

$$d(g_1,g_2)=|g_1|+|g_2|-2|g| \tag{7}$$

Obviously, this equation establishes a relation between the size |g| of the maximum common subgraph of two graphs g_1 and g_2, and their edit distance $d(g_1,g_2)$. Thus given one of the two quantities and the size of g_1 and g_2, we can immediately calculate the other. It was furthermore shown in (Bunke, 1997) that the mapping $f:\hat{V}_1\rightarrow\hat{V}_2$, defining an optimal *etgm* according to Def. 7, represents a maximum common subgraph of g_1 and g_2. I.e., f is a graph isomorphism between $\hat{g}_1$, the graph induced by $\hat{V}_1$, and $\hat{g}_2$, the graph induced by $\hat{V}_2$, and there are no larger subgraphs in g_1 and g_2, respectively, that are isomorphic to each other.

This theoretical result has an interesting practical consequence, namely, any algorithm for graph edit distance computation can be applied for maximum common subgraph computation if it is run under the cost function given in eq. (6). Conversely, any algorithm that computes the maximum common subgraph of two graphs can be used for graph edit distance computation under cost function eq. (6), using eq. (7). A similar relation between string edit distance and longest common subsequence has been known for long (Stephen, 1994).

The results derived in (Bunke, 1997) were recently shown to hold not only for the cost function given in (6), but for a whole class consisting of infinitely many cost functions. In (Bunke, 1999) cost functions C with $c_{ns}=c_{es}=0$ for identical substitutions and

$$c_{nd}+c_{ni}<c_{ns}\ and\ c_{nd}+c_{ni}<c_{es} \tag{8}$$

are considered. (Note that (6) is a special case of this class.) It is shown that for this whole class of cost functions the minimum cost mapping $f : \hat{V}_1 \rightarrow \hat{V}_2$ represents a maximum common subgraph of g_1 and g_2 and, conversely, any maximum common subgraph represents a minimum cost mapping in the sense of Definition 7. Intuitively speaking, the conditions in (8) imply that a node deletion together with a node insertion will be always preferred over a node or an edge substitution because of a smaller cost. This means that all nodes and edges in g_1 that can't be mapped to a node or an edge with an identical label in g_2 will be deleted from g_1. Similarly, all nodes and edges in g_2 that are not part of the mapping f (i.e, that don't have a corresponding node or edge with identical label, respectively) will be inserted. What remains for the mapping f is exactly the maximum common subgraph of g_1 and g_2. An example is the *etgm* f_3 in Example 1. It is optimal under the cost function $C'' = (1,1,7,1,1,7)$ as explained in Example 2. As a matter of fact, f_3 corresponds to the maximum common subgraph of g_1 and g_2 in Figure 1, and cost function C'' satisfies conditions (8). The equivalence of maximum common subgraph and graph edit distance computation shown in (Bunke, 1999) is based on the assumption $c_{ei} = c_{ed} = 0$ for any edge e from $(V_1 \times V_1) - (\hat{V}_1 \times \hat{V}_1)$ and $(V_2 \times V_2) - (\hat{V}_2 \times \hat{V}_2)$, respectively, see (6). Thus, no individual costs for the deletion of edges from $(V_1 \times V_1) - (\hat{V}_1 \times \hat{V}_1)$, and no individual costs for the insertion of edges in $(V_2 \times V_2) - (\hat{V}_2 \times \hat{V}_2)$ are taken into regard. The reason is that these operations are automatically implied by the deletion of nodes from $(V_1 - \hat{V}_1)$, and the insertion of nodes in $(V_2 - \hat{V}_2)$, respectively. Thus, it is assumed that their costs are included in the costs of the corresponding node deletions and insertions. In other words, the cost of a node deletion (insertion) includes not only the cost of deleting (inserting) a node, but also the deletion (insertion) of the edges that connect it to the other nodes of the graph. This assumption may be justified in many applications.

The equivalence of graph edit distance and maximum common subgraph shown in (Bunke, 1999) yields additional insight on the measure $\delta(g_1, g_2)$ of eq. (1). Although no *explicit* costs of graph edit operations are needed to compute $\delta(g_1, g_2)$, there are, nevertheless, costs involved in an *implicit* manner, because the quantity $|mcs(g_1, g_2)|$ in eq. (1) is equivalent to the graph edit distance $d(g_1, g_2)$ in the sense of eq. (7), assuming a cost function satisfying (8). In other words, whenever we compute the maximum common subgraph of two graphs we may consider this as a graph edit distance computation under an arbitrary cost function belonging to the class studied in (Bunke, 1998). From this point of view, the measure defined in eq. (1) may be regarded an advantage over conventional graph

edit distance computation because it is robust against changing the costs of the underlying graph edit operations over a fairly wide range.

Another important result shown in (Bunke, 1999) is the existence of classes of cost functions that always result in the same optimal mapping $f : \hat{V}_1 \rightarrow \hat{V}_2$ for any two given graphs g_1 and g_2. Intuitively speaking, if we consider two cost functions C and C', where C' is a scaled version of C, i.e., $c'_{nd} = \alpha c_{nd}, c'_{ni} = \alpha c_{ni}, \ldots, c'_{es} = \alpha c_{es}$ for some $\alpha > 0$, then we expect that any *etgm* f that is optimal under C is also optimal under C' for any two given graphs g_1 and g_2. Just the absolute cost of the two optimal *etgm*'s would differ by a factor α. In (Bunke, 1998) it was shown that any optimal *etgm* under a cost function C is optimal under another cost function C' not only if C' is a scaled version of C, but for a much larger class of cost functions C'. If the conditions

$$\frac{(c_{ni} + c_{nd})}{c_{ns}} = \frac{(c'_{ni} + c'_{nd})}{c'_{ns}} \tag{9}$$

and

$$\frac{c_{es}}{c_{ns}} = \frac{c'_{es}}{c'_{ns}} \tag{10}$$

for cost functions C and C' are satisfied then any *etgm* f is optimal under C if and only if it is optimal under C' for any two given graphs g_1 and g_2. Furthermore, there is a relation between the values $c(f)$ obtained under two different cost functions that is similar to eq. (7). Given the edit distance under cost function C we can analytically compute the edit distance under C' using just the parameters of C and C' and the size of the two graphs under consideration. Hence, given an algorithm that was designed for a particular cost function C, we can use the same algorithm for any other cost function C' for which (9) and (10) are satisfied. The existence of similar classes of cost functions for string edit distance has been discovered recently (Rice *et al.*, 1997).

As discussed above, maximum common subgraph computation is a special case of graph edit distance under a particular class of cost functions. It was furthermore shown in (Bunke, 1999) that also graph isomorphism and subgraph isomorphism are special cases of *etgm*. If we define $c_{nd} = c_{ni} = c_{ns} = c_{ed} = c_{ei} = c_{es} = \infty$ then an *etgm* f between g_1 and g_2 with $c(f) < \infty$ exists if and only if there exists a graph isomorphism between g_1 and g_2. Clearly, any such graph isomorphism f is optimal and $c(f) = 0$. Similarly, if

$$c_{nd} = c_{ns} = \infty,$$
$$0 \le c_{ni} < \infty,$$
$$c_{es} = c_{ed} = c_{ei} = \infty \quad if \quad e \in \hat{V}_1 \times \hat{V}_1,$$
$$c_{ed}(e) = 0 \quad if \quad e \in (V_1 \times V_1) - (\hat{V}_1 \times \hat{V}_1),$$
$$c_{ei}(e) = 0 \quad if \quad e \in (V_2 \times V_2) - (\hat{V}_2 \times \hat{V}_2)$$

then an optimal *etgm* f with $c(f) < \infty$ between g_1 and g_2 exists if and only if there exists a subgraph isomorphism from g_1 to g_2. Any optimal *etgm* f is in fact a subgraph isomorphism and $c(f) = (|g_2| - |g_1|) \cdot c_{ni}$.

4. ALGORITHMS FOR GRAPH MATCHING

All results presented in Section 3 are independent of the algorithm that is actually employed for graph edit distance or maximum common subgraph computation. In the past, various approaches to *etgm* have been proposed. The most common approach is based on tree search with A^*-like algorithms (Nilsson, 1980). The search space of the A^* algorithm can be greatly reduced by applying heuristic error estimation functions. Numerous heuristics have been proposed (Tsai and Fu, 1979; Shapiro and Haralick, 1981; Sanfeliu and Fu, 1983; Eshera and Fu, 1984; Wong, 1990). All of these methods are guaranteed to find the optimal solution but require exponential time and space in the worst case. 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 (Wilson and Hancock, 1994; Christmas *et al.*, 1995) probabilistic relaxation schemes are described. Other approaches are based on neural networks such as the Hopfield network (Feng *et al.*, 1994) or the Kohonen map (Xu and Oja, 1990). Also genetic algorithms have been proposed recently (Cross *et al.*, 1996; Wang *et al.*, 1997). In (Wang *et al.*, 1994) an approximate method based on maximum flow is introduced. However, all of these approximate methods may get tracked in local minima and miss the optimal solution. Optimal algorithms to find a maximum common subgraph of two graphs are based on maximum clique detection (Levi, 1972) or backtracking (McGregor, 1982). A suboptimal method using a neural network has been reported in (Shonkry and Aboutabl, 1996). For optimal graph and subgraph isomorphism detection see (Ullman, 1976).

In the remainder of this section we briefly review three optimal graph matching methods that were proposed recently. In (Messmer, 1995; Messmer and Bunke, 1998a) 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 (Messmer, 1995; Messmer and Bunke, 1998a) 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 (Messmer, 1995; Messmer and Bunke, 1999b) the same procedure is applied not to graph edit distance computation, but subgraph and graph isomorphism detection.

Figures 2 and 3 show the results of an experiment that was done to compare the new method with a traditional A^*-based algorithm for *etgm*. In this experiment random graphs were used as input. Figure 2 shows the computation time needed by the new and the traditional algorithm depending on a growing number of nodes in the graphs to be matched, keeping the number of errors, i.e., the edit distance between the two graphs, constant. Figure 3 shows a similar experiment where the size of the underlying graphs is kept constant but their edit distance is increased. The figures clearly show the superior performance of the new method. For further experimental results and a more detailed discussion see (Messmer, 1995; Messmer and Bunke, 1998a).

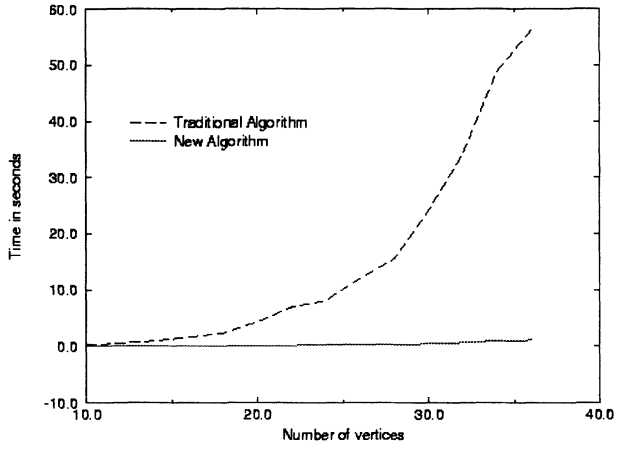

Figure 2. Computation time depending on the size of the underlying graphs for constant edit distance.

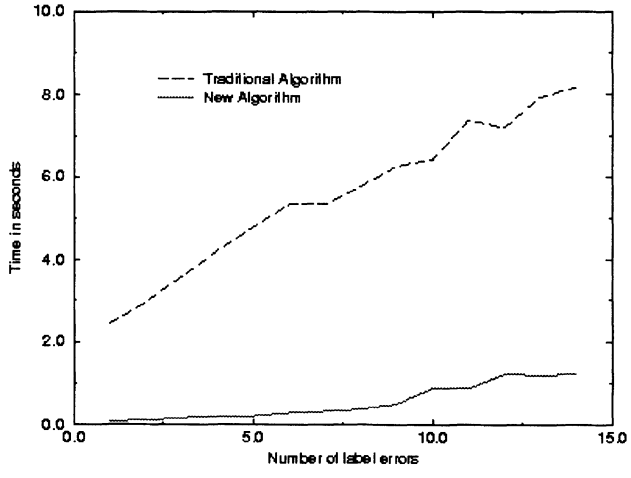

Figure 3. Computation time depending on the edit distance for constant size of the underlying graphs.

In (Messmer, 1995; Messmer and Bunke, 1999a) a fast algorithm for graph and subgraph isomorphism detection was 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 edges in any of the 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 *etgm* (Messmer and Bunke, 1998b). Actually, there are different possible approaches. 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 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 $O\left(\vartheta n^{2(\vartheta+1)}\right)$ 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.

Figures 4 and 5 show the results of an experiment where the second approach was compared to a conventional A^*-based *etgm* algorithm. In this experiment the threshold was set to $\vartheta = 1$. The input graphs were generated by copying one of the model graphs and then inserting or deleting an edge. Figure 4 shows the time needed by both algorithms when matching an input graph with one model graph depending

on the number of nodes of the input and model. In Figure 4 the input and the model graph consists of 11 vertices and the number of models is varied from 1 to 5. Figure 5 confirms the result of the theoretical complexity analysis, i.e., the time complexity of the decision tree algorithm is independent of the number of models in the database. The present implementation is limited to graphs consisting of up to a maximum of 16 nodes in case of just one error. For further details and additional experimental results see (Messmer and Bunke, 1998b).

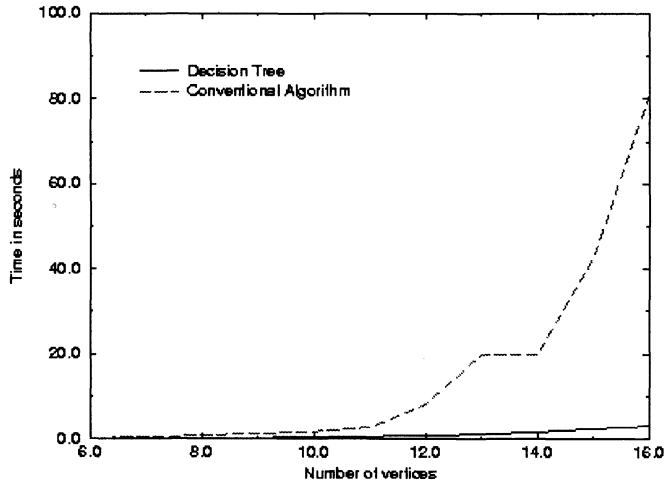

Figure 4. Computation time in seconds for $\vartheta = 1$ and a growing number of vertices

The decision tree approach was furthermore extended to maximum common subgraph detection (Shearer, Bunke, Venkatexh, Kieronska, 1998; Shearer, 1998). For this problem it is necessary, unfortunately, to consider all permutations of the adjacency matrix of the input graph, which leads to an exponential time complexity at run time despite the fact that all permutations of the models have already been encoded in the decision tree. Using a pruning strategy, however, the run time of the resulting algorithm is still significantly better than that of traditional algorithms. For further details and experimental results see (Shearer, Bunke, Venkatexh, Kieronska, 1998; Shearer, 1998).

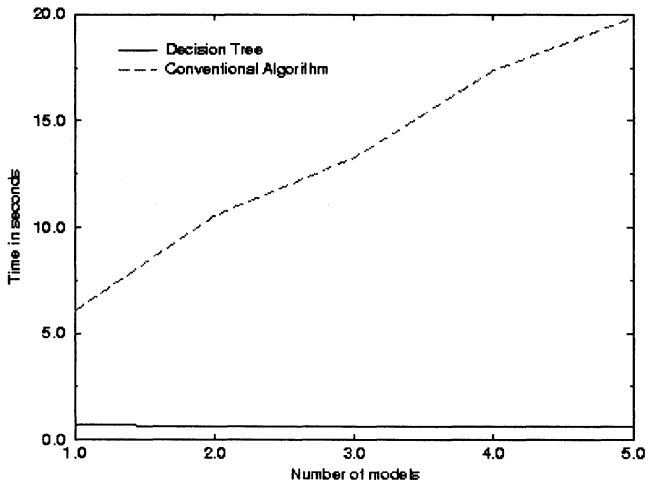

Figure 5. Computation time in seconds for $\vartheta = 2$ and a growing number of models

5. FURTHER REMARKS AND CONCLUSIONS

There are a few more issues in graph matching that have not been discussed in the previous sections. They will be briefly addressed in the following.

For clustering applications it is often desirable to represent a set of given graphs by just a single graph. We may think of the single representative as a kind of median or mean. Given a set of graphs $G = \{g_1, \ldots, g_N\}$, one possible solution is to select the graph g_i from G that has the smallest average edit distance to all members in G as a representative. That is, $g_i \in G$ is defined by requiring

$$\sum_{j=1}^{N} d(g_i, g_j) = \min\left\{\sum_{j=1}^{N} d(g_l, g_j) \middle| l = 1, \ldots, N\right\} \tag{11}$$

The implementation of this method is straightforward. It requires $O(N^2)$ edit distance computations. An interesting alternative to eq. (11) is to select a graph $\overline{g}$ from the universe U of all graphs – i.e. not necessarily from the set G – such that

$$\sum_{j=1}^{N} d(\overline{g}, g_j) = \min\left\{\sum_{j=1}^{N} d(g, g_j) \middle| g \in U\right\} \tag{12}$$

Apparently, this problem is much harder than the problem corresponding to eq. (11). Two algorithms for solving eq. (12) have been proposed recently (Jiang *et al.*, 1999a). The first solution is based on combinatorial search. However, as the considered problem is exponential in both the number and size of the elements of G, this solution is restricted to a small number of graphs with a few nodes only. For larger problems, a genetic algorithm was proposed. For details and application examples, see (Jiang *et al.*, 1999a).

In Section 3 it was pointed out that the maximum common subgraph of two graphs is a very useful concept. Another concept, minimum common supergraph of a pair of graphs, which is similar to maximum common subgraph, was proposed recently (Bunke *et al.*, 1999). It is defined as the smallest graph – in terms of number of nodes and edges – that contains the two given graphs as subgraphs. There are some interesting relationships between maximum common subgraph and minimum common supergraph. For example, the computation of the one can be reduced to the computation of the other. Moreover, not only maximum common subgraph, but also minimum common supergraph computation is a special case of edit distance computation under a particular class of cost functions. While maximum common subgraph may 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 the investigation of graph operators with some algebraic properties.

Approximate algorithms have become very popular recently, because of the exponential complexity of graph matching. There are, however, special subclasses of graphs where certain matching problems can be solved in polynomial time. In this context, much attention has been payed to the graph isomorphism problem. For instance, Luks (Luks, 1982) described a polynomially bounded method for the isomorphism detection of graphs with bounded valence. For the special case of trivalent graph isomorphism, it was shown in (Luks, 1982) that algorithms with a computational complexity of n^4 exist. Low-order polynomial-time methods (Hopcroft and Tarjan, 1973; Hopcroft and Weinberg, 1966) are also known for planar graphs. Further special graph classes, for which the isomorphism problem is solvable in polynomial time, are trees (Aho *et al.*, 1974), interval graphs (Booth and Lueker), permutation graphs (Colbourn, 1981), chordal (6,3) graphs (Babel, 1995),

graphs with bounded genus (Miller, 1980), graphs with bounded treewidth (Bodlaender, 1990), graphs with bounded eigenvalue multiplicity (Babai *et al.*, 1982), and rooted directed path graphs (Babel *et al*, 1996).

Additional classes of graphs have been discovered recently. In particular, so-called ordered graphs have been investigated in (Jiang and Bunke, 1998a; Jiang and Bunke, 1999). In an ordered graph the edges incident to a vertex possess a unique order. In many applications one is faced with ordered graphs where the ordering information is naturally derived from the underlying geometry of the patterns represented by the graphs. Note that plane graphs and triply connected planar graphs are special cases of ordered graphs. Consequently, the vertex-edge-graphs of polyhedra with no holes are ordered graphs since the graph of such polyhedra is triply connected and planar. Actually, even (non-planar) graphs of polyhedra with holes are ordered graphs as well. The reason is that we can always order the edges connected to a vertex in a natural way, for instance, clockwise if we look at the polyhedron from outside. In (Jiang and Bunke, 1998a; Jiang and Bunke, 1999) it was shown that the isomorphism problem for ordered graphs can be solved in $O(m_1 m_2)$ time where m_1 and m_2 represent the number of edges of two graphs. For this class of graphs, a special form of subgraph isomorphism has been considered in (Jiang and Bunke, 1998b). Under the assumption that the degree of some distinguished vertices is preserved under the subgraph isomorphism mapping, it was shown that the subgraph isomorphism problem is solvable in quadratic time as well.

It can be concluded that graphs constitute versatile and flexible representation formalism suitable for a wide range of problems in intelligent information processing. In many applications, graph matching, i.e. determining some kind of similarity of graphs, is an important issue. There has been steady progress in graph matching during the past years.

Recently, the focus of attention has shifted from "simple" combinatorial procedures, comparing two graphs at a time, to suboptimal stochastic algorithms and optimal algorithms that employ some kind of preprocessing to reduce the computational effort at run time. There are many interesting open problems in graph matching, for example, the combination of stochastic and preprocessing based optimal methods, a deeper study of the influence of the cost function on the complexity of matching algorithms, or the matching of dynamically changing graphs.

References

Aho, A.V., Hopcroft, J. E., and Ullman, J.D. (1974). *The Design and Analysis of Computer Algorithms*, Reading: Addison-Wesley.

Babai, L., Grigorer, D.Y., and Mount, D.Y. (1982)."Isomorphism of graphs with bounded eigenvalue multiplicity," *Proc. of 14th ACM Symposium on Theory of Computing*, pp. 310-324.

Babel, L. (1995). "Isomorphism of chordal (6,3) graphs", Computing, Vol. 54, pp. 303-316.

Babel, L., Ponomarenko, I.N., and Tinhofer, G. (1996). "The isomorphism problem for directed path graphs and for rooted directed path graphs," *Journal of Algorithms*, Vol. 21, pp. 542-564.

Balakrishnau, V. K. (1997). *Theory and Problems of Graph Theory*, McGraw-Hill.

Bodlaender, H. L. (1990). "Polynomial algorithms for graph isomorphism and chromatic index on partial *k*-tress", *Journal of Algorithms*, Vol. 11, pp. 631-643.

Booth, K.S. and Lueker, G.S. (1979). "A linear-time algorithm for deciding interval graph isomorphism," *JACM*, Vol. 26, pp. 183-195.

Borner, K., Pippig, E., Tammer, E., and Coulon, C. (1996). "Structural similarity and adaption," in I. Smith and B.Faltings (Eds.): *Advances in Case-based Reasoning*, Lectures Notes in Computer Science, Vol. 1168, Springer, pp. 58-75.

Bunke, H. (1997). "On a relation between graph edit distance and maximum common subgraph," *Pattern Recognition Letters*, Vol. 18, pp. 689-694.

Bunke, H. (1998). "Error-tolerant graph matching: a formal framework and algorithms",in A. Amin, D. Dori, P. Pudil, and H. Freeman (Eds.): *Advances in Pattern Recognition*, LNCS 1451, Springer Verlag, pp. 1-14.

Bunke, H. "Error correcting graph matching: On the influence of the underlying cost function," submitted for publication.

Bunke, H. and Allerman, G. (1983). "A metric on graphs for structural pattern recognition," in H. W. Schussler (Ed.): *Signal Processing II: Theories and Applications*, Elsevier Science Publishers B.V. (North-Holland).

Bunke, H., Jiang, X., and Kandel, A. "On the minimum common supergraph of two graphs", submitted for publication.

Bunke, H. and Shearer, K. (1998). "A graph distance metric based on maximal common subgraph," *Pattern Recognition Letters*, Vol. 19, Nos. 3-4, pp. 255-259.

Christmas, W. J., Kittler, J., and Petrou, M. (1995). "Structural matching in computer vision using probabilistic relaxation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 8, pp. 749-764.

Colbourn, G. J. (1981). "On testing isomorphism of permutation graphs," *Networks*, Vol. 11, pp. 13-21.

Cook, D. J. and Holder, L. B. (1994). "Substructure discovery using minimum description length and background knowledge", *Journal of Artificial Intelligence Research*, pp. 231-255.

Cross, A., Wilson, R., and Hancock, E. (1996). " Genetic search for structural matching," In B. Buxton, R. Cipolla (Eds.): *Computer Vision – FCCV'96, Lecture Notes in Comp. Science* 1064, Springer Verlag, pp. 514-525.

Ehrig, H. (1992). "Introduction to graph grammars with applications to semantic networks," *Computers and Mathematics with Applications*, Vol. 23, pp. 557-572, September.

Eshera, M.A. and Fu, K.S. (1984). "A graph distance measure for image analysis," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 14, No. 3, pp. 398-408.

Feng, J., Laumy, M., and Dhome, M. (1994)."Inexact matching using neural networks," In E.S. Gelsema and L.N. Kanal (Eds.): *Pattern Recognition in Practice IV: Multiple Paradigms*, Comparative Studies and Hybrid Systems, pp. 177-184. North-Holland.

Fisher, D.H. (1990). "Knowledge acquisition via incremental conceptual clustering," in J. W. Shavlik and T. G. Dietterich (Eds.): *Readings in Machine Learning*, pp. 267-283, Morgan Kaufmann.

McGregor, J. (1982). "Backtrack search algorithms and the maximal common subgraph problem," *Software-Practice and Experience*, Vol. 12, pp. 23-34.

Hopcroft, J. E. and Tarjan, R. E. (1973). "A *V log V* algorithm for isomorphism of triconnected planar graphs," *Journal of Computer and System Sciences*, Vol. 7, pp. 323-331.

Hopcroft, J. E. and Wong, J. K. (1974)."Linear time algorithm for isomorphism of planar graphs," *Proc. of 6th Annual ACM Symposium on Theory of Computing*, pp. 172-184.

Jiang, X. and Bunke, H. (1998). "On the coding of ordered graphs," *Computing*, Vol. 61, No. 1, pp. 23-38.

Jiang, X. and Bunke, H. (1998). "Marked subgraph isomorphism of ordered graphs," in A. Amin, D. Dori, P. Pudil, and H. Freeman (Eds.): *Advances in Pattern Recognition*, LNCS 1451, Springer Verlag, pp. 122-131, 1998.

Jiang, X. and Bunke, H. "Optimal quadratic-time isomorphism of ordered graphs," to appear in Pattern Recognition.

Jiang, X., Munger, A., and Bunke, H. (1999). "Combinatorial search vs. genetic algorithms: a case study based on the mean graph problem," to appear *in Proc. of Pattern Recognition in Practice VI.*

Jiang, X., Munger, A., and Bunke, H. "Synthesis of representative graphical symbols by mean graph computation," submitted for publication.

Lee, S. W., Kim, J. H., and Groen, F. C A. (1990). "Translation- rotation- and scale invariant recognition of hand-drawn symbols in schematic diagrams," *Int. Journal of Pattern Recognition and Artificial Intelligence*, Vol. 4, No. 1, pp 1-15.

Levi, G. (1972). "A note on the derivation of maximal common subgraphs of two directed or undirected graphs," *Calcolo*, Vol. 9, pp. 341-354.

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

Lu, S. W., Ren, Y., and Suen, C.Y. (1991). "Hierarchical attributed graph representation and recognition of handwritten Chinese characters," *Pattern Recognition*, Vol. 24, pp. 617-632.

Luks, E.M. (1982). "Isomorphism of graphs of bounded valence can be tested in polynomial time," *Journal of Computer and System Science*, Vol. 25, pp. 42-65.

Maher, P. (1993). "A similarity measure for conceptual graphs," *Int. Journal of Intelligent Systems*, Vol. 8, pp. 819-837.

Messmer, B.T. (1995). "*Efficient graph matching algorithms for preprocessed model graphs*," PhD thesis, University of Bern, Switzerland.

Messmer, B. T. and Bunke, H. (1996). "Automatic learning and recognition of graphical symboles in engineering drawings," in K. Tombre and R. Kasturi (Eds.): *Graphics Recognition*, Lecture Notes in Computer Science 1072, pp. 123-134, Springer Verlag.

Messmer, B.T. and Bunke, H. (1998). "A new algorithm for error tolerant subgraph isomorphism," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 20, pp. 493-505.

Messmer, B.T. and Bunke, H. (1998). "Error-correcting graph isomorphism using decision trees," *Int. Journal of Pattern Recognition and Art. Intelligence*, Vol. 12, No. 6, pp. 721-742.

Messmer, B.T. and Bunke, H. "A decision tree approach to graph and subgraph isomorphism," to appear in *Pattern Recognition.*

Messmer, B.T. and Bunke, H. "Efficient subgraph isomorphism detection – a decompostion approach," to appear in *IEEE Trans. on Data and Knowledge Engineering*.

Miller, G.L. (1980). "Isomorphism testing for graphs with bounded genus," *Proc. of 12th ACM Symposium on Theory of Computing*, pp. 225-235.

Nilsson, N.J. (1980). *Principles of Artificial Intelligence*, Tioga, Palo Alto.

Poole, J. (1993). "Similarity in legal case based reasoning as degree of matching in conceptual graphs," in M M. Richter, S. Wess, K.-D. Althoff, and F. Maurer (Eds.): *Preproceedings: First European Workshop on Case-Based Reasoning*, pp. 54-58.

Rekers, J. and Schurr, A. (1997). "Defining and parsing visual languages with layered graph grammars," *Journal of Visual Languages and Computing*, Vol. 8, pp. 27-55.

Rice, S., Bunke, H., and Nartker, T. (1997). "Classes of cost functions for string matching," *Algorithmica*, Vol. 18, No. 2, pp. 271-280.

Rocha, J. and Pavlidis, T. (1994). "A shape analysis model with applications to a character recognition system," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 16, pp. 393-404.

Rodgers, P.J. and King, P.J.H. (1997). "A graph-rewriting visual language for database programming," *Journal of Visual Languages and Computing*, Vol. 8, pp. 641-674.

Rouvray, D.H. and Balaban, A.T. (1979). "Chemical applications of graph theory," in R.J.Wilson and L.W.Beineke (Eds.): *Applications of Graph Theory*, pp. 177-221, Academic Press.

Sanders, K., Kettler, B., and Hendler, J. (1997). "The case for graph-structured representations,"in D. Leake and E. Plaza (Eds.): *Case-Based Reasoning Research and Development*, Lecture Notes in Computer Science, Vol. 1266, Springer, pp. 245-254.

Sanfeliu, A. and Fu, K.S. (1983). "A distance measure between attributed relational graphs for pattern recognition," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 13, pp. 353-363.

Shapiro, L.G. and Haralick, R.M. (1981). "Structural descriptions and inexact matching," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 3, pp. 504-519.

Shearer, K.R. (1998). *"Indexing and retrieval of video using spatial reasoning techniques,"* PhD thesis, Curtin University of Technology, Perth, Australia.

Shearer, K., Bunke, H., Venkatesh, S., and Kieronska, D. (1998). "Efficient graph matching for video indexing," *Computing*, Suppl 12 (Graph Based Representations in Pattern Recognition), pp. 53-62.

Shonkry, A. and Aboutabl, M. (1996). "Neural network approach for solving the maximal common subgraph problem," *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 26, pp. 785-790.

Shoubridge, P., Krarne, M., and Ray, D. (1999). "Detection of abnormal change in dynamic networks," *Proc. of IDC'99*, Adelaide, pp. 557-562.

Stephen, G.A. (1994). *String Searching Algorithms*, World Scientific, Publ. Co.

Tsai, W.H. and Fu, K.S. (1979). "Error-correcting isomorphisms of attributed relational graphs for pattern recognition," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 9, pp. 757-768.

Ullman, J.R. (1976). "An algorithm for subgraph isomorphism," *Journal of the Association for Computing Machinery*, Vol. 23, No. 1, pp. 31-42.

Wang, Y.-K., Fan, K.-C., and Horng, J.-T. (1997). "Genetic-based search for error-correcting graph isomorphism," *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 27, No. 4, pp. 588-597.

Wang, I., Zhang, K., and Chirn, G. (1994). "The approximate graph matching problem," *Proc. of 12th Int. Conf. on Pattern Recognition*, pp. 284-288, Jerusalem.

Weinberg, L. (1966). "A simple and efficient algorithm for determining isomorphism of planar triply connected graphs," *IEEE Trans. on Circuit Theory*, Vol. 13, No. 2, pp. 142-148.

Wilson, R. and Hancock, E. (1994). "Graph matching by discrete relaxation," In E.S. Gelsema and L.N. Kanal (Eds.): *Pattern Recognition in Practice IV*: Multiple Paradigms, Comparative Studies and Hybrid Systems, pp. 165-176. North-Holland.

Wong, E.K. (1990). "Three-dimensional object recognition by attributed graphs,"In H.Bunke and A.Sanfeliu (Eds.): *Syntactic and Structural Pattern Recognition-Theory and Applications*, pp. 381-414. World Scientific.

Wong, E.K. (1992). "Model matching in robot vision by subgraph isomorphism," *Pattern Recognition*, Vol. 25, No. 3, pp. 287-304.

Xu, L. and Oja, E. (1990). "Improved simulated annealing, Boltzmann machine, and attributed graph matching," In L.Almeida (Ed.): *Lecture Notes in Computer Science* 412, pp. 151-161. Springer Verlag.