Pattern Recognition Letters 31 (2010) 394–406

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

A graph matching method and a graph matching distance based on subgraph assignments

Romain Raveaux *, Jean-Christophe Burie, Jean-Marc Ogier

L3I Laboratory, University of La Rochelle, av M. Crépeau, 17042 La Rochelle Cedex 1, France

ARTICLE INFO

Article history:
Received 17 November 2008
Received in revised form 21 September 2009
Available online 24 October 2009

Communicated by A. Shokoufandeh

Keywords:
Graph matching
Graph distance
Bipartite graph matching
Graph based representation

ABSTRACT

During the last decade, the use of graph-based object representation has drastically increased. As a matter of fact, object representation by means of graphs has a number of advantages over feature vectors. As a consequence, methods to compare graphs have become of first interest. In this paper, a graph matching method and a distance between attributed graphs are defined. Both approaches are based on subgraphs. In this context, subgraphs can be seen as structural features extracted from a given graph, their nature enables them to represent local information of a root node. Given two graphs G_1, G_2, the univalent mapping can be expressed as the minimum-weight subgraph matching between G_1 and G_2 with respect to a cost function. This metric between subgraphs is directly derived from well-known graph distances. In experiments on four different data sets, the distance induced by our graph matching was applied to measure the accuracy of the graph matching. Finally, we demonstrate a substantial speed-up compared to conventional methods while keeping a relevant precision.

1. Introduction

Graphs are frequently used in various fields of computer science since they constitute a universal modeling tool which allows the description of structured data. The handled objects and their relations are described in a single and human-readable formalism. A graph G is a set of vertex (nodes) V connected by edges (links) E. Thus $G = (V, E)$. Tools for graphs supervised classification and graph mining are more and more required in many applications such as pattern recognition (Serrau et al., 2005), case-based reasoning (Antoine Champin and Solnon, 2003), chemical components analysis (Ralaivola et al., 2005) and semi-structured data retrieval (Schenker et al., 2004). To initiate the graph matching topic, we mention that a comprehensive survey of the technical achievements over the last 30 years is provided in (Conte et al., 2004).

In model-based pattern recognition problems, two graphs are given, the model graph G_M and the data graph G_D. The procedure for comparing them involves to check whether they are similar or not. Generally speaking, we can state the graph matching problem as follows: Given two graphs $G_M = (V_M, E_M)$ and $G_D = (V_D, E_D)$, with $|V_M| = |V_D|$, the problem is to find a one-to-one mapping $f : V_D \to V_M$ such that $(u, v) \in E_D$ *iff* $(f(u), f(v)) \in E_M$. When such a mapping f exists, this is called an isomorphism, and G_D is said to be isomorphic to G_M. This type of problem is known as exact graph matching. On the other hand, the term "inexact" applied to graph matching problems means that it is not possible to find an isomorphism between the two graphs. This is the case when the number of vertices or the labels are different in both the model and data graphs. Therefore, in these cases no isomorphism can be expected between both graphs, and the graph matching problem does not consist in searching for the exact way of matching vertices of a graph with vertices of the other, but in finding the best matching between them. This leads to a class of problems known as inexact graph matching. In that case, the matching aims at finding a non-bijective correspondence between a data graph and a model graph. If one of the graphs involved in the matching is larger than the other, in terms of the number of nodes, then the matching is performed by a subgraph isomorphism. A subgraph isomorphism from G_M to G_D means finding a subgraph sg of G_D such that G_M and sg are isomorphic.

Two drawbacks can be stated for the use of graph matching. Firstly, its computational complexity. This is an inherent difficulty of the graph matching problem. A brute-force approach requires a computational cost of $O(n!)$ for a graph with n nodes. The subgraph isomorphism is proven to be NP-complete (Mehlhorn, 1984). However, a research effort has been made to develop computationally tractable graph matching algorithms in particular applications. Such applications use some heuristics to cut down the computational effort to a manageable size. Graph matching can even be computed in polynomial time by using approximate algorithms under particular conditions. The second drawback is dealing with noise and distortion. The encoding of an object of an image by an attributed graph may not be perfect due to noise and errors

* Corresponding author. Fax: +33 5 46 45 82 42.
E-mail address: romain.raveaux01@univ-lr.fr (R. Raveaux).

0167-8655/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2009.10.011

introduced in low-level stages. In such situations, the presence of noise and distortion results in distorted graphs with different attribute values, missing or added vertices and edges, etc. This fact means exact graph matching is useless in many computer vision applications. The matching must incorporate an error model able to identify the distortions which make one graph a distorted version of the other. A matching between two graphs involving an error model is referred to as inexact graph matching and is computed by an error-correcting or error-tolerant (sub) graph isomorphism (Bunke and Messmer, 1997).

Several techniques have been put forward to solve the (sub) graph isomorphism problem, e.g. probabilistic relaxation (Bengoetxea et al., 2002; Coughlan and Ferreira, 2002; Christmas and Kittler, 1995), EM algorithm (Cross and Hancock, 1998; Luo and Hancock, 2000), neural networks (Lee and Park, 2002; Lee and Liu, 2000), decision trees (Messmer and Bunke, 1999) and a genetic algorithm (Cross and Hancock, 1996; Auwatanamongkol, 2007). Let us now give an overview of the main approaches and report on some of the most representative references. See reference (Lladós, 1997) for further study.

1.1. Error-tolerant algorithms

Concerning graph matching in the presence of noise and distortion, the procedural solutions to find an optimal error-tolerant subgraph isomorphism between two graphs are based on the construction of a state-space which is then searched with branch and bound techniques. A different approach to modelize the uncertainty of structural patterns was proposed by Wong and You (1985). They defined random graphs as a particular type of graphs which convey a probabilistic description of the data. Seong et al. (1994) developed a branch-and-bound algorithm to find the optimal isomorphism between two random graphs in terms of an entropy minimization formulation.

1.2. Approximate algorithms

Approximate or continuous optimization algorithms for graph matching offer the advantage that they can reach a solution in polynomial time and, moreover, they can solve both the exact and the inexact graph matching problem. However, since the similarity function which they minimize can converge in a local minimum, they may not find the optimal solution. Perhaps, the most successful of the optimization methods for graph matching use some form of probabilistic relaxation (Christmas and Kittler, 1995; Finch and Wilson, 1997; Gold and Rangarajan, 1996; Wilson and Hancock, 1996). The idea is similar to the discrete relaxation methods; however, the compatibility constraints between vertex-to-vertex assignments do not have a binary formulation, but are defined in terms of a probability function that is iteratively updated by the relaxation procedure. Another continuous optimization approach is based on neural networks (Kuner and Ueberreiter, 1988; Suganthan and Teoh, 1995; Suganthan and Teoh, 1995). The nodes of a neural network can represent vertex-to-vertex mappings and the connection weights between two network nodes represent a measure of the compatibility between the corresponding mappings. The network is programmed in order to minimize an energy (cost) function which is defined in terms of the compatibility between mappings. The problem of neural networks is that the minimization procedure is strongly dependent on the initialization of the network. Genetic algorithms is another technique used to find the best match between two graphs (Cross and Wilson, 1997; Ford and Zhang, 1992; Jiang et al., 2000). Vectors of genes are defined to represent mappings from model vertices to input vertices. These solution vectors are combined by genetic operators to find a solution.

Now that we have detailed the main concepts, let's introduce our proposal. In this paper, an error-tolerant graph matching algorithm is described. It is based on subgraph decomposition and wise use of the assignment problem. The assignment problem is one of the fundamental combinatorial optimization problems in the branch of optimization or operations research in mathematics. It consists of finding a maximum weight matching in a weighted bipartite graph.

In its proposed form, the problem is as follows:

- There are V_M number of subgraphs from G_M and V_D number of subgraphs from G_D. Any subgraph (sg_M) from G_M can be assigned to any subgraph (sg_D) of G_D, incurring some cost that may vary depending on the sg_M—sg_D assignment. It is required to map all subgraphs by assigning exactly one sg_M to each sg_D in such a way that the total cost of the assignment is minimized. This matching cost is directly linked to the cost function that measures the similarity between subgraphs.
- The adopted strategy tackles non-deterministic methods (i.e. evolutionary algorithms) thanks to a combinatorial optimization algorithm which confers a better stability, in such a way that for a given case, every time we run the program we will obtain the same results. Moreover, this combinatorial framework cuts down the algorithmic complexity to an $O(n^3)$ upper bound, depending on the number of nodes in the largest graph. Hence, the matching can be achieved in polynomial time which tackles the computational barrier. On the other hand, the number of calls to the graph distance is highly increased. In fact, n^2 calls to the cost function are needed to complete the weighted bipartite graph. This drawback is reasonably acceptable since the comparisons are performed on rather small subgraphs. Finally, the formulation into a bipartite graph matching offers the possibility to base the cost function on any kind of graph dissimilarity measures, making the system much more generic where the choice of the graph distance can be seen as a meta parameter.

All the later methods have as a common point the use of an optimization algorithm to best fit a graph into another. Note that in these cases, the fitness function measures the quality of the similarity. This function is designed taking into account the cost of mapping $V_D \rightarrow V_M$.

It is the author's belief that a suitable matching would lead to an accurate graph distance. According to this assumption, the performance evaluation question evolves into a graph distance problem. Furthermore, this point of view on the graph matching issue will allow a quantitative benchmark of our approach.

In the next section, a short survey is presented and graph distances used in this paper are introduced.

The rest of the paper is organized as follows: in Section 3, the proposed method is theoretically defined and explained. Section 4 is divided into two parts: The experimental evaluation of the algorithm is described and results are examined. Finally, some discussions conclude the paper.

2. Dissimilarity measures between graphs

All of the methods discussed here begin with a crisply labeled set of training data $T = \{\langle x_i, y_i\rangle\}_{i=1}^{L}$. Our presumption is that T contains at least one item with class label j, $1 \leqslant j \leqslant c$. Let x be an unlabeled object that we wish to label as belonging to one of c classes. The standard nearest prototype $1 - NN$ classification rule assigns x to the class of the "most similar" element in a set of labeled references. This notion of "the most similar one" is directly linked to the concept of graph distance. Hence, the graph classification problem can be stated as follows: it consists in inducing a mapping

$f(x) : \chi \rightarrow C$, from given training examples, $T = \{\langle x_i, y_i \rangle\}_{i=1}^{L}$, where $x_i \in \chi$ is a labeled graph and $y_i \in C$ is a class label associated with the training data.

Different approaches have been put forward over the last decade to tackle the problem of graph classification. A first one consists into transforming the initial problem in a common statistical pattern recognition problem by describing the objects with vectors in a Euclidean space. In such a context, some features (vertex degree, labels occurrence histograms, etc.) are extracted from the graph. Hence, the graph is projected in a Euclidean space and classical machine learning algorithms can be applied (Papadopoulos and Manolopoulos, 1999). Such approaches suffer from a main drawback: in order to have a satisfactory description of topological structure and graph content, the number of such features has to be very large and dimensionality issues occur.

Other approaches suggest using embeddings of the graphs in a Euclidean space of a given dimensionality using an optimization process. The aim of which is to best fit the distance matrix between each of the graphs. In such cases, a measure allowing graph comparison has to be designed. It is the case for multidimensional scaling methods proposed in (Bonabeau, 2002; Cox and Cox, 2001).

Another family of approaches also consists in using classical machine learning algorithms. At the opposite of the approaches mentioned above, the graphs are not explicitly but implicitly projected in a Euclidean space, through the use of a similarity measure adapted to the processed data in the learning algorithm.

In such a context, many kernel-based methods such as support vector machine or Kernel principal analysis were recently put forward (Kashima and Tsuboi, 2004; Borgwardt and Kriegel, 2005). They consist in designing an appropriate graph-based kernel for computing inner products in the graph space. Many kernels have been proposed in the literature (Suard et al., 2006; Mahé et al., 2004; Mahé et al., 2005). In most cases, the graph is embedded in a feature space composed of label sequences through a graph traversal. According to this traversal, the kernel value is then computed by measuring the similarity between label sequences. Even if such approaches have proven to achieve high performance, they suffer from a computationally intensive cost if the dataset is large (Vapnik, 1982). This problem of computational cost is not inherent to kernel-based methods. It also occurs when using other classification algorithms like k-NN. In conclusion, the problem of classifying graphs require the use of a fast but yet effective graph distance.

Our contribution in this paper is twofold; a sub-optimal inexact graph matching and a measure allowing to compare graphs with a low computational cost. This section offers a study of the different measures used to compare graphs in the context of nearest-neighbor search. Then, based on the accuracy and the performance, it justifies the choice of a measure based on subgraph assignments.

A dissimilarity measure is a function:

$$d : X \times X \rightarrow \Re,$$

where X is the representation space for the object description. It has the following properties:

- non-negativity

$$d(x,y) \geqslant 0, \tag{1}$$

- uniqueness

$$d(x,y) = 0 \Rightarrow x = y, \tag{2}$$

- symmetry

$$d(x,y) = d(y,x). \tag{3}$$

Measures of dissimilarity can often be transformed into measures of similarity (e.g. $s(x,y) = k - d(x,y)$, with k being a constant). If a dissimilarity measure also respects the triangle inequality (4), it is said to be a metric.

$$d(x,y) \leqslant d(x,z) + d(z,y). \tag{4}$$

Pseudo-metrics are another kind of function which allows to compare objects. Pseudo-metrics respect the non-negativity, symmetry and triangle inequality properties, but do not respect the uniqueness property. Pseudo-metrics can be obtained from dissimilarity measures, thanks to transformations that keep the order relation (e.g. $D(x,y) = \frac{d(x,y)}{1+d(x,y)} + 1$ (Gordon, 1999)).

The triangle inequality property is often used to optimize similarity search in metric spaces as it is done in (Vidal, 1994) or (Ciaccia et al., 1997), with direct application to classification (k-NN) and information retrieval tasks. When the compared objects are graphs, the uniqueness condition turns into an equivalence between a null dissimilarity and graph isomorphism. Graph isomorphism search is known to be a NP-Complete problem. However, if one defines a metric which is computationally tractable, then the graph isomorphism problem is also present. The edit distance (ED) is a dissimilarity measure for graphs that represents the minimum-cost sequence of basic editing operations to transform a graph into another graph by means of insertion, deletion and substitution of nodes or edges. Under certain conditions imposed to the cost associated with basic operations, the edit distance is a metric (Bunke and Shearer, 1998). In order to apply edit distance to a real world application, we have to consider that costs for basic operations are application dependent. This issue is tackled by automatic learning of cost functions (Neuhaus and Bunke, 2007). But, the edit distance computation also has a worst case exponential complexity which prevents its use in the context of nearest-neighbor search in large datasets.

2.1. Conditions for the edit distance being a metric

The original graph to graph correction algorithm defined elementary edit operations, $(a,b) \neq (\epsilon,\epsilon)$, where a and b are symbols from the two graphs or the NULL symbol, ϵ. Thus, changing symbol x to y is denoted (x,y), inserting y is denoted (ϵ,y), and deleting x is denoted (x,ϵ). Formally, the edit distance can be expressed as the sum of the edit operations to change a graph G_1 into a subgraph G_2.

$$d_{ED}(G_1,G_2) = \min_{(e_1,\ldots,e_k)\in\gamma(G_1,G_2)} \sum_{i=1}^{k} (edit(e_i)),$$

where $\gamma(G_1,G_2)$ denotes the set of edit paths transforming G_1 into G_2, and $edit$ denotes the cost function measuring the strength $edit(e_i)$ of edit operation e_i. From the conclusion drew in (Myers et al., 2000), an interesting property of this quantity is that it is a metric if $edit(e_i) > 0$ for all non-identical pairs and 0 otherwise, and if $edit(e_i)$ is self-inverse.

In order to define measures of dissimilarity between complex objects (sets, strings, graphs, etc.), another possibility is to base the measure on the quantity of shared terms. The simplest similarity measure between two complex objects o_1 and o_2 is the matching coefficient mc, which is based on the number of shared terms.

$$mc = \frac{o_1 \wedge o_2}{o_1 \vee o_2}, \tag{5}$$

where $o_1 \wedge o_2$ denotes the intersection of o_1, o_2 and $o_1 \vee o_2$ stands for the union between the two objects.

Based on this idea, dissimilarity measures which take into account the maximal common subgraph (mcs) of two graphs were put forward:

$$d(G_1,G_2) = 1 - \frac{mcs(G_1,G_2)}{max(|G_1|,|G_2|)}, \tag{6}$$

where $|G|$ denotes a combination of the number of nodes and the number of edges in G. From Eq. (5), the expression $o_1 \vee o_2$ is substituted by the size of the largest graph and the intersection of two graphs ($o_1 \wedge o_2$) is represented by the maximum common subgraph.

$$d(G_1, G_2) = 1 - \frac{mcs(G_1, G_2)}{|G_1| + |G_2| - mcs(G_1, G_2)}, \quad (7)$$

where $mcs(G_1, G_2)$ is the largest subgraph common to G_1 and G_2, i.e. it cannot be extended to another common subgraph by the addition of any vertex or edge.

The edit distance (ED) and the size of mcs observe the following equation:

$$ED(G_1, G_2) = |G_1| + |G_2| - 2|mcs(G_1, G_2)|. \quad (8)$$

As long as the cost functions associated to the edit distance respect the conditions presented in (Bunke and Shearer, 1998). The way to calculate the mcs size of two graphs can be used to compute the edit distance and viceversa. Then, both methods share the same computational complexity. Due to the difficulty in applying these metrics, several approaches relying on different types of approximations were proposed in (Hidovic and Pelillo, 2004). Three other group of techniques can be employed to evaluate graph similarity, spectral graph theory (Robles-Kelly and Hancock, 2005), probabilistic methods (Myers et al., 2000) or combinatorial optimization (Gold and Rangarajan, 1996; Peter Kriegel and Schonauer, 2003).

Among them, the node/edge matching distance (NMD) proposed in (Peter Kriegel and Schonauer, 2003) is a combinatorial optimization problem. It is based on the approximation of the topological conservation of isomorphism by the search of a minimum cost matching between two nodes set. The matrix cost for matching different labeled nodes serves as an input for the Hungarian algorithm. The node matching distance between two graphs G_1 and G_2 results in the cost of the minimum-weight edge matching which is given with a worst case complexity of $O(n^3)$, where n is the largest number of edges. The node cost function has to be determined taking into account a distance label matrix. The node matching distance for attributed graphs respects the non-negativity (1), symmetry (3), triangle inequality (4) properties from the metric definition as it is shown in (Peter Kriegel and Schonauer, 2003). Recently, Shokoufandeh et al. (2006) draws on spectral graph theory to derive a new algorithm for computing node correspondence. In computing a bipartite matching of nodes where their topological contexts is embedded into structural signature vectors.

A faster technique for estimating graph similarity consists in extracting a graph description as a vector of probes. This method, called graph probing proposed by Lopresti and Wilfong (2003), can deal with graphs with hundreds or thousands of vertices and edges in linear time and can be applied to directed attributed graphs.

Definition. Let G be a directed attributed graph and let L denote a finite set of edge labels: $\{l_1, l_2, \ldots, l_a\}$. Based on this notation, the edge structure of a given vertex can be described with a numerical vectors composed of a 2a-tuple of non-negative integers $\{x_1, x_2, \ldots, x_a, y_1, y_2, \ldots, y_a\}$ such that the vertex has exactly x_i incoming edges labeled l_i, and y_j outgoing edges labeled l_j.

The Fig. 1 illustrates the principle of construction of an edge structure for a given vertex. In this context, two types of probes are defined:

- $Probe1(G)$: a vector which gathers the counts of vertices sharing the same edge structure, for all encountered edge structures.
- $Probe2(G)$: a vector which gathers the number of vertices for each vertex label.

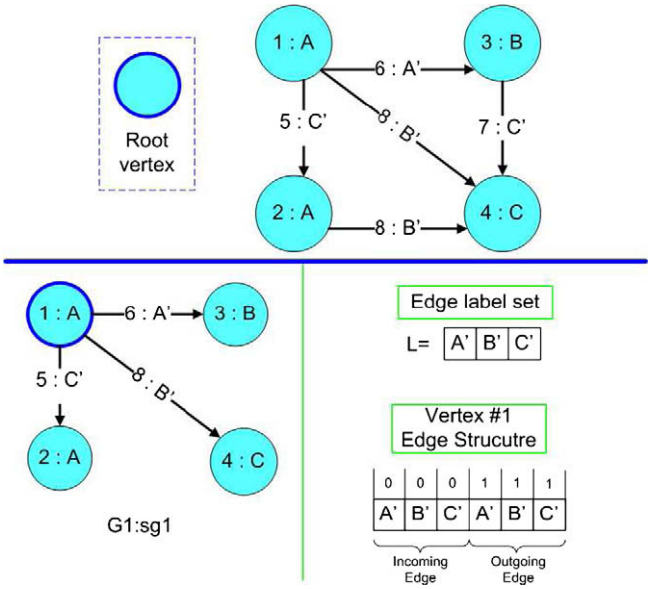


Fig. 1. Edge structure of a vertex in the graph probing context.

The Fig. 1 illustrates the principle of construction of an edge structure for a given vertex. Based on these probes and on the 1-norm $L1$, the graph probing distance is defined as:

$$GP(G_1, G_2) = L1(Probe1(G_1), Probe1(G_2)) + L1(Probe2(G_1), Probe2(G_2)).$$

The graph probing distance (GP) only respects the non-negativity, symmetry, and triangle inequality properties from the metric definition, but not the uniqueness property. In other words, GP is a pseudo-metric and two non-isomorphic graphs can have the same graph probes. However, a upper bound relation within a factor of four exists between the graph probing and the edit distance (Bunke and Shearer, 1998).

$$GP(G_1, G_2) \leqslant 4 \cdot ED(G_1, G_2). \quad (9)$$

In this context, the graph topology can be partially ignored by counting the number of occurrences of a set of subgraphs (named fingerprints or probes in different contexts) from each graph and to describe the objects to be compared as vectors. Consequently, this histogram view of a graph cannot lead to an univalent mapping process.

2.2. Comparison with the related work

In (Lopresti and Wilfong, 2003), Wilfong and Lopresti proposed a graph decomposition into an histogram where histogram bins are very simple sub-structures coded as numerical vectors. This strong assumption implies sub-elements to be very simple in term of structural information while cutting off drastically the computation time. This histogram viewpoint makes the graph matching computation not feasible loosing relationship between items. Instead of an histogram organization, in our case, the information is laid out in a bipartite graph, hence, a point to point mapping can be carried out.

In (Shokoufandeh et al., 2006), a "topological signature vector" described the structural context of a node. This vector was derived from the spectral properties of the directed acyclic subgraph rooted at that node. Thereby, a bipartite graph was defined between the nodes in two graphs, and edge costs were distances between two nodes' corresponding signatures, see Fig. 2. In such a way, the structural information is partially ignored to be embedded into a numerical vector.

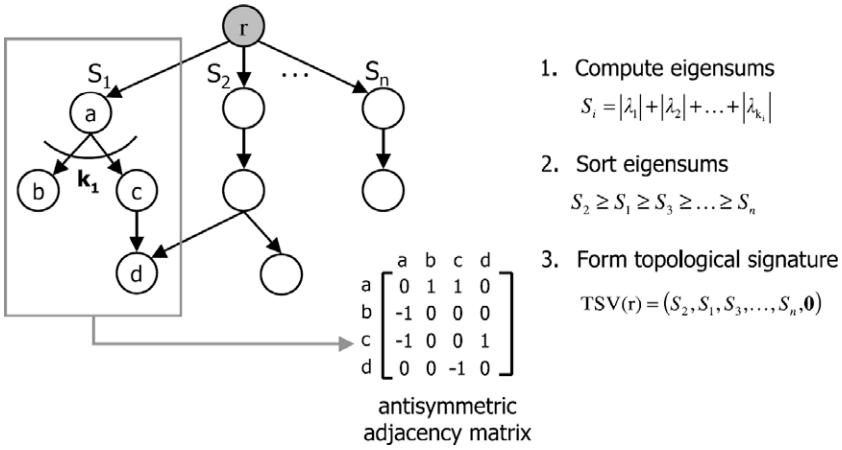


Fig. 2. Forming the structural signature.

On the contrary, our strong point is the combination of a graph data structure encoding combined with a bipartite matching procedure to find the optimal match. This formal description gives good properties to our method. The subgraph decomposition makes different graph distances applicable, thus, a wise use of the past-work in this field of science can be done.

By now, from the original idea stated in (Peter Kriegel and Schonauer, 2003; Shokoufandeh et al., 2006), the minimum cost matching between two element sets, the authors extended this paradigm to more complex and discriminating objects called subgraphs. Where a subgraph takes into account the vertex information and its neighborhood context. The rest of the paper will present a new metric that involves an univalent subgraph mapping that involves adjacent vertices into the matching process.

3. Subgraph matching and subgraph matching distance (*SGMD*)

3.1. Definition and notation

3.1.1. Graph definition

In this work, the problem which is considered concerns the matching of directed labeled graphs. Such graphs can be defined as follows: Let L_V and L_E denote the set of node and edge labels, respectively. A labeled graph G is a 4-tuple $G = (V, E, \mu, \xi)$, where

- V is the set of nodes,
- $E \subseteq V \times V$ is the set of edges,
- $\mu : V \rightarrow L_V$ is a function assigning labels to the nodes, and
- $\xi : E \rightarrow L_E$ is a function assigning labels to the edges.

3.1.2. Subgraph decomposition

From this definition of a given graph, the subparts for the matching problem can be expressed as follows:

Let G be an attributed graph with edges labeled from the finite set $\{l_1, l_2, \ldots, l_a\}$. Let SG be a set of subgraphs extracted from G. There is a subgraph sg associated to each vertex of the graph G. A subgraph (sg) is defined as a structure gathering the edges and their corresponding ending vertices from a root vertex. In such a way, the neighborhood information of a given vertex is taken into account. A subgraph represents a local information, a "star" structure from a root node. The mapping of these subparts should lead to a meaningful graph matching approximation. The subgraph extraction is done by parsing the graph which is achievable in linear time through the joint use of the adjacency matrix. The subgraph decomposition is illustrated in Fig. 3.

3.2. Subgraph matching

Let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ be two attributed graphs. Without loss of generality, we assume that $|SG_1| \geqslant |SG_2|$. The complete bipartite graph $G_{em}(V_{em} = SG_1 \cup SG_2 \cup \triangle, SG_1 \times (SG_2 \cup \triangle))$, where $\triangle$ represents an empty dummy subgraph, is called the subgraph matching graph of G_1 and G_2. A subgraph matching between G_1 and G_2 is defined as a maximal matching in G_{em}. We define the matching distance between G_1 and G_2, denoted by $SGMD(G_1, G_2)$,

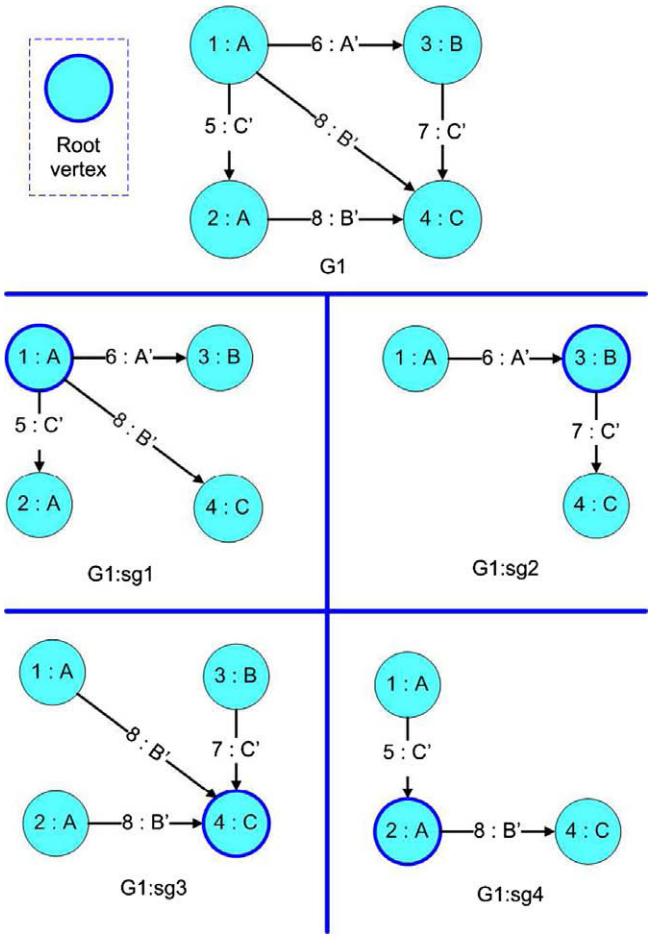


Fig. 3. Graph decomposition into subgraph world.

as the cost of the minimum-weight subgraph matching between G_1 and G_2 with respect to the cost function c' (i.e. Section 3.3). This optimal subgraph assignment induces an univalent vertex mapping between G_1 and G_2, such as the function $SGMD : SG_1 \times (SG_2 \cup \triangle) \rightarrow \Re_0^+$ minimized the cost of subgraph matching. If the numbers of subgraphs are not equal in both graphs, then empty "dummy" subgraphs are added until equality $|G_1| = |G_2|$ is reached. The cost to match an empty "dummy" subgraph is equal to the cost of inserting a whole unmapped subgraph $(c'(\emptyset, sg))$. The approximation lies in the fact that the vertex mapping is not executed on the whole structure, but more likely for subparts of it. The node matching is only constrained by the assumption of "close" neighborhood imposed by the subgraph viewpoint of a vertex. Why such a restriction? The mapping of two graphs when considering the entire structure is closely coupled with the maximum common subgraph search which is known to be a NP-Complete dilemma. More likely, this paper adopts a "Divide and Conquer strategy". An example of graph matching is proposed in Fig. 4.

3.3. Cost matrix construction

Definition. The Assignment Problem. Let us assume there are two sets A and B together with an $n \times n$ cost matrix C of real numbers given, where $|A| = |B| = n$

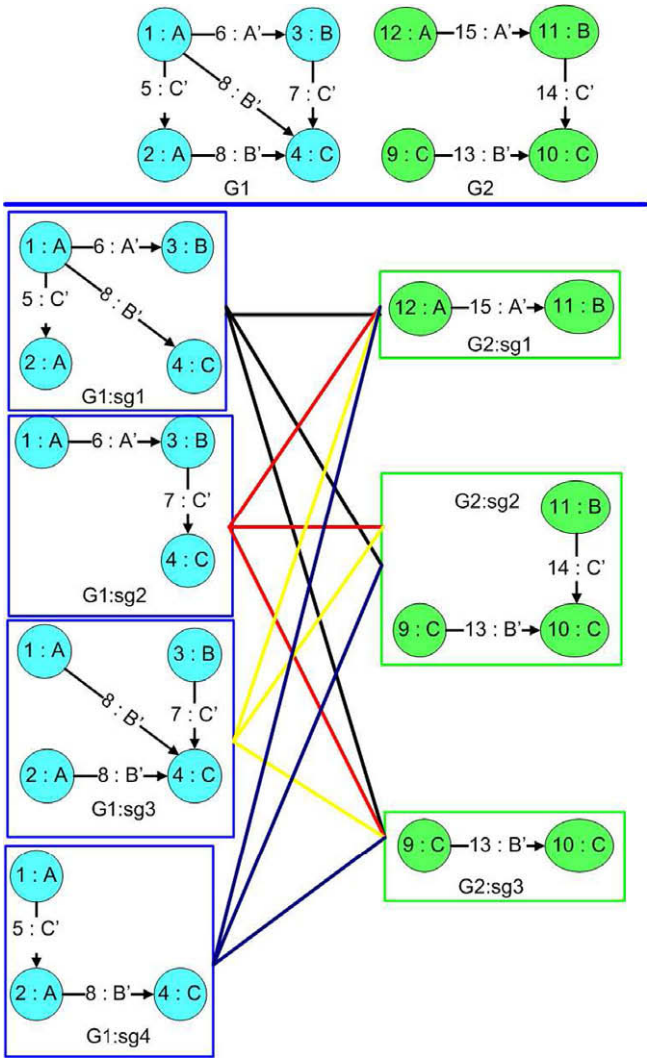


Fig. 4. Subgraph matching: a bipartite graph.

The matrix elements C_{ij} correspond to the costs of assigning the ith element of A to the jth element of B. The assignment problem can be stated as finding a permutation $p = p_1, p_2, \ldots p_n$ of the integers $1, 2, \ldots, n$ that minimizes $\sum_{i=1}^{n} C_{ij}$

In our approach, the cost matrix contains the distances between every pair of subgraphs from G_1 and G_2. The cost matrix C' is a $n \times n$ matrix where $n = \max(|G_1|, |G_2|) = \min(|G_1|, |G_2|) + |\Delta|$.

$$C' = \begin{vmatrix} c'_{1,1} & \ldots & \ldots & c'_{1,m} \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ c'_{n,1} & \ldots & \ldots & c'_{n,m} \end{vmatrix},$$

where $c'_{i,j}$ denotes the cost between two subgraphs. According to our formalism, a subgraph of depth "1" is defined from a root node. Hence, any graph distances can be applied to build that cost matrix. A straightforward comment, our method does not strictly rely on the edit distance.

With the aim of highlighting this difference of paradigm, a graph distance called Graph Probing (Lopresti and Wilfong, 2003) is also evaluated. Therefore $SGMD_{ED}$ and $SGMD_{GP}$ will respectively denote a graph matching based on edit distance or on graph probing.

3.4. The subgraph matching distance for attributed graphs is a pseudo metric

Proof. To show that the subgraph matching distance ($SGMD$) is a pseudo metric, we have to prove three properties for this similarity measure.

- $SGMD(G_1, G_2) \geqslant 0$
  The subgraph matching distance between two graphs is the sum of the cost for each subgraph matching. As the cost function is non-negative, any sum of cost values is also non-negative.
- $SGMD(G_1, G_2) = SGMD(G_2, G_1)$
  The minimum-weight maximal matching in a bipartite graph is symmetric, if the edges in the bipartite graph are undirected. This is equivalent to the cost function being symmetric. As the cost function is a metric, the cost for matching two subgraphs is symmetric. Therefore, the subgraph matching distance is symmetric.
- $SGMD(G_1, G_2) \leqslant SGMD(G_1, G_2) + SGMD(G_2, G_3)$
  As the cost function is a metric, the triangle inequality holds for each triple of subgraphs in G_1, G_2 and G_3 and for those subgraphs that are mapped to an empty subgraph. The subgraph matching distance is the sum of the cost of the matching of individual subgraphs. Therefore, the triangle inequality also holds for the subgraph matching distance. □

The subgraph matching distance respects the non-negativity, symmetry, and triangle inequality properties from the metric definition, but not the uniqueness property. In other words, $SGMD$ is a pseudo-metric and two non-isomorphic graphs can have the same graph subgraphs.

4. Experiments

This section is devoted to the experimental evaluation of the proposed approach. All tests based on a simple idea; the more significant is the distance induced by a graph matching, the better the matching is. This assumption turns the question into a graph distance comparison. Both data sets and the experimental protocol are firstly described before investigating and discussing the merits

of the proposed approach. In this practical work, the exact graph edit distance was provided by the SUBDUE substructure discovery system (SUBDUE), while other methods were re-implemented by us from the literature.

4.1. Databases in use

In recent years the use of graph based representation has gained popularity in pattern recognition and machine learning. As a matter of fact, object representation by means of graphs has a number of advantages over feature vectors. Therefore, various algorithms for graph based machine learning have been proposed in the literature. However, in contrast with the emerging interest in graph based representation, a lack of standardized graph data sets for benchmarking can be observed. In order to overcome this difficulty, we chose to carry out our tests on four databases. The first one is composed of synthetic data allowing an evaluation in a general context on a huge dataset. The others sets are domain specific, they are related to pattern recognition subjects where graphs are meaningful. The content of each database is summarized in Table 1.

4.1.1. Synthetic dataset: Base A

This data set contains over 28,000 graphs, uniformly distributed into 50 classes. The graphs are directed with edges and nodes labeled from two distinct alphabets. As the generic framework used to construct random graphs proposed in (Erd"os and Rényi, 1959) does not have the aim to depict classes, in the sense of similar graphs, we proposed a two step process to create classes of graphs. In a first step a number N (where N is the desired number of classes) of graphs are constructed using the Erdös–Rényi model (Erd"os and Rényi, 1959). The input of this model is the number of vertices of the graph to be generated, and the probability of having an edge between two nodes. Having a low probability for edges leads to sparse graphs, that occur frequently in proximity-based graph representations found in pattern recognition (see Section 4.1.3). In a second step each of these graphs are modified by edge and vertex deletion or relabeling. A second stage of modifications is applied, by selecting a node from a graph and replacing it with a random subgraph. This process leads to graph classes where intra class similarity is greater than inter class similarity. Numerical details concerning this data set are presented in Table 1. The large size of this data set is a key point for scaling up our approach.

4.1.2. Symbol recognition related data set: Base B

Our data is made of graphs corresponding to a corpus of 170 noisy symbol images, generated from 10 ideal models proposed in a symbol recognition contest (Valveny and Dosch, 2004), (GREC workshop). In a first step, considering the symbol binary image, we extract both black and white connected components. These connected components are automatically labeled with a partitional clustering algorithm (Kaufman and Rousseeuw, 1990), applied on a set of features called Zernike moments (Khotanzad and Hong, 1990). Using these labeled items, a graph is built. Each connected component represents an attributed vertex in this graph. Edges are then built using the following rule: two vertices are linked with an undirected and unlabeled edge if one of the nodes is a neighbor of the other node in the corresponding image. An example of the association between two symbol images and the corresponding graphs is illustrated in Fig. 5. Further details on this data set are presented in Table 1.

Table 1
Characteristics of the four data sets used in our computational experiments.

	Base A	Base B	Base C	Base D
Number of classes (N)	50	10	32	15
|Training|	14,128	114	9600	5062
|Validation|	14,101	56	3200	1688
Average number of nodes	12.03	5.56	8.84	4.7
Average number of edges	9.86	11.71	10.15	3.6
Average degree of nodes	1.63	4.21	1.15	1.3

4.1.3. Ferrer data set: Base C

In (Ferrer et al., 2006), a structural representation is extracted from a collection of graphical symbols, 12,800 images are distributed among 32 classes. These images of symbols, without rotation and scaling changes are derived from the GREC database (Valveny and Dosch, 2004). When examining symbol samples in Fig. 6, it is clear that their construction is based on straight-lines. Each segment terminates either with a terminal point or a junction point (the confluence point between two or more segments). For convenience, from now to the end of this work, we will refer to these kinds of points as TP and JP, respectively.

In order to prove the robustness of the prototypes against noise, four different levels of distortion were introduced. Distortion is generated by moving each TP or JP randomly within a circle of radius r, given as a parameter for each level, centered at original coordinates of the point. If a JP is randomly moved, all the segments connected to it are also moved. With such distortion, gaps in line segments, missing line segments and wrong line segments are not allowed. But the number of nodes of each symbol is not changed. Fig. 2 shows an example of such distortions. For each class and for each distortion 100 noisy images are created. Thus for each class we have 400 elements (100 for each distortion), straightforwardly, the amount of images is 12,800 (32×400).

In Ferrer's case, a symbol is represented as an undirected labeled graph, where the TPs and JPs are represented as nodes. Edges correspond to the segments connecting those points. The information associated to nodes or edges are their coordinates (x,y). Due to the graph spectral theory limitation, Ferrer's graphs are labeled using real positive or null values. Consequently, this restriction leads to the construction of two graphs for a single symbol, a graph G_x labeled with x coordinates and G_y with y coordinates. In our case, the subgraph distances impose the use of nominal labels. A

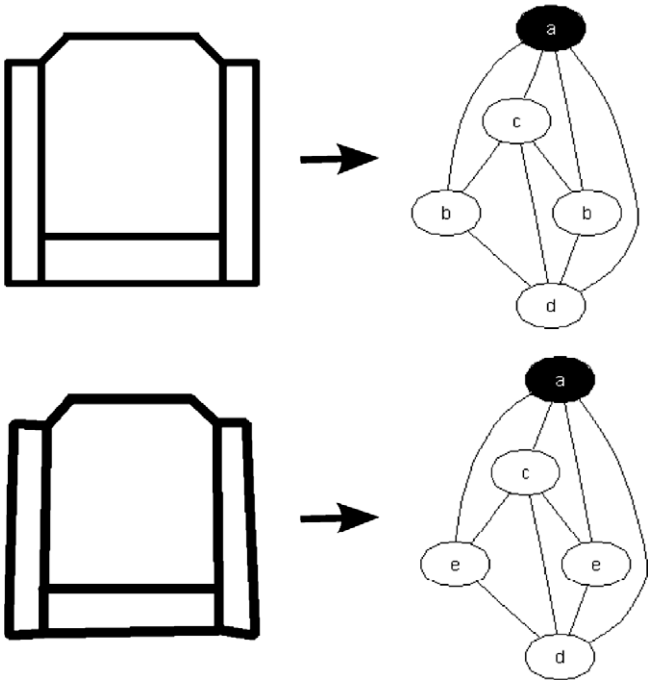


Fig. 5. From symbols to graphs through connected component analysis.

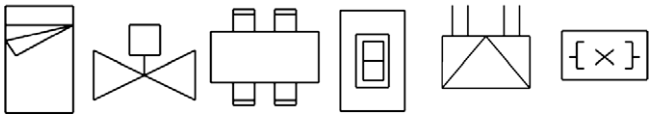


Fig. 6. Symbol samples.

2-Dimensional mesh aims to achieve the JP and TP discretization (i.e. Fig. 7). In addition, an experimental study which is not presented in this paper has been used in order to choose mesh granularity.

4.1.4. Letter database: Base D

The last database used in the experiments consists of graphs representing distorted letter drawings (IAM). In this experiment we consider the 15 capital letters of the Roman alphabet that consists of straight-lines only (*A*, *E*, *F*, etc.). For each class, a prototype line drawing is manually constructed. To obtain arbitrarily large sample sets of drawings with arbitrarily strong distortions, distortion operators are applied to the prototype line drawings. This results in randomly shifted, removed, and added lines. These drawings are then converted into graphs in a simple manner by representing lines by edges and ending points of lines by nodes. Each node is labeled with a two-dimensional attribute giving its position. Since our approach only focuses on nominal attributes, a quantification is performed by the use of a mesh, as in the case of database C and more information concerning that data is detailed in Table 1.

4.2. Protocol

Two ways for assessing our approach are proposed. Firstly, a statistical framework was designed to score the relation between our approach and the edit distance. Secondly, a pattern recognition stage was undertaken to measure-up the behavior in classification.

- In the first experiment, we assess the correlation concerning the responses to *k*-NN queries when using edit distance (*ED*) or subgraph matching distance (*SGMD*) as dissimilarity measures. The setting is the following: in a graph data set (Base D), we select a number *M* of graphs, that are used to query the rest of the data set by similarity. Top *k* responses to each query obtained in the first place using edit distance and subgraph matching distance are compared using the Kendall correlation coefficient. We consider a null hypothesis of independence(*H*0) between the two responses and then, we compute, by means of a two-sided statistical hypothesis test, the probability (*p*-value) of getting a value of the statistic as extreme or more extreme than observed by chance alone, if *H*0 is true. The Kendall's rank correlation measures the strength of monotonic association between the vectors x and y containing k elements. (x and y may represent ranks or ordered categorical variables). Kendall's rank correlation coefficient τ may be expressed as

$$\tau = \frac{S}{D},$$

where

$$S = \sum_{i<j} (sign(x[j] - x[i]) \cdot sign(y[i] - y[j])), \tag{10}$$

and

$$D = \frac{k(k-1)}{2}. \tag{11}$$

  In a second step, the distance matrices($M \times M$) between *ED* and *SGMD* are evaluated using the Pearson correlation. Finally, these two steps are repeated to compare the responses to *k*-NN queries when using edit distance (*ED*) or node matching distance (NMD) or Graph Probing (GP).
- The classification stage is the last experiment. It consists in a graph classification stage. Let $X = \{x_1, \ldots, x_n\}$ a crispy labeled set of training data. Our presumption is that X contains at least one graph with class label $i, : 1 < i < c$. Let x be an unlabeled object that we wish to label as belonging to one of c classes. The standard nearest-neighbor ($1 - NN$) classification rule assigns x to the class of the *most similar* prototype in a set of labeled training data (or reference set). Why use a nearest prototype classifier? Because the graph classification problem is defined in a dissimilarity space, only graph kernel-based classifiers and a *k*-NN classifier can be used to categorize objects in such a space. The 1-NN classifier is pertinent in our context, since it is simple (parameterless) and often, pretty accurate. Hereafter, $E_{np}(X_{tr}; X_{test})$ denotes the test error committed by the 1-NN rule that uses X_{test} when applied to the training data.

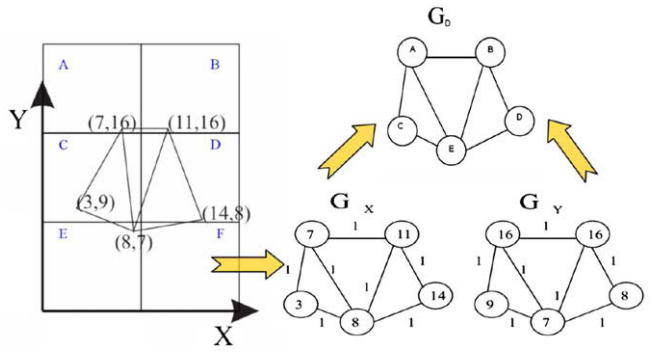


Fig. 7. From symbols to graphs using a 2D mesh.

4.3. Correlation between SGMD and edit distance

4.3.1. Kendall test

Using $M = 1200, k = 30$, we present in Fig. 8, the results obtained in terms of τ values. As the differences become larger when considering elements farther and farther from the query, the fact of dealing with a huge number of significant neighbors (k) may not be relevant. The notion of order could be simply corrupted by a saturation phenomenon directly implied by very high distances. Hence, from our point of view the examination of the 30 closest neighbors is fair enough. From the 1200 tests, only 124 have a *p*-value greater than 0.05, so we can say that the hypothesis *H*0 of independence can be rejected in 89.67% cases, with a risk of 5%. The observed correlation between the responses to *k*-NN queries when using edit distance (*ED*) and node matching distance(*NMD*) tends to reveal a rank relation between *ED* and $SGMD_{ED}$ (median value of $\tau = 0.733$). Moreover, the subgraph matching distance overrides the node matching distance in terms of relation with the edit distance while keeping a reasonable time complexity (Fig. 12).

4.3.2. Pearson correlation on distance matrices (1200×1200)

The histograms of Pearson correlations in Fig. 9 lead to the following conclusion; the distance values between *ED* and *SGMD* are highly correlated, a linear relation does exist between *ED* and *SGMD* (median value of the Pearson correlation = 0.858). This strengthens our decision to use a faster (and simpler) dissimilarity measure than edit distance in order to perform a graph classification.

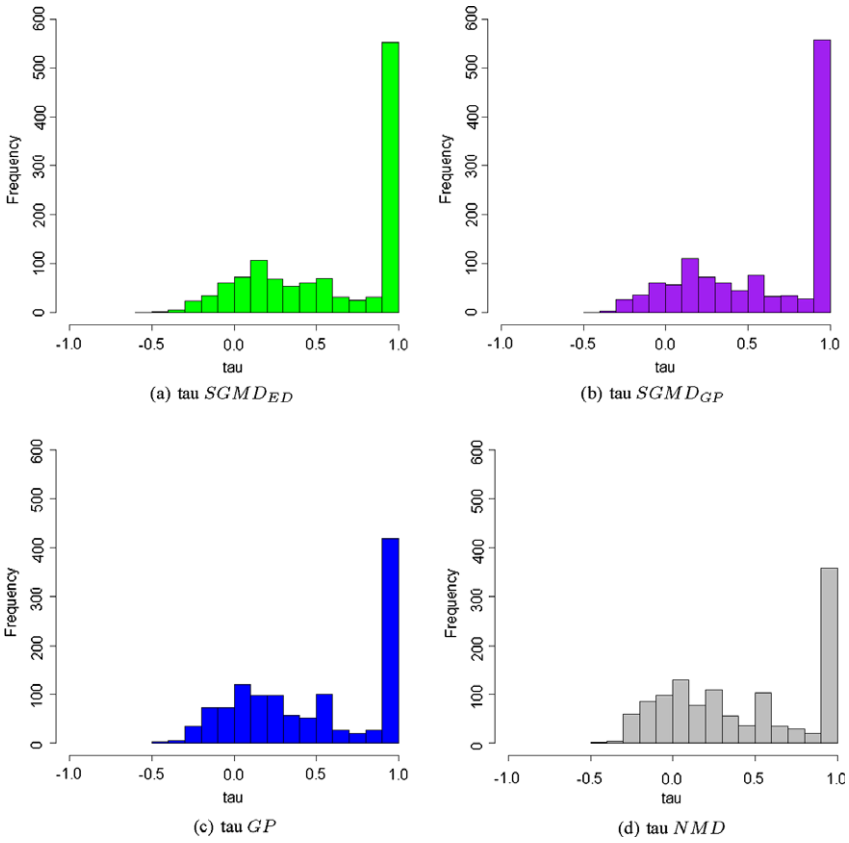


Fig. 8. Histogram of Kendall correlations, rank correlation to the responses to the *k*-NN queries.

4.3.3. PCA on correlation matrix

A correlation matrix is built from the mean values of the Kendall correlations (illustrated in Table 2). This matrix aims to compare the different graph distances between them. A matrix is not expressive enough, barely readable, in fact and so, a principal component analysis is performed to express its substantial sense on a 2D plot, called the correlation circle. Each eigenvalue corresponds to a factor, and each factor to a one dimension. A factor is a linear combination of the initial variables, and all the factors are un-correlated ($\tau = 0$). The eigenvalues and the corresponding factors are sorted by descending order of how much of the initial variability they represent (converted to %). In our situation stated in Fig. 11, the first two factors allow us to represent 71.40% of the initial variability of the data. This is a good result, and we can be confident with the reliability of the representation of the data. The correlation circle (on axes F1 and F2) shows a projection of the initial variables in the factors space. When variables are close to the circle edge, it means that the variable is well expressed by the two factors and so an interpretation is feasible. If variables are: close to each other, they are significantly positively correlated (τ close to 1); If they are orthogonal, they are not correlated (τ close to 0); If they are on the opposite side of the center, then they are significantly negatively correlated (τ close to -1). From these explicative guidelines, a first statement leads to conclude that the most highly correlated variables with *ED* are $SGMD_{ED}$ and $SGMD_{GP}$. Secondly, $SGMD_{ED}$ and $SGMD_{GP}$ are closely coupled. The simple reason being that both distances rely on the same principles to compute the matching, they only stand apart from each other by the cost function involved. Finally, NMD is the farthest from *ED*; this demonstrates the weakness of this method that does not take into account the edge information.

4.3.4. Pairwise distance scatter plot

These scatter plots give us a visual representation of the accuracy of the sub-optimal methods on the letter data. We plot for each pair of graphs its exact (horizontal axis) and approximate (vertical axis) distance value. Based on the scatter plots given in Fig. 10, we express the mean and the standard deviation of the difference between the approximate and exact distances. These measurements are given after a normalization by their maxima, respectively, hence the errors are comparable to each other. The residuals from the least squares method are an estimation of the fitness of the linear model between *ED* and other distances. Another indicator called ISE (Integral Square Errors) denotes the sum of the square errors between the linear model and the data, this estimator brings to light the amount of mistakes provoked by the linear approximation of the data. Lowest values are obtained by *SGMD* distances when high values for NMD tend to reveal the limits of a linear model for such sparse data. Note that all distances computed by the sub-optimal methods ($SGMD_{ED}$ and $SGMD_{GP}$) are equal to, or larger than, the exact distances.

4.4. Classification context

Back on track, we keep in mind that the final purpose is to perform a classification stage in order to measure our graph matching precision. Based on the data sets described in Section 4.1, a 1-NN rule is applied to obtain the number of correctly classified instances (CCI) and the corresponding classification rate. These results are reported in Table 3. Note that the classification stage using the graph edit distance could only be achieved on two datasets, the datasets *B* and *D* which are made up of relatively small graphs (i.e. Table 1). The computational complexity is exponential in the number of nodes of the involved graphs. Consequently, exact

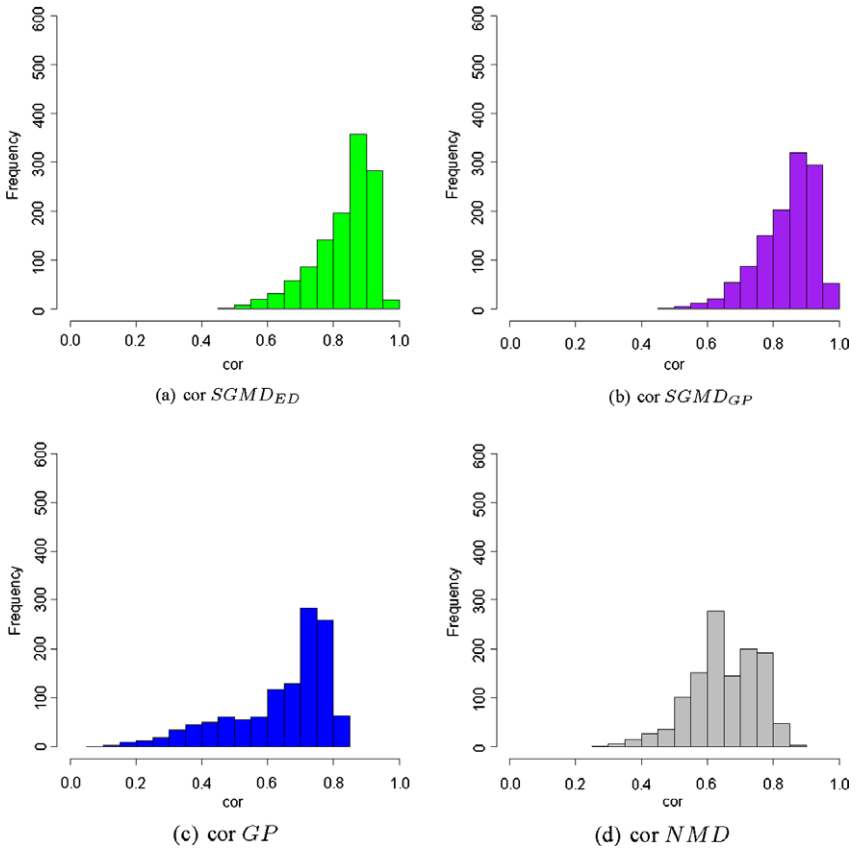


Fig. 9. Histogram of Pearson correlations, numeric correlations on distance matrices.

graph edit distance is feasible for graphs of rather small size only. Over the four databases, the subgraph matching distance outperforms the node matching distance. This observation finds a straightforward explanation, it confirms the interests of considering subgraphs which is to say local structured information when NMD only focuses on simple node sets. Another remark leads us to mention that *SGMD* provides better results than GP. Two reasons can explain this behavior. Firstly, in GP, probes are incorporated into a histogram comparison. This simplicity imposes the loss of relations between probes of two graphs and therefore, no matching can be expected from GP. On the contrary, *SGMD* aims to search the best subgraph to subgraph mapping according to a cost function. Hence, in GP, probes are treated independently whereas in *SGMD*, the mapping of two subgraphs is made considering all possibilities, meaning that the mapping of a given subgraph will impact the rest of the assignment problem. Secondly, we can underline the fact that in GP, the edge structures ($Probe2$) do not gather any node information whereas our approach does. In GP, nodes and edges are processed separately.

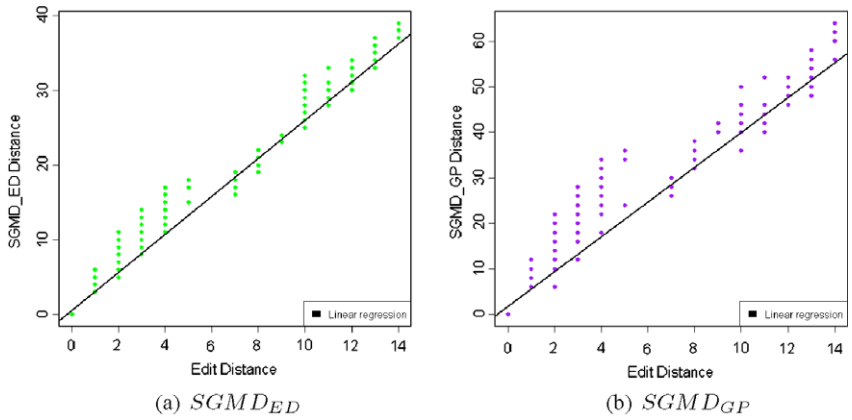
On one out of four data sets, the classification accuracy of a nearest-neighbor classifier improves when the exact edit distances are replaced by the sub-optimal ones returned by our algorithm. This can be explained by the fact that all distances computed by the sub-optimal methods are equal to, or larger than, the exact edit distances. A sub-optimal graph distance will not necessarily lead to a deterioration of the classification accuracy of a distance based classifier.

Finally, the edit distance gives a better result on Base D than the two others approaches. Objectively, this last remark denotes the loss information due to the approximation introduced by the concept of small subgraph of depth 1. In fact, the proposed algorithm considers only local, rather than global information. However, this little loss of accuracy (3%) should not discourage the use of the *SGMD* considering the important speed-up it provides while being quite accurate.

4.5. Time complexity analysis

The matching distance can be calculated in $O(n^3)$ time in the worst case. To calculate the matching distance between two attributed graphs G_1 and G_2, a minimum-weight subgraph matching between the two graphs has to be determined. This is equivalent to determining a minimum-weight maximal matching in the subgraph matching of G_1 and G_2. To achieve this, the method of Kuhn (1955) and Munkres (1957) can be used. This algorithm, also known as the Hungarian method, has a worst case complexity of $O(n^3)$, where n is the number of subgraphs in the larger one of the two graphs.

A way to compare the computational cost of the different types of distance was to undertake an empirical study. The Fig. 12 depicts a comparison of the runtime execution according to the kind of distances. This test was performed when calculating the distance matrices on 1200 graphs taken from Base D. A first comment aims at illustrating the high time consumption of the edit distance. This over-load discourages the use of this graph measure, even in case of low dimension graphs when its computation is feasible. On the other hand, the experiments demonstrated that the GP is four times faster than $SGMD_{ED}$. Firstly, *GP* and *SGMD* do not have the same purpose, GP is fast but do not express any mapping between vertices. Secondly, the time gap is low enough to do not reject *SGMD* as a suitable solution considering the significant accuracy gain it implies. *SGMD* is a good trade-off between time complexity and performance.



	$SGMD_{ED}$	$SGMD_{GP}$	NMD	GP
μ	0.046	0.054	0.2007	0.1381
σ	0.885	0.857	1.540	1.238
ISE	989	1246	4852	3232

Fig. 10. Scatter plots of the sub-optimal distances (y-axis) and the exact edit distances (x-axis). The mean and the standard deviation of the difference between the approximate and exact distances are reported in the table above.

Table 2
Kendall auto-correlation matrix (mean values).

	$SGMD_{ED}$	$SGMD_{GP}$	NMD	GP	ED
$SGMD_{ED}$	1.000	0.608	0.400	0.444	0.604
$SGMD_{GP}$	0.608	1.000	0.435	0.474	0.614
NMD	0.400	0.435	1.000	0.544	0.442
GP	0.444	0.474	0.544	1.000	0.494
ED	0.604	0.614	0.442	0.494	1.000

Table 3
Classification rate according to the graph distance in use.

Method	Base A	Base B	Base C	Base D
ED(%)	–	92.86	–	82.10
$SGMD_{ED}$(%)	88.54	94.64	99.54	80.86
$SGMD_{GP}$(%)	88.48	94.64	99.21	78.79
GP(%)	57.01	92.86	98.33	59.89
NMD(%)	29.49	89.28	88.75	36.96

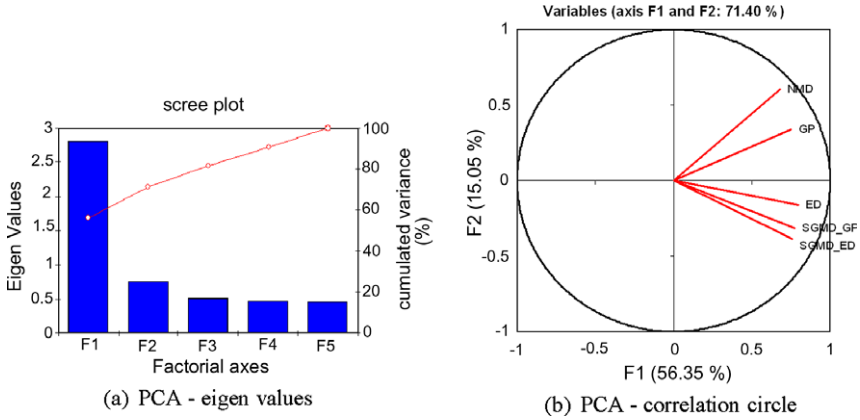


Fig. 11. Correlation matrix representation.

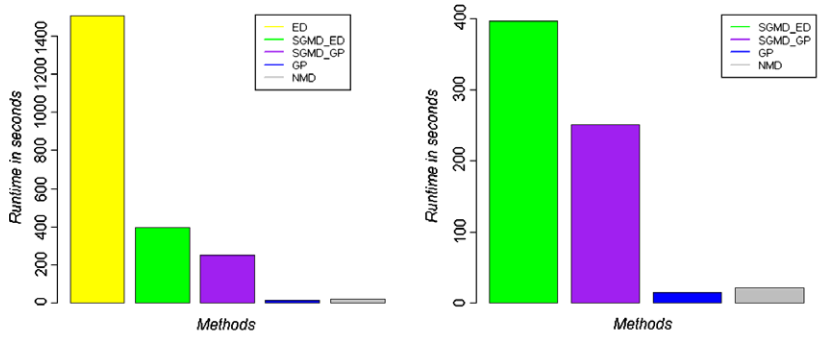


Fig. 12. Time complexity.

5. Conclusion

In the context of graph data classification, the complexity issues linked to graph dissimilarities measures make the process of classification for a large data set an important topic. After experimentally testing the correlation between subgraph matching distance and edit distance, it came up that the subgraph matching distance was of first interest, the best trade-off between accuracy and velocity. Furthermore, we obtain better results in terms of classification accuracy, than conventional graph measures on multi-class graph classification problems. Our contribution gives the proof for the use of a rapid and simple, yet sufficient graph distance which can be processed to scale up a k-NN classification step.

This paper pointed out the following fact, when the edit distance is not applicable, in the case of high dimension graphs, our approach is an alternative to process an accurate classification. Another strong advantage of our method relies on its deterministic computation.

In addition, graph distances often suffer from their shallow aspect. A black box concept where a single number expresses the link between two graphs. On the other hand, the proposed distance is induced by a graph matching algorithm, this observation implies a precious property. Our global distance is made up of local similarities that can be traced to find out precisely where the defects are. It provides a real explanation of how similar two graphs are.

We can conclude that in general the classification accuracy of the 1-NN classifier is not negatively affected by using the approximate rather than the exact edit distances. Our pseudo-metric for graph-based representation will not necessarily lead to a deterioration of the classification accuracy of a distance based classifier.

A future promising work is under investigation. It deals with the graph decomposition into subgraphs of bigger size. In such a way, a closer look will be given to the influence of matching subgraphs of length $1, 2, \ldots, |G|$. Our method makes such possibilities feasible.

References

Antoine Champin, P., Solnon, C., 2003. Measuring the similarity of labeled graphs. Case-based Reason. Res. Dev., 80–95.

Auwatanamongkol, S., 2007. Inexact graph matching using a genetic algorithm for image recognition. Pattern Recognition Lett. 28 (12), 1428–1437 <http://dx.doi.org/10.1016/j.patrec.2007.02.013>.

Bengoetxea, E., Larrañaga, P., Bloch, I., Perchant, A., Boeres, C., 2002. Inexact graph matching by means of estimation of distribution algorithms. Pattern Recognition 35 (12), 2867–2880.

Bonabeau, E., 2002. Graph multidimensional scaling with self-organizing maps. Inform. Sci. 143 (1–4), 159–180.

Borgwardt, K.M., Kriegel, H.-P., 2005. Shortest-path kernels on graphs. IEEE Internat. Conf. Data Mining, 74–81 <http://doi.ieeecomputersociety.org/10.1109/ICDM.2005.132>.

Bunke, H., Messmer, B.T., 1997. Recent advances in graph matching. Internat. J. Pattern Recognition Artif. Intell. (IJPRAI) 11 (1), 169–203.

Bunke, H., Shearer, K., 1998. A graph distance metric based on the maximal common subgraph. Pattern Recognition Lett. 19 (3–4), 255–259.

Christmas, M.W.J., Kittler, J., 1995. Structural matching in computer vision using probabilistic relaxation. IEEE Trans. Pattern Anal. Machine Intell. 17 (8), 749–764.

Ciaccia, P., Patella, M., Zezula, P., 1997. M-tree: An efficient access method for similarity search in metric spaces. In: Proc. 23rd Internat. Conf. on Very Large Data Bases (VLDB'97), pp. 426–435.

Conte, D., Foggia, P., Sansone, C., Vento, M., 2004. Thirty years of graph matching in pattern recognition. Internat. J. Pattern Recognition Artif. Intell. 18 (3), 265–298.

Coughlan, J.M., Ferreira, S.J., 2002. Finding deformable shapes using loopy belief propagation. Internat. Conf. Comput. Vision (-ECCV 2002), 453–468.

Cox, M.F., Cox, M.A.A., 2001. Multidimensional Scaling. Quantitative Applications in the Social Sciences. Chapman and Hall.

Cross, A.D.J., Hancock, E.R., 1996. Inexact graph matching with genetic search. Adv. Struct. Syntact. Pattern Recognition, 150–159.

Cross, A.D.J., Hancock, E.R., 1998. Graph matching with a dual-step em algorithm. IEEE Trans. Pattern Anal. Machine Intell. 20 (11), 1236–1253 <http://dx.doi.org/10.1109/34.730557>.

Cross, E.H.A.D.J., Wilson, R.C., 1997. Inexact graph matching using genetic search. Pattern Recognition 30 (6), 953–970.

Erd"os, P., Rényi, A., 1959. On Random Graphs, Publicationes Mathematicae Debrecen, vol. 6, pp. 290–297.

Ferrer, M., Valveny, E., Serratosa, F., 2006. Spectral median graphs applied to graphical symbol recognition. Progr. Pattern Recognition Image Anal. Appl., 774–783.

Finch, E.H.A.M., Wilson, R.C., 1997. Matching delaunay graphs. Pattern Recognition 30 (1), 123–140.

Ford, G.P., Zhang, J., 1992. Structural graph-matching approach to image understanding. Intell. Robots Comput. Vision X: Algorithms Techniq. 1607 (1), 559–569.

Gold, S., Rangarajan, A., 1996. A graduated assignment for graph matching. IEEE Trans. Pattern Anal. Machine Intell. 18 (4), 377–388.

Gold, S., Rangarajan, A., 1996. Graph matching by graduated assignment. IEEE Trans. Pattern Anal. Machine Intell., 239–244.

Gordon, A.D., 1999. Classification, second ed. Chapman and Hall.

Graph based knowledge discovery (subdue), SUBDUE, <http://ailab.wsu.edu/subdue/>.

Hidovic, D., Pelillo, M., 2004. Metrics for attributed graphs based on the maximal similarity common subgraph. Internat. J. Pattern Recognition Artif. Intell. 18 (3), 299–313.

Jiang, X., Münger, A., Bunke, H., 2000. Synthesis of representative graphical symbols by computing generalized median graph. In: Chhabra, A.K., Dori, D., (Eds.), Graphics Recognition: Recent Advances, pp. 183–192.

Kashima, H., Tsuboi, Y., 2004. Kernel-based discriminative learning algorithms for labeling sequences, trees, and graphs. In: Proc. 21st Internat. Conf. on Machine Learning.

Kaufman, L., Rousseeuw, P., 1990. Finding groups in data: An introduction to cluster analysis. Probability Mathematical Statistics, ISBN:10-0471878766.

Khotanzad, A., Hong, Y.H., 1990. Invariant image recognition by zernike moments. IEEE Trans. Pattern Anal. Mach. Intell. 12 (5), 489–497.

Kuhn, H.W., 1955. The Hungarian method for the assignment problem. Nav. Res. Logist. Quart. 2, 83–97.

Kuner, P., Ueberreiter, B., 1988. Pattern recognition by graph matching: Combinatorial versus continuous optimization. Internat. J. Pattern Recognition Artif. Intell. 2 (3), 527–542.

Lee, R., Liu, J., 2000. Tropical cyclone identification and tracking system using integrated neural oscillatory elastic graph matching and hybrid rbf network track mining techniques. IEEE Trans. on Neural Networks 11 (3), 680–689. doi:10.1109/72.846739.

Lee, Y.-L., Park, R.-H., 2002. A surface-based approach to 3-d object recognition using a mean field annealing neural network. Pattern Recognition 35 (2), 299–316.

Lladós, J., 1997. Combining graph matching and hough transform for hand-drawn graphical document analysis. Application to architectural drawings. Ph.D. thesis, Universitat Auto-noma de Barcelona and University, Paris, p. 8.

Lopresti, D., Wilfong, G., 2003. A fast technique for comparing graph representations with applications to performance evaluation. Internat. J. Doc. Anal. Recognition 6 (4), 219–229 <http://dx.doi.org/10.1007/s10032-003-0106-z>.

Luo, B., Hancock, E.R., 2000. Symbolic graph matching using the em algorithm and singular value decomposition. In: 15th Internat. Conf. on Pattern Recognition (ICPR'00), pp. 2141–2144.

Mahé, P., Ueda, N., Akutsu, T., Perret, J.-L., Vert, J.-P., 2004. Extensions of marginalized graph kernels. In:Proc. 21st Internat. Conf. on Machine Learning.

Mahé, P., Ueda, N., Akutsu, T., Perret, J.-L., Vert, J.-P., 2005. Graph kernels for molecular structure-activity relationship analysis with support vector machines. J. Chem. Inform. Model. 45 (4), 939–951.

Mehlhorn, K., 1984. Graph Algorithms and NP-completeness. Springer-Verlag New York, Inc., New York, NY, USA.

Messmer, B.T., Bunke, H., 1999. A decision tree approach to graph and subgraph isomorphism detection. Pattern Recognition 32 (12), 1979–1998.

Munkres, J., 1957. Algorithms for the assignment and transportation problems. J. Soc. Ind. Appl. Math. 5 (1), 32–38.

Myers, R., Wilson, R.C., Hancock, E.R., 2000. Bayesian graph edit distance. IEEE Trans. Pattern Anal. Mach. Intell. 22 (6), 628–635.

Neuhaus, M., Bunke, H., 2007. Automatic learning of cost functions for graph edit distance. Inform. Sci. 177 (1), 239–247.

Papadopoulos, A., Manolopoulos, Y., 1999. Structure-based similarity search with graph histograms. In: Proc. Structure-based Similarity Search with Graph Histograms, pp. 174–178.

Peter Kriegel, H., Schonauer, S., 2003. Similarity search in structured data. Data Warehousing Knowledge Discovery, 309–319.

Public IAM graph database repository, IAM. <http://iamwww.unibe.ch/fki/databases/iam-graphdatabase>.

Ralaivola, L., Swamidass, S.J., Saigo, H., Baldi, P., 2005. Graph kernels for chemical informatics. Neural Networks 18 (8), 1093–1110.

Robles-Kelly, A., Hancock, E.R., 2005. Graph edit distance from spectral seriation. IEEE Trans. Pattern Anal. Mach. Intell. 27 (3), 365–378.

Schenker, A., Last, M., Bunke, H., Kandel, A., 2004. Classification of web documents using graph matching. Internat. J. Pattern Recognition Artif. Intell. 18 (3), 475–496.

Seong, D.S., Choi, Y.K., Kim, H.S., Park, K.H., 1994. An algorithm for optimal isomorphism between two random graphs. Pattern Recognition Lett. 15 (4), 321–327.

Serrau, A., Marcialis, G.L., Bunke, H., Roli, F., 2005. An experimental comparison of fingerprint classification methods using graphs. Graph-based Representations Pattern Recognition, 281–290.

Shokoufandeh, A., Bretzner, L., Macrini, D., Demirci, M.F., Jönsson, C., Dickinson, S., 2006. The representation and matching of categorical shape. Comput. Vision Image Understanding 103 (2), 139–154 <http://dx.doi.org/10.1016/j.cviu.2006.05.001>.

Suard, F., Rakotomamonjy, A., Bensrhair, A., 2006. Object categorization using kernels combining graphs and histograms of gradients. Image Anal. Recognition, 23–34.

Suganthan, D.M.P.N., Teoh, E.K., 1995. Pattern recognition by graph matching using the potts mft neural networks. Pattern Recognition 28 (7), 997–1009.

Suganthan, D.M.P.N., Teoh, E.K., 1995. Pattern recognition by homomorphic graph matching using hopfield neural networks. Image Vision Comput. 13 (1), 45–60.

Valveny, E., Dosch, P., 2004. Performance evaluation of symbol recognition. Doc. Anal. Syst. VI, 354–365.

Vapnik, V., 1982. Estimation of Dependences Based on Empirical Data, Empirical Inference Science. Series: Information Science and Statistics.

Vidal, E., 1994. New formulation and improvements of the nearest-neighbour approximating and eliminating search algorithm (aesa). Pattern Recognition Lett. 15 (1), 1–7.

Wilson, R., Hancock, E., 1996. A bayesian compatibility model for graph matching. Pattern Recognition Lett. 17, 263–276.

Wong, A.K.C., You, M., 1985. Entropy and distance of random graphs with application to structural pattern recognition. IEEE Trans. Pattern Anal. Machine Intell. 7 (5), 599–609.