Pattern Anal Applic (2010) 13:113–129
DOI 10.1007/s10044-008-0141-y

THEORETICAL ADVANCES

A survey of graph edit distance

Xinbo Gao · Bing Xiao · Dacheng Tao · Xuelong Li

Received: 15 November 2007 / Accepted: 16 October 2008 / Published online: 13 January 2009

Abstract Inexact graph matching has been one of the significant research foci in the area of pattern analysis. As an important way to measure the similarity between pairwise graphs error-tolerantly, graph edit distance (GED) is the base of inexact graph matching. The research advance of GED is surveyed in order to provide a review of the existing literatures and offer some insights into the studies of GED. Since graphs may be attributed or non-attributed and the definition of costs for edit operations is various, the existing GED algorithms are categorized according to these two factors and described in detail. After these algorithms are analyzed and their limitations are identified, several promising directions for further research are proposed.

Keywords Inexact graph matching · Graph edit distance · Attributed graph · Non-attributed graph

X. Gao · B. Xiao
School of Electronic Engineering, Xidian University,
710071 Xi'an, People's Republic of China

X. Gao
e-mail: xbgao@mail.xidian.edu.cn

B. Xiao
e-mail: bingxue_8125@163.com

D. Tao
School of Computer Engineering,
Nanyang Technological University, 50 Nanyang Avenue,
Blk N4, Singapore 639798, Singapore
e-mail: dacheng.tao@gmail.com

X. Li (✉)
School of Computer Science and Information Systems,
Birkbeck College, University of London,
London WC1E 7HX, UK
e-mail: xuelong@dcs.bbk.ac.uk

1 Originality and contribution

Graph edit distance is an important way to measure the similarity between pairwise graphs error-tolerantly in inexact graph matching and has been widely applied to pattern analysis and recognition. However, there is scarcely any survey of GED algorithms up to now, and therefore this paper is novel to a certain extent. The contribution of this paper focuses on the following aspects:

On the basis of authors' effort for studying almost all GED algorithms, authors firstly expatiate on the idea of GED with illustrations in order to make the explanation visual and explicit;
Secondly, the development of GED algorithms and significant research findings at each stage, which should be comprehended by researchers in this area, are summarized and analyzed;
Thirdly, the most important part of this paper is providing a review of the existing literatures and offering some insights into the studies of GED. Since graphs may be attributed or non-attributed and the definition of costs for edit operations is various, existing GED algorithms are categorized according to these two factors and are described in detail. Advantages and disadvantages of these algorithms are analyzed and indicated by comparing them experimentally and theoretically;
Finally, a conclusion is given that several promising directions for further research are proposed in terms of limitations of the existing GED algorithms.

2 Introduction

In structural pattern recognition, graphs have invariability to rotation and translation of images, and in addition,

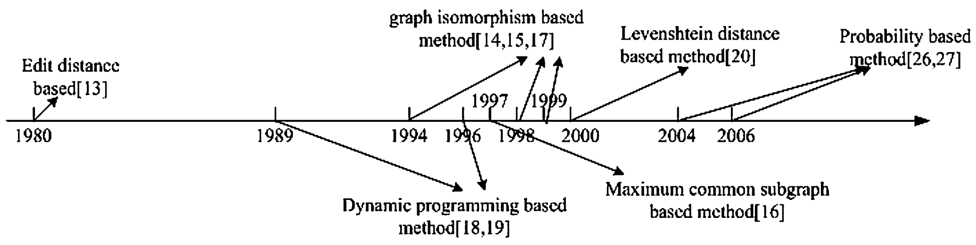
Edit distance based[13]
graph isomorphism based method[14,15,17]
Levenshtein distance based method[20]
Probability based method[26,27]
1980
1989
1994
1996
1997
1998
1999
2000
2004
2006
Dynamic programming based method[18,19]
Maximum common subgraph based method[16]

Fig. 1 The development of GED

transformation of an image into a 'mirror' image; thus they are used widely as the potent representation of objects. With the representation of graphs, pattern recognition becomes a problem of graph matching. The presence of noise means that the graph representations of identical real world objects may not match exactly. One common approach to this problem is to apply inexact graph matching [1–5], in which error correction is made part of the matching process. Inexact graph matching has been successfully applied to recognition of characters [6, 7], shape analysis [8, 9], image and video indexing [10–14] and image registration [15]. Since photos and their corresponding sketches are similar to each other geometrically and their difference mainly focuses on texture information, photos and sketches can be represented with graphs and then sketch-photo recognition [16] will be realized through inexact graph matching in the future. Central to this approach is the measurement of the similarity of pairwise graphs. This can be measured in many ways but one approach which of late has garnered particular interest because it is error-tolerant to noise and distortion is the graph edit distance (GED), defined as the cost of the least expensive sequence of edit operations that are needed to transform one graph into another.

In the development of GED which is demonstrated in Fig. 1, Sanfeliu and Fu [17] plays an important role, who first introduced edit distance into graph. It is computed by counting node and edge relabelings together with the number of node and edge deletions and insertions necessary to transform a graph into another. Extending their idea, Messmer and Bunke [18, 19] defined the subgraph edit distance by the minimum cost for all error-correcting subgraph isomorphisms, in which common subgraphs of different model graphs are represented only once and the limitation of inexact graph matching algorithms working on only two graphs once can be avoided. But the direct GED lacks some of the formal underpinning of string edit distance, so there is considerable current effort aimed at putting the underlying methodology on a rigorous footing. There have been some development for overcoming this drawback, for instance, the relationship between GED and the size of the maximum common subgraph (MCS) has been demonstrated [20], the uniqueness of the cost-function is commented [21], a probability distribution for local GED has been constructed, extending of string edit to trees and graphs, etc. GED has been computed for both attributed relational graphs and non-attributed graphs so far. Attributed graphs have the attribute of nodes, edges, or both nodes and edges according to which the GED is computed directly. Non-attributed graphs only include the information of connectivity structure; therefore they are usually converted into strings and edit distance is used to compare strings, the coded patterns of graphs. The edit distance between strings can be evaluated by dynamic programming [5], which has been extended to compare trees and graphs on a global level [22, 23]. Hancock et al. used Levenshtein distance, an important kind of edit distance, to evaluate the similarity of pairwise strings which are derived from graphs [24]. Whereas Levenshtein edit distance does not fully exploit the coherence or statistical dependencies existing in the local context, Wei made use of Markov random field to develop the Markov edit distance [25] in 2004. Recently, Marzal and Vidal [26] normalized the edit-distance so that it may be consistently applied across a range of objects in different size and this idea has been used to model the probability distribution for edit path between pairwise graphs [27]. The Hamming distance between two strings is a special case of the edit distance. Hancock measures the GED with Hamming distance between the structural units of graphs together with the size difference between graphs [28]. So, in the development of GED, the role of edit distance [5, 29] cannot be neglected and its advancement promotes the birth of new GED algorithms.

Although the research of GED has been developed flourishingly, GED algorithms are influenced considerably by cost functions which are related to edit operations. The GED between pairwise graphs changes with the change of cost functions and its validity is dependent on the rationality of cost functions definition. Researchers have defined the cost functions in various ways by far, and particularly, the definition of cost functions is cast into a probability framework mainly by Hancock and Bunke lately [30, 31]. The problem is not solved radically. Bunke [21] connects cost functions with multifold graph isomorphism in theory and then the necessity of cost function definition can be removed. But graph isomorphism is a kind of NP-complete problem and has to work together with constraints and heuristics in practice. So, efforts are still needed to develop new GED algorithms having reasonable cost functions or independent of defining cost functions, such as the algorithms in [32, 33].

The aim of this paper is to provide a survey of current development of GED which is related to computing the dissimilarity of graphs in error correcting graph matching. This paper is organized as follows: after some concepts and basic algorithms are given in Sect. 3, the existing GED algorithms are categorized and described in detail in Sect. 4, which compares existing algorithms to show their advantages and disadvantages. In Sect. 5, a summary is presented and some important problems of GED deserving further research are proposed.

3 Basic concepts and algorithms

Many concepts of graph theory and basic algorithms of search and learning strategies are regarded as the foundation of existing GED algorithms. In order to describe and analyze GED algorithms thoroughly, these concepts have to be expounded in advance.

3.1 Concepts related to graph theory

The subject investigated by GED is the graph representation of objects, and, judging from current research results, graph theory is the basis of GED research; therefore, introduction of some concepts related to graph theory is necessary, such as the definitions of the graph with or without attributes, the directed acyclic graph, the common subgraph, the common supergraph, the maximum weight clique, the isomorphism of graphs, the transitive closure, the Fiedler vector and the super clique.

3.1.1 Definitions of the graph and the attributed graph

A *graph* is denoted by $G = (V, E)$, where

$V = \{1, 2, \ldots, M\}$

is the set of vertices (nodes), E is the edge set ($E \subseteqq V \times V$). If nodes in a graph have attributes, the graph is an *attributed graph* denoted by $G = (V, E, l)$, where l is a labeling function

$l : V \rightarrow L_N.$

If both nodes and edges in a graph have attributes, the graph is an *attributed graph* denoted by $G = (V, E, \alpha, \beta)$, where

$\alpha : V \rightarrow L_N$ and $\beta : E \rightarrow L_E$

are node and edge labeling functions. L_N and L_E are restricted to labels consisting of fixed-size tuples, that is, $L_N = R^p$, $L_E = R^q$, $p, q \in N \cup \{0\}$.

3.1.2 Definitions of the directed graph and directed acyclic graph

Given a graph $G = (V, E)$, if E is a set of ordered pairs of vertices, G is a directed graph and edges in E are called directed edges. If there is no non-empty directed path that starts and ends on v for any vertex v in V, G is a directed acyclic graph.

3.1.3 Definition of the subgraph and supergraph

Let $G = (V, E, \alpha, \beta)$ and $G' = (V', E', \alpha', \beta')$ be two graphs; G' is a *subgraph* of G, G is a *supergraph* of G', $G' \subseteqq G$, if

- $V' \subseteqq V$,
- $E' \subseteqq E$,
- $\alpha'(x) = \alpha(x)$ for all $x \in V'$,
- $\beta'((x, y)) = \beta((x, y))$ for all $(x, y) \in E'$.

For non-attributed graphs, only the first two conditions are needed.

3.1.4 Definition of the graph isomorphism

Let $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ be two graphs. A *graph isomorphism* between G_1 and G_2 is a bijective mapping $f : V_1 \rightarrow V_2$ such that

- $\alpha_1(x) = \alpha_2(f(x)), \forall x \in V_1$,
- $\beta_1((x, y)) = \beta_2((f(x), f(y))), \forall (x, y) \in E_1$.

For non-attributed graphs $G_1' = (V_1', E_1')$ and $G_2' = (V_2', E_2')$, a bijective mapping $f : V_1' \rightarrow V_2'$ such that $(u, v) \in E_1' \Leftrightarrow (f(u), f(v)) \in E_2', \forall u, v \in V_1'$ is a *graph isomorphism* between these two graphs.

If $V_1 = V_2 = \phi$, then f is called the empty graph isomorphism.

3.1.5 Definitions of the common subgraph and the maximum common subgraph

Let G_1 and G_2 be two graphs and $G_1' \subseteqq G_1$, $G_2' \subseteqq G_2$. If there exists a graph isomorphism between G_1' and G_2', then both G_1' and G_2' are called a *common subgraph* of G_1 and G_2. If there exists no other common subgraph of G_1 and G_2 that has more nodes than G_1' and G_2', G_1' and G_2' are called a MCS of G_1 and G_2.

3.1.6 Definitions of the common supergraph and the minimum common supergraph

A graph $\breve{G}$ is a *common supergraph* of two graphs G_1 and G_2 if there exist graphs $\breve{G}_1 \subseteq \breve{G}$ and $\breve{G}_2 \subseteq \breve{G}$ such that there exists a graph isomorphism between $\breve{G}_1$ and G_1, and a graph isomorphism between $\breve{G}_2$ and G_2. It is a *minimum common supergraph* if there is no other common supergraph of G_1 and G_2 smaller than $\breve{G}$.

3.1.7 *Definition of the Fiedler vector*

If the degree matrix and adjacency matrix of a graph are diagonal matrix

$D = \mathrm{diag}(\deg(1), \deg(2), \ldots, \deg(M)),$

where M is the last node in the graph, and the symmetric matrix A, respectively, the Laplacian matrix is the matrix $L = D - A$. The eigenvector corresponding to the second smallest eigenvalue of the graph Laplacian is referred to as the *Fiedler vector*.

3.1.8 *Definitions of the clique and the maximum weight clique*

A clique in a graph is a set of nodes which are adjacent to each other, for example, in Fig. 2, node 3, node 4 and node 5 form a clique in the graph. Weight clique is the extension of clique to weighted graphs. Maximum weight clique is the clique with the largest weight.

3.1.9 *Definition of the super clique*

Given a graph $G = (V, E)$, the super clique (or neighborhood) of the node $i \in V$ consists of its center node i together with its immediate neighbors connected by edges in the graph.

3.1.10 *Transitive closure*

The transitive closure of a directed graph $G = (V, E)$ is a graph $G+ = (V, E+)$ such that for all v and w in V, $(v, w) \in E+$ if and only if there is a non-null path from v to w in G.

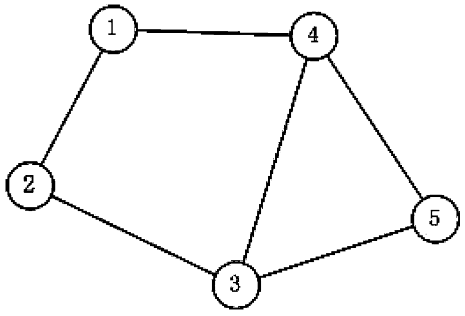

Fig. 2 An example of clique in the graph

3.2 Basic algorithms used in the existing GED algorithms

The definition of cost functions is key issue of GED algorithms and self-organizing map (SOM) can be used to learn cost functions automatically to some extent [34]. GED is defined as the cost of least expensive edit sequences; thus search strategy for shortest path is closely related to GED algorithms. Dijkstra's algorithm is the most popular shortest path algorithm and is applied by Robles-Kelly [35]. In addition, expectation maximum (EM) algorithm is applied to parameter optimization [30]. So, EM algorithm, Dijkstra's algorithm and SOM are presented here before GED algorithms are given.

3.2.1 *EM algorithm*

Expectation maximum algorithm [36] is one of the main approaches for estimating the parameters of a Gaussian mixture model (GMM). There exist two sample spaces X and Y, and a many-one mapping from X to Y. Data Y derived in space Y is observed directly, and corresponding data x in X is not observed directly, but only indirectly through Y. Data x and y are referred to as the complete data and incomplete data, respectively. Given a set of N random vectors

$Z = \{z_1, z_2, \ldots, z_N\}$

in which each random vector is drawn from an independent and identically distributed mixture model, the likelihood of the observed samples (conditional probability) is defined as the joint density

$P(Z|\theta) = \Pi_{i=1}^{N} p(z_i|\theta).$

Z is the complete data and z_i is the incomplete data, and the aim of EM algorithm is to determine the parameter θ that maximizes $P(Z|\theta)$ given an observed Z.

The EM algorithm is an iterative maximum likelihood (ML) estimation algorithm. Each iteration of EM algorithm involves two steps: expectation step (E-step) and maximization step (M-step). In E-step, the updated posterior probability is computed with the prior probability, and in M-step, according to the posterior probability transferred from E-step, conditional probability is maximized to obtain the updated prior probability and the parameters corresponding to the updated prior probability are transferred to E-step.

3.2.2 *Dijkstra's algorithm*

Dijkstra's algorithm [37] is developed by Dijkstra. It is a greedy algorithm that solves the single-source shortest path problem for a directed graph with non-negative edge weights and it can be extended to undirected graph. Given a weighted directed graph $G = (V, E)$, each of edges in E is given a weight value, the cost of moving directly along this edge. The cost of a path between two vertices is the sum of costs of the edges in that path. Given a pair of vertices s and t in V, the algorithm finds the shortest path from s to t.

Let S be the set of nodes visited along the shortest path from s to t. The adjacency matrix of G is the weight value matrix C. The element $d(i)$ is the cost of path from s to $v_i \in V$, and $d(s) = 0$. The algorithm can be described as below:

- Initialization: $S = \phi$, and $d(i)$ is the weight value of the edge (s, v_i);
- If $d(j) = \min\{d(i)|v_i \in V - S.\}$ is true, $S = S \cup \{v_j\}$;
- For every node $v_k \in V - S$, if $d(j) + C(j, k) < d(k)$ is true, $d(k)$ is updated, that is, $d(k) = d(j) + C(j, k)$;
- The last two steps are repeated until vertex t is visited and $d(t)$ is unchanged such that the shortest path from s to t is achieved.

3.2.3 SOM

Self-organizing map [38–40] is an unsupervised artificial neural network and maps the training samples into low-dimensional space with the topological properties of the input space unchanged. In SOM, neighboring neurons compete in their activities by means of mutual lateral interactions, and develop adaptively into specific detectors of different signal patterns, so it is unsupervised, self-organizing and competitive.

The SOM network consists of two layers: one is input layer and another is competitive layer. As shown in Fig. 3, hollow nodes denote neurons in input layer and solid nodes are competitive neurons. Each input is connected to all neurons in the competitive layer and every neuron in the competitive layer is connected to the neurons in its neighborhood. For each neuron j, its position is described as neural weight W_j, and for neurons in the competitive layer, the grid connections are regarded as their neighborhood relation. The training process is as below:

- The neurons of the input layer selectively feed input elements into the competitive layer;
- When an input element D is mapped onto the competitive layer, the neurons in competitive layer compete for the input element's position to represent the input element well. The closest neuron c, the winner neuron, is chosen in terms of the distance metric d_v which is the distance of neural weights in the vector space;
- Neighborhood $N^c = \{c, n_1^c, n_2^c, \ldots, n_M^c\}$, where M is the number of neighbors, of the winner neuron c is determined with the distance d_n between neurons which is defined by means of the neighborhood relations. The neuron c and its neighbors in N^c are drawn closer to the input element and weights of the whole neighborhood are moved in the same direction, similar items tend to excite adjacent neurons. The strength of the adaptation for a competitive neuron is decided by a non-increasing activation function $\alpha(t)$, and the weight of the neuron j in competitive layer is adapted according to the following formula:

$$W_j(t+1) = \begin{cases} W_j(t) + \alpha(t)(D(t) - W_j(t)), & j \in N^c \\ W_j(t), & j \notin N^c \end{cases}. \tag{1}$$

- The last two steps are repeated until a terminal condition is achieved.

4 Graph edit distance

A graph can be transformed to another one by a finite sequence of graph edit operations which may be defined differently in various algorithms, and GED is defined by the least-cost edit operation sequence. In the following, an example is used to illustrate the definition of GED. For model graph shown in Fig. 4 and data graph shown in

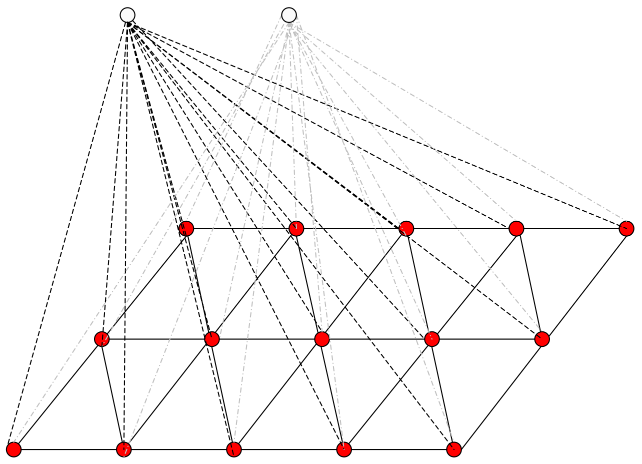

Fig. 3 The structure of SOM

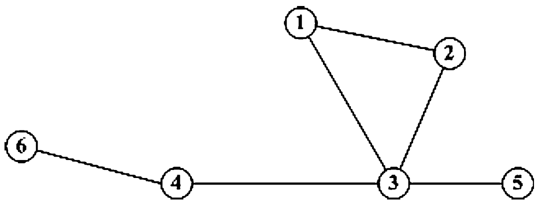

Fig. 4 The model graph

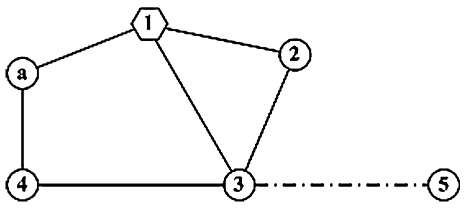

Fig. 5 The data graph

Fig. 5, the task is transforming data graph into model graph. All edit operations are performed on the data graph. One of the edit operation sequences includes node insertion and edge insertion (node 6 and its relative edge), node deletion and edge deletion (node a and its relative edges), node substitution (node 1) and edge substitution (the edge relative to node 5 and node 3). A cost function is defined for each operation and the cost for this edit operation sequence is sum of costs for all operations in the sequence. The sequence of edit operations and its cost needed for transforming a data graph into a model graph is not unique, but the least cost is exclusive. Then edit operation sequence with the least cost is requested and its cost is the GED between these two graphs. It is obvious that how to determine the similarity of components in graphs and define costs of edit operations are the key issues.

A graph may be an attributed relational graph with attributes of nodes, edges, or both nodes and edges, according to which the GED is computed directly. On the other hand, for a structural graph only having the information of connectivity structure, graphs are usually converted into strings according to nodes, edges or the connectivity, and the GED is computed based on the edit distance methods concerning strings. GED algorithms, whose ideas are given in brief, are classified from these two aspects. Algorithms for different kinds of graphs are not comparable. The distances obtained with algorithms of the same kinds are compared in the ability of clustering and classifying images, and accordingly their superiorities and flaws can be concluded, which may be in favor of our further research.

4.1 GED for attributed graphs

Graph edit distance for attributed graphs is computed directly according to the attributes which are various in different algorithms. In the SOM based method [34], probability based approach [30], convolution graph kernel based method [41] and subgraph and supergraph based method [42], attributes are of both nodes and edges, whereas the attribute is of nodes in binary linear programming (BLP) based method [43].

4.1.1 SOM based algorithm

In the existing algorithms for GED, the automatic inference of the cost for edit operations remains an open problem. To this end, the SOM based algorithm [34] is developed, in which the attributed graphs $G = (V, E, \alpha, \beta)$ are the objects to be processed. Every node and edge labels are m-dimensional and n-dimensional vectors, respectively. The space of the node and edge labels in a population of graphs is mapped into a regular grid which is an untrained SOM network, and the grid will be deformed after being trained. One type of edit operation is described by a SOM. The actual edit costs are derived from a distance measure for labels that is defined with the distribution encoded in the SOM. The encoding of node substitution is described as below:

- The m-dimensional node label space is reduced to a regular grid by being sampled at equidistant positions. Each vertex of the grid is connected to its nearest neighbor along the dimensions so as to obtain a representation of the full space. The regular grid is SOM neural network;
- Grid vertices and connections correspond to competitive neurons and neighborhood relations in the SOM;
- When it is being trained, the SOM corresponds to a deformed grid. A label vector at a vertex position of the regular grid is mapped directly onto the same vertex in the deformed map. Any vector in the original space that is not at a vertex can be mapped into the deformed space by a simple linear interpolation of its adjacent grid points.
- The cost of substituting node v_2 for node v_1 is defined with d_v, that is,

  $$c(v_1 \rightarrow v_2) = \beta^n_{\text{sub}} d_v(v'_1, v'_2),$$

  where β^n_{sub} is the weighting factor, and the vector v_i' is v_i in the deformed space. The weighting factor compensate for the dependency of the initial distance between vertexes.

For other edit operations, the SOM networks are constructed in an analogous way. The vertex distribution of each SOM will be changed iteratively in the learning procedure, which results in different costs. The object is to derive the cost functions resulting in small intraclass and large interclass distances; therefore activation function $\alpha(t)$ is defined such that the value of the function decreases when the distance between neurons increases.

The experiments demonstrating the performance of SOM based method are performed on the graph samples consisting of ten distorted copies for each of the three letter-classes A, E, and X. The instances of letter A being sheared are illustrated in Fig. 6. Shearing factor α is used to

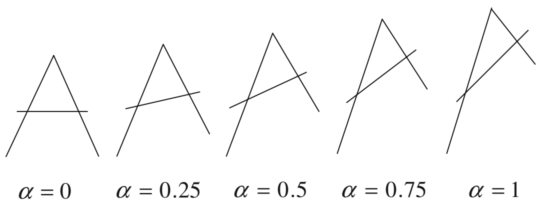

Fig. 6 Sheared letter A with shearing factor α in [34]

indicate the degree of letter distortion. For every shearing factor, the best average index is computed and shown in Fig. 7. The average index, which is normalized to the unit interval [0, 1], is defined by the average value of eight validation indices to evaluate the performance of clustering quantitatively. The Smaller values, the better clustering. Eight validation indices are the Davies–Bouldin index [44], the Dunn index [45], the C-index [46], the Goodman–Kruskal index [47], the Calinski–Harabasz index [48], Rand statistics [49], the Jaccard coefficient [50], and the Fowlkes–Mallows index [51]. In Fig. 7, the average indices corresponding to SOM learning are smaller than those of the Euclidean model under every shearing factor and the superiority of SOM over the Euclidean model is increasingly obvious with the shearing factor increasing, so edit costs derived through SOM learning make the difference between intraclass and interclass distances greater than that derived through Euclidean model, which illustrates that the SOM performs better than Euclidean model for clustering.

In this method, GED is computed based the metric d_v; therefore, the obtained GED is a metric. For a certain application, some areas of the label space are of great relevancy, while other areas are irrelevant. Other existing cost functions treat every part of label space equally, which can be overcome by the SOM based method learning the relevant areas of the label space from graph sample set.

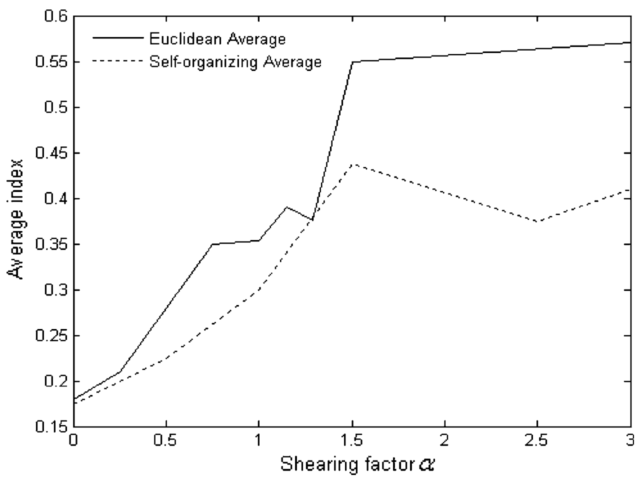

Fig. 7 Comparison of average index on sheared line drawing sample [34]

4.1.2 *Probability based algorithm*

Similar to the cost learning system based on the frequency estimation of edit operations for the string matching [52], Neuhaus proposed a probability based algorithm [30] to compute GED. In this algorithm, if the GED of graphs G_1 and G_2 is to be computed by transferring G_1 into G_2, two independent empty graphs EG_1 and EG_2 are constructed for G_1 and G_2, respectively, by a stochastic generation process. The sequence of node and edge insertion is applied to either both or only one of the two constructed graphs EG_1 and EG_2, which can be interpreted as an edit operation sequence transforming G_1 into G_2 and whose effects on G_1 are presented in Table 1. Edit costs are derived from the distribution estimation of edit operations. Each type of edit operations, regarded as a random event, is modeled with a GMM and the mixtures are weighted to form the probability distribution of edit events. Initialization starts with the empiric mean and covariance matrices of a single component, and new components are added sequentially if the mixture density appears to converge in the training process. Training pairs of graphs required to be similar are extracted and the EM algorithm is employed to find a locally optimized parameter set in terms of the likelihood of the edit events occurring between the pairwise training graphs. If a probability distribution of edit events sequence

$$(e_1, e_2, \ldots, e_l)$$

is given, the probability of two graphs $p(G_1, G_2)$ is defined as

$$p(G_1, G_2) = \int_{(e_1, e_2, \ldots, e_l) \in \psi(G_1, G_2)} \mathrm{d}p(e_1, e_2, \ldots, e_l) \tag{2}$$

where $\psi(G_1, G_2)$ denotes the set of all edit operations transferring G_1 to G_2. Finally, the distance between these two graphs is obtained by setting

$$d(G_1, G_2) = -\log(p(G_1, G_2)). \tag{3}$$

This algorithm is compared with the SOM based algorithm [34]. Three letter classes Z, A, and V are chosen, and 90 graphs (30 samples per class) are constructed to produce five sample sets with different values of the distortion parameter, 0.1, 0.5, 0.8, 1.0, and 1.2. Examples of these graphs are shown in Fig. 8. The average index consisting of the Calinski–Harabasz index [48], Davies–Bouldin index [44],

Table 1 Effects of edit operations on original graph

Edit operations		Effects on G_1
EG_1	EG_2	
Node/edge insertion	/	Node/edge insertion
/	Node/edge insertion	Node/edge deletion
Node/edge insertion	Node/edge insertion	Node/edge substitution

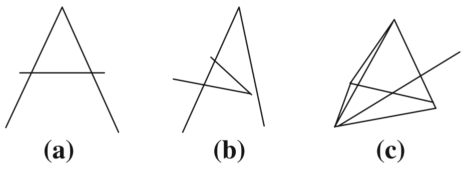

Fig. 8 Line drawing example. **a** Original drawing of letter *A*; **b** distorted instance of the same letter with distortion parameter 0.5 and **c** distortion parameter 1.0 [30]

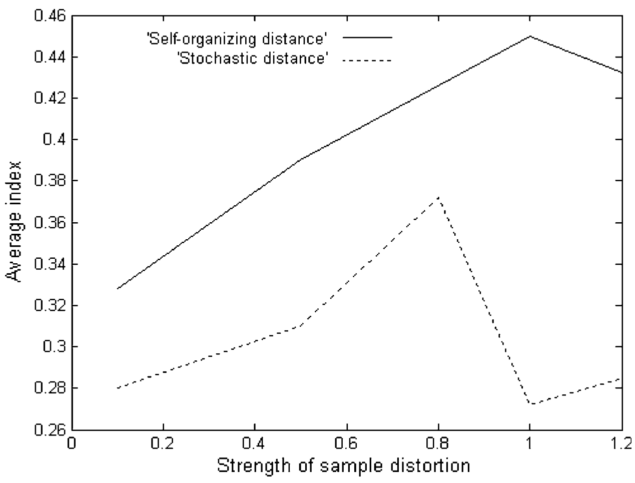

Fig. 9 Comparison of performance with increasing strength of distortion in [30]

Goodman–Kruskal index [47], and *C*-index [46] is computed for every sample set, the result of which is shown in Fig. 9. As mentioned ahead, smaller average indices correspond to better clustering and this method corresponds to smaller average index in every sample set and for every distortion level; thus, it is confirmed that this method clearly leads to better clustering results than SOM based algorithm, and the best average index value is obtained for the second-strongest distortion, although the matching task becomes harder and harder with increasing distortion strength.

Although SOM neural network can derive edit costs automatically and distinct the relevant areas of the label space, edit costs derived according to probability distribution of edit operations are more effective for clustering distorted letters. The advantage of this method is that it is able to cope with large samples of graphs and strong distortions between samples of the same class. It can be found that the key of this algorithm is the probability distribution of edit events.

4.1.3 Method based on convolution graph kernel

Kernel method is a new class of algorithms for pattern analysis based on statistical learning. When kernel functions are used to evaluate graph similarity, the graph matching problem can be formulated in an implicitly existing vector space, and then statistical methods for pattern analysis can be applied. In the algorithm based on convolution graph kernel [41], a novel graph kernel function is proposed to compute the GED so as to avoid the lack of mathematical structure in the space of graphs.

For graphs $G = (V, E, \mu, \nu)$ and $G' = (V', E', \mu', \nu')$, the cost of node substitution $u \to u'$ replacing node $u \in V$ by node $u' \in V'$ is given by the radial basis function:

$$K_{\mathrm{sim}}(u,u') = \exp\left(-\|\mu(u) - \mu'(u')\|^2 \Big/ 2\sigma^2\right). \tag{4}$$

The same function with different parameter σ is also used to evaluate the similarity of edge labels. These radial basis functions favor edit paths containing more substitutions, fewer insertions and fewer deletions. Hence, substitutions are modeled explicitly, while insertions and deletions implicitly.

The set of edit decompositions (sequence consisting of all nodes and edges in graph) of G is denoted by $R^{-1}(G)$ and a function evaluating whether two edit decompositions are equivalent to a valid edit path is denoted by K_{val}. For $x \in R^{-1}(G)$ and $x' \in R^{-1}(G')$, the function K_{val} is defined as follows:

$$K_{\mathrm{val}}(x,x') = \begin{cases} 1, & \text{if edit path } x \text{ and } x' \text{ is valid} \\ 0, & \text{otherwise} \end{cases}. \tag{5}$$

With these notations, the proposed edit kernel function on graphs can finally be written as:

$$k(G,G') = \sum_{\substack{x \in R^{-1}(g) \\ x' \in R^{-1}(g')}} K_{\mathrm{val}}(x,x') \prod_i K_{\mathrm{sim}}((x)_i,(x')_i), \tag{6}$$

where the index i indicates all nodes and edges present in the edit decomposition. In the computation of the kernel value $k(G, G')$, only valid edit paths are considered with the help of function K_{val}.

On the one hand, convolution edit kernel based GED and support vector machines (EK-SVM) are brought together, whose classification performance is compared with that of the traditional edit distance together with the *k*-nearest neighbor classifier [53] (ED-*k*NN). This experiment is conducted on the 15 letters that can be drawn with straight lines only, such as *A*, *E*, *F*, etc. The distorted letter graphs are split into a training set of 150 graphs, a validation set of 150 graphs, and a test set of 750 graphs. The experimental results are shown in Fig. 10, the accuracy of these two methods are heightened gradually with the increase of running time and convolution edit kernel based GED method has higher rate of classification than traditional edit distance under the same running time.

On the other hand, this method is compared with kernel functions derived directly from edit distance [54] (ED-

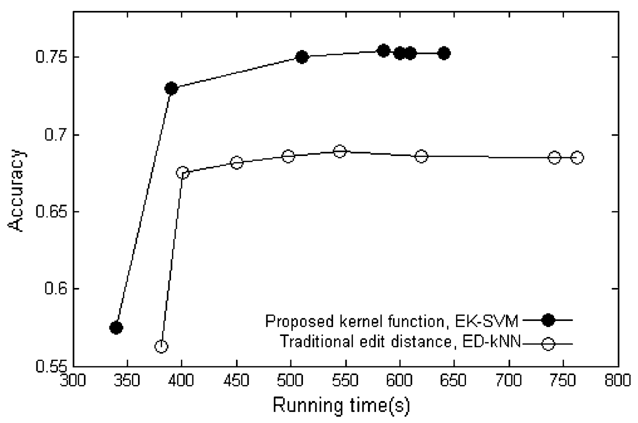

Fig. 10 Running time and accuracy of the proposed kernel function and edit distance in [41]

Table 2 Accuracy of two edit distance methods (ED), a random walk kernel (RW), and the proposed edit kernel (EK) in [41] (%)

	Letter dataset	Image dataset
ED-*k*NN	69.33	48.15
ED-SVM	73.2	59.26
RW-SVM	75.2	33.33
EK-SVM	75.2	68.52

SVM), and random walks in graphs [55] (RW-SVM), respectively. The Letters dataset used in the last experiment and the image dataset which is split into a training set, a validation set, and a test set, each of size 54, is used in this experiment. The images are assigned to one of the classes snowy, countryside, city, people, and streets and they are described in [56] in detail. The classification accuracy of four methods mentioned above is shown in Table 2. The EK-SVM method outperforms all other methods on the second dataset and achieves significantly higher classification accuracy than the traditional edit distance method. RW-SVM performs as well as EK-SVM on the first dataset, but significantly worse than all other methods on the second dataset. Convolution edit kernel based performs best of other methods.

In a word, the convolution edit kernel based GED together with SVM outperforms not only the cooperation of traditional edit distance and *k*NN, but also other kernel functions combining with SVM in classification. Unlike the traditional edit distance, this kernel function makes good use of statistical learning theory in the inner product rather than the graph space directly. The convolution edit kernel is defined by decomposing pairs of graphs into edit path, so it is more closely related to GED than other kernel functions.

4.1.4 Method based on binary linear programming

The BLP based algorithm [43] is for graphs with vertex attributes only and a framework for computing GED on the set of graphs is introduced. Every attributed graph in the set is treated as a subgraph of a larger graph referred to as edit grid, and edit operations of converting a graph into another one are equivalent to the state altering of the edit grid, from which GED can be derived. With the help of graph adjacency matrix, it can be treated as a problem of the BLP.

If the GED between graph $G_0 = (V_0, E_0, l_0)$ and graph $G_1 = (V_1, E_1, l_1)$ is to be computed, the graph G_0 is firstly embedded in a labeled complete graph

$$G_\Omega = (\Omega, \Omega \times \Omega, l_\Omega),$$

such that

- Graph G_0 is a subgraph of graph G_Ω,
- Label $l_\Omega(\omega_i) = \phi$ for all nodes $\omega_i \in \Omega - V_0$,
- Label $l_\Omega(\omega_i, \omega_j) = 0$ for all edges

$$(\omega_i, \omega_j) \in (\Omega \times \Omega) - E_0$$

The $G_\Omega = (\Omega, \Omega \times \Omega, l_\Omega)$ is the edit grid and its state vector is denoted by $\eta \in (\Sigma \cup \phi)^N \times \{0,1\}^{(N^2-N)/2}$, where Σ is the label alphabet of nodes in the graph G_0 and N is the number of nodes in the edit grid.

Then, a sequence of edits used to convert graph G_0 into the graph G_1 can be specified by the sequence of edit grid state vectors $\{\eta_k\}_{k=0}^M$. The GED between G_0 and G_1 is the minimum cost of state transition of edit grid, that is,

$$\begin{aligned} d_c(G_0, G_1) &= \min_{\{\eta_k\}_{k=1}^M | \eta_M \in \Gamma_1} \sum_{k=1}^{M} c(\eta_{k-1}, \eta_{k-1}) \\ &= \min_{\{\eta_k\}_{k=1}^M | \eta_M \in \Gamma_1} \sum_{k=1}^{M} \sum_{i=1}^{I} c\left(\eta^i_{k-1}, \eta^i_k\right), \\ &= \min_{\pi \in \Pi} \sum_{i=1}^{I} c\left(\eta^i_0, \eta^{\pi^i}_1\right) \end{aligned} \tag{7}$$

where $I = N + (N^2 - N)/2$, Γ_1 is the set of state vectors corresponding to all isomorphisms of G_1 on the edit grid, and Π is the set of all permutation mappings for isomorphisms of the edit grid. Permutation maps the element i of a set to other element π^i of the same set. By introducing the Kronecker delta function $\delta : \Re^2 \to \{0, 1\}$, formula (7) is equalized to formula (8):

$$\begin{aligned} d_c(G_0, G_1) = \min_{\pi \in \Pi} \sum_{i=1}^{N} c\left(\eta^i_0, \eta^j_1\right) \delta(\pi^i, j) \\ + c(0,1) \sum_{i=N+1}^{I} \left(1 - \delta\left(\eta^i_0, \eta^{\pi^i}_1\right)\right) \end{aligned} \tag{8}$$

Finally, the edit grid state vector η_k is represented with the adjacency matrix A_k whose elements correspond to edge labels in the state vector and rows (columns) are indexed with node labels, that is

$$A_k^{ij} = \eta_k^{iN+j-\frac{i^2+i}{2}} \quad \text{and} \quad (A_k^i) = \eta_k^i \quad \text{where} \quad 1 \le i, \quad j \le N$$

Formula (8) is converted into formula (9)

$$\begin{aligned} d_c(G_0, G_1) = \min_{P,S,T \in \{0,1\}^{N\times N}} & \sum_{i=1}^{N}\sum_{j=1}^{N} c(l(A_0^i),\ l(A_1^j))P^{ij} \\ & + \frac{1}{2}c(0,\ 1)(S+T)^{ij} \\ \text{s.t.} \quad & (A_0P - PA_1 + S - T)^{ij} = 0 \quad \forall\, i, j, \\ \text{and} \quad & \sum_i P^{ik} = \sum_j P^{kj} = 1 \quad \forall\, k, \end{aligned} \tag{9}$$

where $P^{ij} = \delta(\pi^i, j)$, $i, j \in [1, N]$ is a permutation matrix, S and T are the introduced matrices for formula conversion. Formula (9) is a BLP, and the solved optimal permutation matrix P^* can be used to determine the optimal edit operations.

This method is tested on 135 similar molecules which have only 18 or fewer atoms in the Klotho Biochemical Compounds Declarative Database [57]. Ideally, pairwise distances of all these molecules are the same. Two MCS-based distance metrics are used as references. The GED computed with this method is more concentrated than that of MCS-based distances. Furthermore, classification performance is examined with the "classifier ratio" which is the ratio of the GED between sample graph and the correct prototype to the distance of sample and the nearest incorrect prototype. This method leads to the lowest classifier ratio which indicates the least ambiguous classification.

As demonstrated above, this method tends to reduce the level of ambiguity in graph recognition. But the complexity of BLP makes the computation of GED for large graphs difficult.

4.1.5 Method based on subgraph and supergraph

Concrete edit costs for GED are strongly application-dependent and cannot be obtained in a general way, so subgraph and supergraph based method [42] is proposed. It is a special kind of graph distance to approximate the edit distance, which is totally independent of edit costs. This method is based on the conclusion that GED coincides with the MCS of two graphs under the certain cost function [20]. Let $\widehat{G}$ and $\breve{G}$ be a MCS and a minimum common supergraph of

$$G_1 = (V_1, E_1, \alpha_1) \quad \text{and} \quad G_2 = (V_2, E_2, \alpha_2)$$

The distance between G_1 and G_2 is defined by

$$\delta(G_1, G_2) = \left|\breve{G}\right| - \left|\widehat{G}\right|,$$

where $\left|\breve{G}\right|$ is the number of nodes in graph $\breve{G}$ and $\left|\widehat{G}\right|$ is similar. A cost function C is defined as a vector consisting of non-negative real functions

$$(c_{\text{nd}}(v), c_{\text{ni}}(v), c_{\text{ns}}(v_1, v_2), c_{\text{ed}}(e), c_{\text{ei}}(e), c_{\text{es}}(e_1, e_2)),$$

where v, $v_1 \in V_1$, e, $e_1 \in E_1$, $v_2 \in V_2$, $e_2 \in E_2$ and the components orderly represent costs for node deletion, node insertion, node substitution, edge deletion, edge insertion and edge substitution. If the cost function C is specified as

$$C = (c, c, c_{\text{ns}}, c, c, c_{\text{es}}),$$

where c is a constant function which holds that

$$c_{\text{ns}}(v_1, v_2) > 2c \quad \text{and} \quad c_{\text{es}}(e_1, e_2) > 2c$$

for all $v_1 \in V_1$ and $v_2 \in V_2$ with $\alpha_1(v_1) \neq \alpha_2(v_2)$, the GED between $G_1$and G_2 can be computed by the formula $d(G_1, G_2) = c\delta(G_1, G_2)$.

Construction of this method is simple and this method is not relying on fundamental graph edit operations, that is to say, it is independent of cost functions.

The first four algorithms take different approaches to defining cost functions and they are proved to be potent for classifying or clustering some specific images; therefore, they are limited to some specific data. The last method has less limitation and can be used for general attributed graphs; however, this is related to the search of MCS and a minimum common supergraph, which is also difficult for implementing in practice.

4.2 GED for non-attributed graphs

For the non-attributed graphs only having the information of connectivity structure, GED algorithms [31, 35] usually include two parts: conversion of graphs to strings and computation of edit distance for strings [58–60]. Especially, a structural graph may be a tree. Although trees can be viewed as a special kind of graphs, specific characteristics of trees suggest that posing the tree-matching problem as a variant on graph matching is not the best approach. In particular, complexity of both tree isomorphism and subtree isomorphism problems is polynomial time, which is more efficient than general graphs. The similarity of labeled trees is compared in [61] by various methods, in which definitions of cost functions are given ahead. In this paper, specific methods for non-attributed tree matching problem are summarized. Tree edit distance (TED) can be obtained by searching for the maximum weight cliques [62, 63], or embedding trees into a pattern space by constructing super-tree [64], which are presented separately from those of general graphs.

4.2.1 *Tree edit distance*

4.2.1.1 Maximum weight cliques based method The edit distance of the unordered tree still presents a computational bottleneck, therefore, the computation of unordered TED should be efficiently approximated. Bunke's idea of the equivalence of MCS and edit distance computation has been applied to the GED [42], and it can also be extended to the TED [62, 63, 65]. In these algorithms, there is a strong connection between the computation of maximum common subtree and the TED, and searching for the maximum common subtree is transformed into finding a maximum weight clique, so computation of TED is converted into a series of maximum weight clique problems, which is illustrated in Fig. 11.

Similar with the graphs, data tree needs converting into model tree. Under the constraint [66] that the cost of deleting and reinserting the same element with a different label is not greater than the cost of relabeling it, node substitution is to be replaced by node removal and insertion on the data tree. The cost of node insertion on the data tree is dual to that of node removal on the model tree, so the operations to be performed are further reduced to node removal on both trees, which makes the optimal matching completely determined by the subset of nodes left after the minimum edit sequence. The edit distance problem is equal to a particular substructure isomorphism problem.

Given two directed acyclic graphs (DAGs) t and t_0' to be matched, the transitive closures $\ell(t)$ and $\ell(t')$ are calculated. A tree $\widehat{t}$ is an obtainable subtree of the $\ell(t)$ if and only if $\widehat{t}$ is generated from a tree t with a sequence of node removal operations. The minimum cost edited tree isomorphism between t and t_0' is a maximum common obtainable subtree of the two $\ell(t)$ and $\ell(t')$.

Then a maximum common obtainable subtree of the two trees $\ell(t)$ and $\ell(t')$ is searched to induce the optimal matches, which can be transformed into computing the maximum weight clique. It is a quadratic programming problem:

- The objective function is: $\min_x x^T Cx$,

$$\text{s.t.} \quad x \in \Delta \quad \text{where} \quad C = (c_{ij})_{i,j \in V}, \tag{10}$$

$$c_{ij} = \begin{cases} \frac{1}{2w_i} & \text{if } i = j \\ k_{ij} \geq c_{ii} + c_{jj} & \text{if } (i,j) \in E,\ i \neq j, \\ 0 & \text{otherwise} \end{cases}$$

  and w_i is the weight associated with node i.

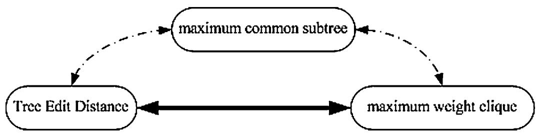

Fig. 11 The relation of TED and maximum weight clique

- Given a set of nodes S, its characteristic vector x^S defined as

$$x_i^S = \begin{cases} \frac{w(i)}{\sum_{j \in S} w(j)} & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases}. \tag{11}$$

  S is a maximum weight clique if and only if x^S is a global (local) minimum for the quadratic problem.

The TED is transformed into a series of maximum weight clique problem and the experiment on 25 shapes illustrates that this algorithm can effectively group similar shapes together.

4.2.1.2 Super-tree based method The analysis of graphs has proved to be considerably more elusive than the analysis of vector patterns; thus, graphs have to be embedded in a vector pattern space in which similar structures are close together and dissimilar ones are far apart. However, there are few methods which can be used to construct low dimensional pattern spaces for sets of graphs. A super-tree, a special kind of graph, is constructed for each set of trees to represent the variations present in the set [64], dimensions of which correspond to principal modes of structural variation. Each tree in a set can be obtained by removing nodes and edges from its corresponding super-tree. Trees are mapped to vectors of fixed length by the super-tree and vectors can be embedded in a low dimension space with principal component analysis. The edit distance between trees can be computed by computing the distance between the low dimensional pattern vectors corresponding to them.

When the database of 25 shapes is used to test the classification performance, this method outperforms the maximum weight cliques based method by 16%.

Consistency of node correspondences during matching is imposed to avoid the underestimation of the distance. A "natural" embedding space of tree structures is derived to analyze how tree representations vary. Although this algorithm has many advantages, it fails to confront larger databases of several shape classes.

4.2.2 *Distance of general graphs*

Compared with string edit distance, GED for general graphs lacks rigorous footing, so graphs are converted into strings and GED can be computed by using string alignment algorithms, for example Dijkstra's algorithm based method [35], maximum a posteriori probability (MAP) based method [31, 59], string kernel based method [58], subgraph based method [60] .

4.2.2.1 Dijkstra's algorithm based method In Dijkstra's algorithm based method [35], graphs are converted into strings, and then the shortest edit path corresponding to the least cost between pairwise strings is determined with Dijkstra's algorithm.

Based on the conclusion that adjacency matrix is associated to the Markov chain, the transition matrix of the Markov chains is the normalized adjacency matrix of a graph $G = (V, E)$, where $V = \{1, 2, \ldots, N\}$. Its leading eigenvector gives the node sequence of the steady state random walk on the graph so that a graph is converted into a string and global structural properties of graphs is characterized. The procedure is shown as below:

1. The adjacent matrix A of the graph is defined;
2. A transition probability matrix P is defined as:

$$P(i,j) = A(i,j) \Big/ \sum_{j \in V} A(i,j); \quad (12)$$

3. The matrix P is converted into a symmetric form for an eigenvector expansion. The diagonal degree matrix D is computed, and its elements are

$$D(i,j) = \begin{cases} 1/d(i) = 1 \Big/ \sum_{j=1}^{|V|} P(i,j), & i = j \\ 0, & \text{otherwise} \end{cases}; \quad (13)$$

The symmetric version of the matrix P is $W = D^{\frac{1}{2}} A D^{\frac{1}{2}}$.

4. The spectral analysis for the symmetric transition matrix W is

$$W = \sum_{i=1}^{|V|} \lambda_i \phi_i \phi_i^T, \quad (14)$$

where λ_i is an eigenvalue of W and ϕ_i is the corresponding eigenvector of unit length.

5. The leading eigenvector ϕ^* gives the sequence of nodes in an iterative procedure and at each iteration k, a list L_k denotes the nodes visited:

   - In the first step, let $L_1 = j_1$, where

     $j_1 = \arg\max_j \phi^*(j)$,

     and neighbors of j_1 is the set $N_{j_1} = \{m | (j_1, m) \in E\}$;
   - In the second step, node j_2 satisfying

     $j_2 = \arg\max_{j \in N_{j_1}} \phi^*(j)$

     is found to form $L_2 = \{j_1, j_2\}$ and the set of neighbors

     $N_{j_2} = \{m | (j_2, m) \in E \wedge m \neq j_1\}$

     of j_2 is hunted;

In the kth step, the node visited is j_k and the list of nodes visited is L_k. The set $N_{j_k} = \{m | (j_k, m) \in E\}$ consists of neighbors of j_k, and then in the $k + 1$th step, node j_{k+1} satisfying $j_{k+1} = \arg\max_{j \in C_k} \phi^*(j)$ is chosen, where $C_k = \{j | j \in N_{j_k} \wedge j \notin L_k\}$;

- The number of step $k = k + 1$;
- The third and fourth steps are repeated until every node in the graph is traversed.

Given the model graph $G_M = (V_M, E_M)$ and the data graph $G_D = (V_D, E_D)$ whose GED is to be computed, strings of these two graphs are determined by the procedure above. The model graph is denoted by

$X = \{x_1, x_2, \ldots, x_{|V_M|}\}$

and the data graph is denoted by

$Y = \{y_1, y_2, \ldots, y_{|V_D|}\}$.

A lattice is constructed, rows of which are indexed using the data-graph string, whereas columns of which are indexed using the model-graph string. An edit path can be found to transform string of data graph into string of model graph, which is denoted by

$\Gamma = \langle \gamma_1, \gamma_2, \ldots, \gamma_k, \ldots, \gamma_L \rangle$

and its elements are Cartesian pairs $\gamma_k \in (V_D \cup \varepsilon) \times (V_M \cup \varepsilon)$, where ε denotes the empty set. The path is constrained to be connected on the edit lattice. The diagonal transition corresponds to the match of an edge of the data-graph to an edge of the model graph. A horizontal transition corresponds to the case where the traversed nodes of the model graph do not have matched nodes in data graph. Similarly, when a vertical transition is made, then the traversed nodes of the data graph do not have matched nodes in model graph. The cost of the edit path is the sum of the costs for the elementary edit operations:

$$C(\Gamma) = \sum_{\gamma_k \in \Gamma} \eta(\gamma_k \to \gamma_{k+1}), \quad (15)$$

where $\eta(\gamma_k \to \gamma_{k+1}) = -\ln P(\gamma_k \to \gamma_{k+1})$ is the cost of the transition from state $\gamma_k = (a, b)$ to $\gamma_{k+1} = (c, d)$. The probability $P(\gamma_k \to \gamma_{k+1})$ is defined as below:

$P(\gamma_k \to \gamma_{k+1}) = \beta_{a,b} \beta_{c,d} R_D(a, c) R_M(b, d)$,

where $\beta_{a,b}$ and $\beta_{c,d}$ are the morphological affinity,

$$R_D(a, c) = \begin{cases} P_D(a, c) & \text{if } (a, c) \in E_D \\ \frac{2|(|V_M| - |V_D|)|}{|V_M| + |V_D|} & \text{if } a = \varepsilon \text{ or } c = \varepsilon, \\ 0 & \text{otherwise} \end{cases}$$

P_D is the transition probability matrix of data graph G_D, and $R_M(b, d)$ is similar with $R_D(a, c)$. The optimal edit path is the one with the minimum cost, that is, $\Gamma^* = \arg\text{-}\min_\Gamma C(\Gamma)$. So, the problem of computing GED is posed as finding the shortest path through the lattice by Dijkstra's algorithm and the GED between these two graphs is $C(\Gamma^*)$.

This is a relatively preliminary work for applying the eigenstructure of the graph adjacency matrix to the graph-matching, and it is improved to be the method in the probability framework.

4.2.2.2 MAP based method The idea of the MAP estimation based algorithm [31, 59] is developed from the Dijkstra's algorithm based method [35]. They differ in the establishment of strings and match of strings. Edit costs are related to different features. All these differences are presented in Table 3.

In the MAP based algorithm, graphs are converted into strings with graph spectral method according to the leading eigenvectors of their adjacency matrices. Similar with the Dijkstra's algorithm based method, GED is the cost of the least expensive edit path Γ^*, but the path Γ^* is found based on the idea of Levenshtein distance in probability framework. The cost for the elementary edit operations is defined as:

$$\eta(\gamma_k \rightarrow \gamma_{k+1}) = -\ln P(\gamma_k | \phi_X^*(x_j), \phi_Y^*(y_i)) - \ln P(\gamma_{k+1} | \phi_X^*(x_{j+1}), \phi_Y^*(y_{i+1})) - \ln R_{k,k+1}, \tag{16}$$

where the edge compatibility coefficient $R_{k,k+1}$ is

$$R_{k,k+1} = \frac{P(\gamma_k, \gamma_{k+1})}{P(\gamma_k)P(\gamma_{k+1})} = \begin{cases} \rho_M \rho_D & \text{if } \gamma_k \rightarrow \gamma_{k+1} \text{ is a diagonal transition on the the edit lattice, i.e., } (y_i, y_{i+1}) \in E_D \text{ and } (x_j, x_{j+1}) \in E_M \\ \rho_M & \text{if } \gamma_k \rightarrow \gamma_{k+1} \text{ is a vertical transition on the the edit lattice, i.e., } (y_i, y_{i+1}) \in E_D \text{ and } x_j = \varepsilon \text{ or } x_{j+1} = \varepsilon \\ \rho_D & \text{if } \gamma_k \rightarrow \gamma_{k+1} \text{ is a horizontal transition on the the edit lattice, } i.e., y_i = \varepsilon \text{ or } y_{i+1} = \varepsilon \text{ and } (x_j, x_{j+1}) \in E_M \\ 1 & \text{if } y_i = \varepsilon \text{ or } y_{i+1} = \varepsilon \text{ and } x_j = \varepsilon \text{ or } x_{j+1} = \varepsilon \end{cases}$$

and

$$P(\gamma_k | \phi_X^*(x_j), \phi_Y^*(y_i)) = \begin{cases} \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{1}{2\sigma^2} (\phi_X^*(x_j) - \phi_Y^*(y_i))^2 \right\} & \text{if } x_j \neq \varepsilon \text{ and } y_i \neq \varepsilon \\ \alpha & \text{if } x_j = \varepsilon \text{ or } y_i = \varepsilon \end{cases}$$

Given images in three sequences: CMU-VASC sequence [67], the INRIA MOVI sequence [68], and a sequence of views of a model Swiss chalet, their GED matrix is computed with this method. The result is shown in Fig. 12. Each element of the matrix specifies the color of a rectilinear patch in Fig. 12 and the deeper color corresponds to the smaller distance. All patches constitute nine blocks. Coordinates 1–10, 11–20 and 21–30 correspond to CMU-VASC sequence, INRIA MOVI sequence and Swiss chalet sequence, respectively; therefore blocks along the diagonal present within-class distances and other blocks present between-class distances. In each block, the row and column indexes increase monotonically according to the viewing angle of each sequence. Color of diagonal blocks is deeper than that of other areas, and it is obvious that GED within a class is lower than that between classes on the whole.

Table 3 Comparison of the MAP based algorithm and the Dijkstra's algorithm based method

Methods	MAP based algorithm	Dijkstra's algorithm based method
Establishment of the serial ordering	Using the leading eigenvector of the graph adjacency matrix	Using the leading eigenvector of the normalized graph adjacency matrix
String matching	A MAP alignment of the strings for pairwise graphs	Searching for the optimal edit sequence using Dijkstra's algorithm
Edit costs	Related to the edge density of two graphs	Related to the degree and adjacency of nodes

Compared with the Dijkstra's algorithm based method, this method has the following two advantages: when graphs are converted into strings, the adjacency matrix needs not normalization, which decreases computation complexity; when strings are matched, the computation of minimal edit distance is cast in a probabilistic setting so that statistical models can be used for the cost definition.

4.2.2.3 String kernel based method String kernels can be used to measure the similarity of seriated graphs, which makes the computation of GED more efficient. In the string kernel based algorithm [58], graphs are seriated into strings with semidefinite programming (SDP) whose steps are given as below.

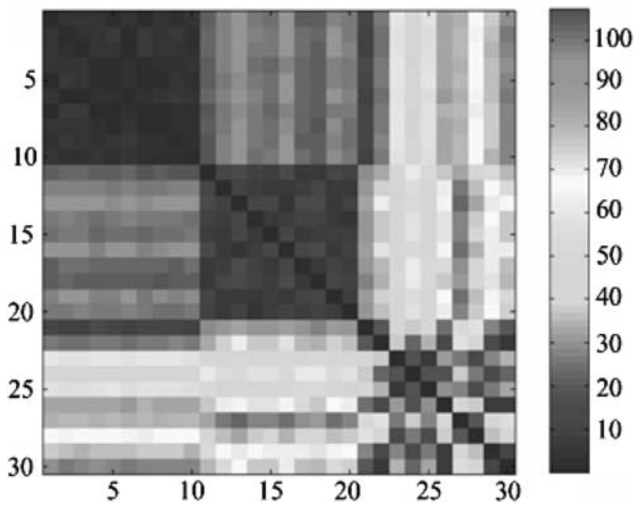

Fig. 12 GED matrix in [31]

- Let B be $\Omega^{1/2}A\Omega^{-1/2}$ and y be $\Omega^{1/2}x^*$, where

$$\Omega = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 2 & 0 & \cdots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \cdots & 2 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix},$$

  A is the adjacency matrix of the graph to be converted into a string, and x^* denotes the value to be solved. If $Y^* = yy^T$, the SDP is represented as the following formula:

$$\arg\min_{Y^*} \operatorname{trace}(BY^*),$$

  such that $\operatorname{trace}(EY^*) = 1$, where E is the unit matrix. Matrix Y^* can be solved with the method in [69] so as to obtain x^*.
- Similar with the idea of converting a graph into a string with the leading eigenvector in [35], the graph is converted into a string according to the vector x^*.

With strings obtained, kernel feature is applied to presenting the times of a substring occurring in a string and is weighted by the length of the substring. Elements of a kernel feature vector for a string correspond to substrings. The inner product of the kernel feature vectors corresponding to two strings is called as string kernel function. The kernel function gives sum of frequency of all common substrings weighted by their lengths. String kernel function works based on the idea that the strings are more similar if they share more common substrings.

COIL image database [70] is used to evaluate this method. Six objects are selected from the database and each object has 20 different views. Distance of images belonging to the same class is much smaller than that of images between classes and images corresponding to different objects can be clustered well.

The SDP overcomes local optimality of the graph spectral method used in the Dijkstra's algorithm based and MAP based methods, and string kernel function is more efficient than aligning strings with Dijkstra's algorithm.

4.2.2.4 Subgraph based method Because of potentially exponential complexity of the general inexact graph-matching problem, it is decomposed into a series of simpler subgraph matching problems [60]. A graph $G = (V, E)$ is partitioned into non-overlapping super cliques according to Fiedler vector:

- The list $\Gamma = \{j_1, j_2, \ldots, j_{|V|}\}$ is the node rank-order which is determined under the conditions that the permutation satisfies

$$\pi(j_1) < \pi(j_2) < \cdots < \pi(j_{|V|})$$

  and the components of the Fiedler vector is

$$x_{j_1} > x_{j_2} > \cdots > x_{j_{|V|}}.$$

  The weight assigned to the node $i \in V$ is $w_i = \text{rank}(\pi(i))$ and the significance score of the node i being a center node is computed, that is S_i, according to degree and weight of node i;
- The list Γ is traversed until a node k is founded which is neither in the perimeter nor whose score s_k exceeds those of its neighbors. Node k and its neighborhood N_k constitute a super clique and they are deleted from the list Γ, that is $\Gamma = \Gamma - \{k\} \cup N_k$. This procedure is repeated until $\Gamma = \phi$, and then the non-overlapping neighborhoods of the graph G are located.

With super cliques in hand, a graph G' containing super cliques of the original graph G is constructed, in which the nodes denote the super cliques and the edges indicate whether these super cliques are connected in the original graph. Such graphs are matched based on the matching of the super clique set, that is, the super clique-to-super clique matching, which is computed by the conversion of super cliques into strings based on the cyclic permutations of the peripheral nodes about the center nodes and the Levenshtein distance between strings.

This method partitions a graph into subgraphs, and therefore the process may cast into a hierarchical framework and be suitable for parallel computation.

Trees, as a special kind of graphs, have some attributes superior to general graphs and TED can be computed without definition of cost functions, which has been applied to shape classification [71] with shock-tree as a structural representation of 2D shape. But it is obvious that the key issue of GED algorithms for general graphs is still definition of cost functions. Each method defines cost functions in a task specific way from heuristics of the problem domain in a trial-and-error fashion and further research is still needed to derive cost functions in a general method.

5 Conclusion

Graph edit distance is a flexible error-tolerant mechanism to measure distance between two graphs, which has been widely applied to pattern recognition and image retrieval. The research of GED is studied and surveyed in this paper. Existing GED algorithms are categorized and presented in detail. Advantages and disadvantages of these algorithms are uncovered by comparing them experimentally and theoretically. Although the research has remained for several decades and yielded substantial results, there are few robust algorithms suitable for all kinds of graphs and several problems deserve future research.

1. In the computation of GED, how to compare the similarity of corresponding nodes and edges in two graphs is still not solved well. For attributed graphs, attributes of nodes and edges can be used for comparing the similarity. But which attributes should be adopted and available for computing distance remains an open problem. For non-attributed graphs, the connectivity of the graph can be used for comparing the similarity. But how to characterize the connectivity to achieve a better evaluation of similarity remains unsolved.
2. The definition of costs for edit operations is also important for GED, which affects the rationality of GED directly. Existing researches of GED mainly focus on this problem and each of them is available for limited applications, or under some constrains, so some definitions of costs, which can be applied extensively and easily, are demanded.
3. Many ways of searching for least expensive edit sequence have been used previously. The search strategy should be consistent with the method of similarity comparison and the definition of edit cost, instead of the best one in theory. So an appropriate search strategy for the minimum edit costs sequence should be studied to improve both the efficiency and accuracy of GED algorithms.

Acknowledgments We want to thank the helpful comments and suggestions from the anonymous reviewers. This research has been partially supported by National Science Foundation of China (60771068, 60702061, 60832005), the Open-End Fund of National Laboratory of Pattern Recognition in China and National Laboratory of Automatic Target Recognition, Shenzhen University, China, the Program for Changjiang Scholars and innovative Research Team in University of China (IRT0645).

References

1. Umeyama S (1988) An eigendecomposition approach to weighted graph matching problems. IEEE Trans Pattern Anal Mach Intell 10(5):695–703
2. Bunke H (2000) Recent developments in graph matching. In: Proceedings of IEEE international conference on pattern recognition, Barcelona, pp 117–124
3. Caelli T, Kosinov S (2004) An eigenspace projection clustering method for inexact graph matching. IEEE Trans Pattern Anal Mach Intell 26(4):515–519
4. Cross ADJ, Wilson RC, Hancock ER (1997) Inexact graph matching using genetic search. Pattern Recognit 30(7):953–970
5. Wagner RA, Fischer MJ (1974) The string-to-string correction problem. J ACM 21(1):168–173
6. Pavlidis JRT (1994) A shape analysis model with applications to a character recognition system. IEEE Trans Pattern Anal Mach Intell 16(4):393–404
7. Wang Y-K, Fan K-C, Horng J-T (1997) Genetic-based search for error-correcting graph isomorphism. IEEE Trans Syst Man Cybern B Cybern 27(4):588–597
8. Sebastian TB, Klien P, Kimia BB (2004) Recognition of shapes by editing their shock graphs. IEEE Trans Pattern Anal Mach Intell 26(5):550–571
9. He L, Han CY, Wee WG (2006) Object recognition and recovery by skeleton graph matching. In: Proceedings of IEEE international conference on multimedia and expo, Toronto, pp 993–996
10. Shearer K, Bunke H, Venkatesh S (2001) Video indexing and similarity retrieval by largest common subgraph detection using decision trees. Pattern Recognit 34(5):1075–1091
11. Lee J (2006) A graph-based approach for modeling and indexing video data. In: Proceedings of IEEE international symposium on multimedia, San Diego, pp 348–355
12. Tao D, Tang X (2004) Nonparametric discriminant analysis in relevance feedback for content-based image retrieval. In: Proceedings of IEEE international conference on pattern recognition, Cambridge, pp 1013–1016
13. Tao D, Tang X, Li X et al (2006) Kernel direct biased discriminant analysis: a new content-based image retrieval relevance feedback algorithm. IEEE Trans Multimedia 8(4):716–727
14. Tao D, Tang X, Li X (2008) Which components are important for interactive image searching? IEEE Trans Circuits Syst Video Technol 18(1):1–11
15. Christmas WJ, Kittler J, Petrou M (1995) Structural matching in computer vision using probabilistic relaxation. IEEE Trans Pattern Anal Mach Intell 17(8):749–764
16. Gao X, Zhong J, Tao D et al (2008) Local face sketch synthesis learning. Neurocomputing 71(10–12):1921–1930
17. Sanfeliu A, Fu KS (1983) A distance measure between attributed relational graphs for pattern recognition. IEEE Trans Syst Man Cybern 13(3):353–362
18. Messmer BT, Bunke H (1994) Efficient error-tolerant subgraph isomorphism detection. Shape Struct Pattern Recognit:231–240
19. Messmer BT, Bunke H (1998) A new algorithm for error-tolerant subgraph isomorphism detection. IEEE Trans Pattern Anal Mach Intell 20(5):493–504
20. Bunke H (1997) On a relation between graph edit distance and maximum common subgraph. Pattern Recognit Lett 18(8):689–694
21. Bunke H (1999) Error correcting graph matching: on the influence of the underlying cost function. IEEE Trans Pattern Anal Mach Intell 21(9):917–922

22. Shasha D, Zhang K (1989) Simple fast algorithms for the editing distance between trees and related problems. SIAM J Comput 18(6):1245–1262
23. Zhang K (1996) A constrained edit distance between unordered labeled trees. Algorithmica 15(3):205–222
24. Myers R, Wilson RC, Hancock ER (2000) Bayesian graph edit distance. IEEE Trans Pattern Anal Mach Intell 22(6):628–635
25. Wei J (2004) Markov edit distance. IEEE Trans Pattern Anal Mach Intell 26(3):311–321
26. Marzal A, Vidal E (1993) Computation of normalized edit distance and applications. IEEE Trans Pattern Anal Mach Intell 15(9):926–932
27. Myers R, Wilson RC, Hancock ER (1998) Efficient relational matching with local edit distance. In: Proceedings of IEEE international conference on pattern recognition, Brisbane, pp 1711–1714
28. Wilson RC, Hancock ER (1997) Structural matching by discrete relaxation. IEEE Trans Pattern Anal Mach Intell 19(6):634–648
29. Levenshtein V (1966) Binary codes capable of correcting deletions, insertions, and reversals. Sov Phys Dokl 10(8):707–710
30. Neuhaus M, Bunke H (2004) A probabilistic approach to learning costs for graph edit distance. In: Proceedings of IEEE international conference on pattern recognition, Cambridge, pp 389–393
31. Robles-Kelly A, Hancock ER (2005) Graph edit distance from spectral seriation. IEEE Trans Pattern Anal Mach Intell 27(3):365–378
32. Xiao B, Gao X, Tao D et al (2008) HMM-based graph edit distance for image indexing. Int J Imag Syst Tech 18(2–3):209–218
33. Gao X, Xiao B, Tao D et al (2008) Image categorization: graph edit distance + edge direction histogram. Pattern Recognit 47(10):3179–3191
34. Neuhaus M, Bunke H (2005) Self-organizing maps for learning the edit costs in graph matching. IEEE Trans Syst Man Cybern B Cybern 35(3):503–514
35. Robles-Kelly A, Hancock ER (2004) String edit distance, random walks and graph matching. Int J Pattern Recogn Artif Intell 18(3):315–327
36. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc B 39(1):1–38
37. Dijkstra EW (1959) A note on two problems in connexion with graphs. Numer Math 1:269–271
38. Azcarraga AP, Hsieh M-H, Pan SL et al (2005) Extracting salient dimensions for automatic SOM labeling. IEEE Trans Syst Man Cybern C Appl Rev 35(4):595–600
39. Kohonen T (1995) Self organizing maps. Springer, New York
40. Bhattacharyya S, Dutta P, Maulik U (2007) Binary object extraction using bi-directional self-organizing neural network (BDSONN) architecture with fuzzy context sensitive thresholding. Pattern Anal Appl 10(4):345–360
41. Neuhaus M, Bunke H (2006) A convolution edit kernel for error-tolerant graph matching. In: Proceedings of IEEE international conference on pattern recognition, Hong Kong, pp 220–223
42. Fernández M-L, Valiente G (2001) A graph distance metric combining maximum common subgraph and minimum common supergraph. Pattern Recognit Lett 22(6–7):753–758
43. Justice D, Hero A (2006) A binary linear programming formulation of the graph edit distance. IEEE Trans Pattern Anal Mach Intell 28(8):1200–1214
44. Davies DL, Bouldin DW (1979) A cluster separation measure. IEEE Trans Pattern Anal Mach Intell 1(2):224–227
45. Dunn JC (1974) Well-separated clusters and optimal fuzzy partitions. J Cybern 4:95–104
46. Hubert LJ, Schultz JV (1976) Quadratic assignment as a general data analysis strategy. Br J Math Stat Psychol 29:190–241
47. Goodman LA, Kruskal WH (1954) Measures of association for cross classification. J Am Stat Assoc 49:732–764
48. Calinski T, Harabasz J (1974) A dendrite method for cluster analysis. Commun Stat 3(1):1–27
49. Rand W (1971) Objective criteria for the evaluation of clustering methods. J Am Stat Assoc 66(336):846–850
50. Jain A, Dubes R (1988) Algorithms for clustering data. Prentice-Hall, Englewood Cliffs
51. Fowlkes EB, Mallows CL (1983) A method for comparing two hierarchical clusterings. J Am Stat Assoc 78:553–584
52. Ristad E, Yianilos P (1998) Learning string edit distance. IEEE Trans Pattern Anal Mach Intell 20(5):522–532
53. García V, Mollineda RA, Sánchez JS (2008) On the k-NN performance in a challenging scenario of imbalance and overlapping. Pattern Anal Appl 11(3–4):269–280
54. Neuhaus M, Bunke H (2005) Edit distance based kernel functions for attributed graph matching. In: Proceedings of 5th international workshop graph-based representations in pattern recognition, Poitiers, pp 352–361
55. Artner TG, Flach P, Wrobel S (2003) On graph kernels: hardness results and efficient alternatives. In: Proceedings of 16th annual conference on learning theory, Washington, pp 129–143
56. Saux BL, Bunke H (2005) Feature selection for graph-based image classifiers. In: Proceedings of 2nd Iberian conference on pattern recognition and image analysis, Estoril, pp 147–154
57. Dunford-Shore B, Sulaman W, Feng B et al (2002) Klotho: biochemical compounds declarative database. http://www.biocheminfo.org/klotho/
58. Yu H, Hancock ER (2006) String kernels for matching seriated graphs. In: Proceedings of IEEE international conference on pattern recognition, Hong Kong, pp 224–228
59. Robles-Kelly A, Hancock ER (2003) Edit distance from graph spectra. In: Proceedings of IEEE international conference on computer vision, Nice, pp 234–241
60. Qiu HJ, Hancock ER (2006) Graph matching and clustering using spectral partitions. Pattern Recognit 39(1):22–34
61. Bille P (2005) A survey on tree edit distance and related problems. Theor Comput Sci 337(1–3):217–239
62. Torsello A, Robles-Kelly A, Hancock ER (2007) Discovering shape classes using tree edit-distance and pairwise clustering. Int J Comput Vis 72(3):259–285
63. Torsello A, Hancock ER (2003) Computing approximate tree edit distance using relaxation labeling. Pattern Recognit Lett 24(8):1089–1097
64. Torsello A, Hancock ER (2007) Graph embedding using tree edit-union. Pattern Recognit 40(5):1393–1405
65. Torsello A, Hancock ER (2001) Efficiently computing weighted tree edit distance using relaxation labeling. In: Proceedings of energy minimization methods in computer vision and pattern recognition. Springer, Sophia Antipolis, pp 438–453
66. Bunke H, Kandel A (2000) Mean and maximum common subgraph of two graphs. Pattern Recognit Lett 21(2):163–168
67. Vision and Autonomous Systems Center's Image Database. http://vasc.ri.cmu.edu//idb/html/motion/house/index.html
68. INRIA-MOVI houses. http://www.inria.fr/recherche/equipes/movi.en.html
69. Fujisawa K, Futakata Y, Kojima M et al (2008) Sdpa-m user's manual. http://sdpa.is.titech.ac.jp/SDPA-M
70. Columbia Object Image Library. http://www1.cs.columbia.edu/CAVE/software/softlib/coil-20.php
71. Torsello A, Robles-Kelly A, Hancock ER (2007) Discovering shape classes using tree edit-distance and pairwise clustering. Int J Comput Vis 72(3):259–285

Author Biographies

Xinbo Gao received his Ph.D. degree in signal and information processing in 1999, at Xidian University, Xi'an, China. Since 2001, he joined Xidian University, where he is currently a Full Professor at the School of Electronic Engineering and Director of VIPS Lab. His research interests include computational intelligence, machine learning, visual information processing and analysis, and pattern recognition. In these areas, he has published four books and over 100 technical articles in refereed journals and proceedings. He is on the editorial boards of several journals including EURASIP Signal Processing Journal. He served as general chair/co-chair or program committee chair/co-chair or PC member for around 30 major international conferences. He is a senior member of IEEE and IET.

Bing Xiao received the B.S. degree in Computer Science and Technology and the M.Eng. degree in Computer Software and Theory from Shaanxi Normal University, Xi'an, China, in 2003 and 2006, respectively. Since August 2006, she has been working toward the Ph.D. degree in Intelligent Information Processing at Xidian University, Xi'an, China. Her research interests include pattern recognition and computer vision.

Dacheng Tao received the B.Eng. degree from USTC, the MPhil degree from CUHK, and the PhD degree from Lon. Currently, he is a Nanyang Assistant Professor in Nanyang Technological University, a Visiting Professor in Xidian University, a Guest Professor in Wuhan University, and a Visiting Research Fellow in Lon. His research is mainly on applying statistics and mathematics for data analysis problems in computer vision, data mining, and machine learning. He has published more 90 scientific papers in IEEE T-PAMI, T-KDE, T-IP, with best paper awards. He is an associate editor of IEEE T-KDE, Neurocomputing (Elsevier) and CSDA (Elsevier).

Xuelong Li holds a permanent academic post at Birkbeck College, University of London. He is also a visiting/guest professor at Tianjin University and USTC. His research focuses on cognitive computing, image/video processing, pattern recognition, and multimedia. He has 140 publications with several Best Paper Awards and finalists. He is an author/editor of four books, an associate editor of Pattern Analysis and Applications (Springer), four IEEE transactions, and ten other journals.