



Graph embedding using tree edit-union

Andrea Torsello^{a,*}, Edwin R. Hancock^b

^a*Dip. di Informatica, Università Ca' Foscari di Venezia, via Torino 155 Mestre (VE), Italy*

^b*Department of Computer Science, University of York, York YO10 5DD, UK*

Received 16 July 2005; received in revised form 30 August 2006; accepted 8 September 2006

Abstract

In this paper we address the problem of how to learn a structural prototype that can be used to represent the variations present in a set of trees. The prototype serves as a pattern space representation for the set of trees. To do this we construct a super-tree to span the union of the set of trees. This is a chicken and egg problem, since before the structure can be estimated correspondences between the nodes of the super-tree and the nodes of the sample tree must be hand. We demonstrate how to simultaneously estimate the structure of the super-tree and recover the required correspondences by minimizing the sum of the tree edit-distances over pairs of trees, subject to edge consistency constraints. Each node of the super-tree corresponds to a dimension of the pattern space, and for each tree we construct a pattern vector in which the elements of the weights corresponding to each of the dimensions of the super-tree. We perform pattern analysis on the set of trees by performing principal components analysis on the vectors. The method is illustrated on a shape analysis problem involving shock-trees extracted from the skeletons of 2D objects.

© 2006 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: 2D shape; Skeleton; Tree-union; Embedding

1. Introduction

The analysis of relational patterns, or graphs, has proved to be considerably more elusive than the analysis of vectorial patterns. Relational patterns arise naturally in the representation of data in which there is a structural arrangement, and are encountered in computer vision, genomics and network analysis. One of the challenges that arises in these domains is that of knowledge discovery from large graph datasets. The tools that are required in this endeavor are robust algorithms that can be used to organize, query and navigate large sets of graphs. In particular, the graphs need to be embedded in a pattern space so that similar structures are close together and dissimilar ones are far apart. This is a routine procedure with pattern vectors, and may be addressed using a variety of techniques including principal components analysis [1], locally linear embedding [2], isomap [3] or the Laplacian

eigenmap [4]. Collectively these methods are sometimes referred to as manifold learning theory. However, there are few analogous methods which can be used to construct low-dimensional pattern spaces for sets of graphs.

1.1. Background

There are two reasons why pattern vectors are more easily manipulated than graphs. First, there is no canonical ordering for the nodes in a graph, unlike the components of a vector. Hence, correspondences to a reference structure must be established as a prerequisite. The second problem is that the variation in the graphs of a particular class may manifest itself as subtle changes in structure, which may involve different numbers of nodes or different edge structure. Even if the nodes or the edges of a graph could be encoded in a vectorial manner, then the vectors would be of variable length. This problem becomes particularly acute when the central clustering of graphs is attempted. Here an explicit class archetype and correspondences with it must be maintained [5,6]. This class archetype must capture both

* Corresponding author. Tel.: +39 3498774166; fax: +39 0412348419.

E-mail addresses: torsello@dsi.unive.it (A. Torsello),

erh@cs.york.ac.uk (E.R. Hancock).

the salient structure of the class and the modes of variation contained within it.

There has been work aimed at recovering such archetypes. Munger et al. [7] have taken some important steps in this direction by developing a genetic algorithm for searching for median graphs. Variations in class structure can be captured and learned provided that a suitable probability distribution can be defined over the nodes and edges of the archetype. As a concrete example, the random graphs of Wong et al. [8] capture this distribution using a discretely defined probability distribution, and Bagdanov and Worring [9] have overcome some of the computational difficulties associated with this method by using continuous Gaussian distributions. There is a considerable body of related literature in the graphical models community concerned with learning the structure of Bayesian networks from data [10].

Recently there has been renewed interest how to learn the class archetype for the purpose of clustering data abstracted in terms of graphs. Lozano and Escolano [11], and Bunke et al. [12] summarize the data by creating supergraph representation from available samples. Jain and Wyszotzki, adopt a geometric approach which aims to embed graphs in a high-dimensional space by means of the Schur–Hadamard inner product [13]. Central clustering methods are then deployed to learn the class structure of the graphs. The embedding offers the advantage that it is guaranteed to preserve structural information present. Unfortunately, the algorithm does not have a means of characterizing the modes of structural variation encountered.

However, in general the problem of constructing class archetypes has proved to be an elusive one. For this reasons much of the work in the literature has turned to simpler alternatives.

An alternative to constructing a class archetype, is to use pairwise methods. Here the starting point is a matrix of pairwise affinities between graphs. Using pairwise distances both embedding and clustering can be effected. Graphs may be embedded in a low-dimensional pattern space using either multi-dimensional scaling [14] or variants of the ISOMAP algorithm [15]. There are a variety of algorithms available for performing pairwise clustering. One of the most popular approaches is to use spectral methods which use the eigenvectors of an affinity matrix to define clusters [16,17]. Pairwise methods have been successfully applied to the graph clustering problem using both graph edit-distance and more informal similarity measures [14,18]. However, pairwise methods do not necessarily result in a pattern space where the dimensions reflect the modes of structural variation of the trees. Furthermore, pairwise distance algorithms consistently underestimate the distance between patterns belonging to different clusters. When two graphs are similar, the node correspondences can be estimated reliably, but as the graphs move farther apart in pattern space the estimation becomes less reliable. This is due to the fact that correspondences are chosen to minimize the distance between trees and, as the

trees move farther apart, the advantage the “correct” correspondence has over alternative ones diminishes, until, eventually, a match which yields a lower distance is selected.

A second alternative to the use of structural archetypes is to extract feature vectors from the graphs and to use these to construct a “proxy” pattern space representation. There are a number of ways in which this can be realized for graphs. One approach is to extract structural or topological features from the graphs under study. Candidates here include diameter, edge density or path length distribution. Graph spectral features extracted from the eigenvalues and eigenvectors of the adjacency or Laplacian matrices have also proved effective here [19–21]. Specific examples include the use of the Laplacian or adjacency matrix spectrum [19], or the use of permutation invariants derived from graph polynomials [21]. Once such feature vectors are to hand, then manifold learning theory may be applied to the representation to project them into a pattern space. However, this method has the limitation that the “proxy” representation does not capture the underlying structural variation of the graphs under study.

1.2. Contribution

This paper is concerned with the analysis of sets of trees, and we focus on the problem of estimating a structural archetype for a set of trees. Specifically, we address the problem of how to organize trees into a pattern space where (a) similar trees are close to one another and (b) the space is traversed in a relatively uniform manner as the trees are gradually modified. In other words, the aim is to embed the trees in a vector–space where the dimensions correspond to principal modes of structural variation. In many ways this is a prerequisite to learning a representation for a set of graphs.

To overcome the limitations of the approaches noted above, here we take a different approach to the problem. We aim to embed trees in a pattern space by mapping them to vectors of fixed length. We do this as follows. We commence from a set of trees, and from this we construct a super-tree from which each tree may be obtained by the edit operations of node and edge removal. Hence, each tree is an edited subtree of the super-tree. The super-tree is constructed so that it minimizes the total edit-distance to the set of trees. To embed the individual trees in a vector–space we allow each node of the super-tree to represent a dimension of the space. Each tree is represented in this space by a vector which has non-zero components only in the directions corresponding to its constituent nodes. The non-zero components of the vectors are the weights of the nodes. In this space, the edit-distance between trees is the L_1 norm between their embedded vectors.

The outline of this paper is as follows. In Section 2 we detail how tree edit-distance is computed, and how it relates to the concept of obtainability. Section 3 describes our method for constructing the tree-union. This account is

divided into two parts. We commence by describing how a tree can be matched to a tree-union so as to minimize edit-distance. Next, we describe how trees can be structurally merged together, so that the resulting structure is itself a tree. Section 4 describes how principal components analysis can be applied to the resulting structure, so that the sample trees can be embedded in a low-dimensional pattern space. Section 5 describes experiments on clustering sets of shock-trees, extracted from 2D shape silhouettes. Finally, Section 6 offers some conclusions and suggests directions for future research.

2. Tree edit-distance

We commence with some definitions that will be used throughout the paper. A directed graph $G = (V, E)$ is a pair consisting of a node set V and a set of edges $E \subseteq V \times V$. Given two nodes $a, b \in V$, the node b is said to be *adjacent* to a (denoted with $a \sim b$) if $(a, b) \in E$. A *path* is a sequence of nodes a_1, a_2, \dots, a_n such that $\forall i = 1, \dots, n-1$, we have $a_i \sim a_{i+1}$. Node b is said to be *reachable* from node a (denoted $a \rightsquigarrow b$) if there is a path where a is the first node and b is the last node. A *cycle* is a path where the last node is the same as the first node. A graph $t = (V, E)$ is a *rooted tree* if

- (1) There is no cycle in t .
- (2) There is a node $r \in V$ such that every other node in V is reachable from it ($\forall n \in V \setminus \{r\}, r \rightsquigarrow n$).

Let $t = (V, E)$ be a tree and let $a, b, c \in V$, $a \sim b$ and $a \sim c$, then a is said to be the *parent* of b and c , b and c are said to be *children* of a , and b is said to be a *sibling* of c . Furthermore, if $a \rightsquigarrow b$, then a is said to be an *ancestor* of b and b is said to be a *descendent* of a . Given two trees $t_1 = (V_1, E_1)$ and $t_2 = (V_2, E_2)$, a function $\phi : V_1 \rightarrow V_2$ is said to be a *tree isomorphism* if it is invertible and $\forall a, b \in V_1 [(a, b) \in E_1 \iff (\phi(a), \phi(b)) \in E_2]$. Two trees t_1 and t_2 are said to be *isomorphic* (denoted by $t_1 \approx t_2$) if there exists a tree isomorphism between them.

The idea behind edit-distance is that it is possible to identify a set of basic edit operations on nodes and edges of a structure, and to associate with these operations a cost. The edit-distance is found by searching for the sequence of edit operations that will make the two graphs isomorphic with one another and which has minimum cost. Formally, given two trees $t_1 = (V_1, E_1)$ and $t_2 = (V_2, E_2)$, the edit-distance between them is defined in terms of the sequence of basic edit operations that make t_1 and t_2 isomorphic with one another. Following common use, we consider three fundamental operations:

- *Node removal*: This operation removes a node and links the children to the parent of said node.
- *Node insertion*: The dual of node removal, inserts a new node between a parent and its children.

- *Node relabel*: This operation changes the weight or attribute of a node.

Note that these operations cannot form cycles and maintain a path from the root to any node remaining in the graph. Hence, the edit operations transform trees into trees. Let \mathcal{S}_1 and \mathcal{S}_2 be sequences of edit operations on trees t_1 and t_2 , respectively, and denote with $\mathcal{S}_1(t_1)$ and $\mathcal{S}_2(t_2)$ the transformed trees. We say that \mathcal{S}_1 and \mathcal{S}_2 form a *matching edit sequence* if they transform the trees into isomorphic trees, i.e. if $\mathcal{S}_1(t_1) \approx \mathcal{S}_2(t_2)$. A matching sequence is *minimal* if the sum of the cost of the edit operations is the minimum over all possible matching sequences. The edit-distance between t_1 and t_2 is defined as the cost of a minimal matching edit sequence. By making the evaluation of structural modification explicit, edit-distance provides a very effective way of measuring the similarity of relational structures. Moreover, the method has considerable potential for error tolerant object recognition and indexing problems [22–24].

Unfortunately, the task of calculating graph edit-distance is a computationally hard problem [25]. The special case of computing graph edit-distance for trees with an order relation among siblings is solvable in polynomial time [25,35]. However, the general case of tree edit-distance is still NP-hard, hence, goal-directed approximations are necessary to calculate it [26]. In Ref. [26] we presented an approximation algorithm. It works by associating with each possible match between a node of one tree to a node of the other tree, a max-clique problem, for a total of NM clique problems, where N and M are the size of the trees to be matched. The max-clique problems are then approximated using an iterative algorithm that is guaranteed to converge to a feasible but suboptimal solution. The complexity of each iteration is of order $O(n^2m^2)$ where $n < N$ and $m < M$ are the number of nodes in each tree that are descendents of the nodes forming the match associated with the clique. Assuming a constant number of iterations k , the complexity turns out to be $O(kN^2M^2)$. In this paper we will build upon this matching algorithm to construct a pattern space for tree structures.

2.1. Obtainability

To commence, we provide some results concerning edit operations on trees that are instrumental to the construction of a structural archetype for a set of trees. The idea central to this development is that by transforming node insertions in one tree into node removals in the other allows us to use only structure reducing operations. This, in turn, means that the edit-distance between two trees is completely determined by the subset of nodes remaining after the optimal removal sequence.

To pursue this development we need to restate some of the results presented in Ref. [26]. The *closure* of a tree $t = (v, E)$, denoted by $\mathcal{C}(t)$, is a graph $G = (V, E')$ that has the same

node set as the original tree and an edge set that satisfies the condition $(a, b) \in E' \iff a \rightsquigarrow b$ in t . Let $\mathcal{C}(t)$ be the closure of tree t and $E_v(t)$ be the edit operation that removes node v from t . We can define an equivalent edit operation $\mathcal{E}_v(\mathcal{C}(t))$ that removes v from the closure as follows: we remove the node v and any edge adjacent to v , i.e. that has v either as origin or destination. The first result is that edit and closure operations commute, i.e. $\mathcal{E}_v(\mathcal{C}(t)) = \mathcal{C}(E_v(t))$. For the second result we need some more definitions: we call a subtree s of $\mathcal{C}(t)$ *obtainable* if for each node v of s there cannot be two children a and b so that (a, b) is in $\mathcal{C}(t)$. In other words, for s to be obtainable, there cannot be a path in t connecting two nodes that are siblings in s . We can, now, introduce the following:

Theorem 1. *A tree \hat{t} can be generated from a tree t with a sequence of node removal operations if and only if \hat{t} is an obtainable subtree of the directed acyclic graph $\mathcal{C}(t)$.*

By virtue of the theorem above, the node correspondences yielding the minimum edit-distance between trees t and t' form an obtainable subtree of both $\mathcal{C}(t)$ and $\mathcal{C}(t')$. Hence, we reduce the problem to that of searching for a common substructure, or the maximum common obtainable subtree (MCOS).

2.2. Distance between weighted trees

Formally, we are interested in the problem of computing the edit-distance between the trees $t_1 = (V_1, E_1)$ and $t_2 = (V_2, E_2)$ where V_1 and V_2 are the node sets of the trees and $E_1 \subseteq V_1 \times V_1$ and $E_2 \subseteq V_2 \times V_2$ are their edge sets. Further suppose that \mathcal{S}_1 and \mathcal{S}_2 form a matching sequence of edit operations, that R_1 and R_2 are the sets of nodes of t_1 and t_2 , that are removed by \mathcal{S}_1 and \mathcal{S}_2 , respectively, and that $M_1 = V_1 \setminus R_1$ and $M_2 = V_2 \setminus R_2$ are the nodes that are left after editing the trees. Since \mathcal{S}_1 and \mathcal{S}_2 form a matching sequence, there will be at least one isomorphism $\phi : M_1 \rightarrow M_2$. We say that $M = \{(x, \phi(x)) | x \in M_1\}$ is a set of matches induced by the matching sequence. According to these definitions, the edit-distance between two trees is completely determined by the set of nodes that are not removed by the edit operations and, therefore, have a possible set of matches in the set M . In fact, the edit-distance between the trees t_1 and t_2 is given by

$$d(t_1, t_2) = \min_S \left[\sum_{v \in R_1} r_v + \sum_{u \in R_2} r_u + \sum_{(v,u) \in M} m_{vu} \right], \quad (1)$$

where r_u and r_v are the costs of removing nodes u and v , respectively, and m_{uv} is the cost of matching node u to node v . That is, the cost of relabeling nodes u and v so that the attributes or weights associated with the two nodes are the same.

This can be rewritten as

$$\begin{aligned} d(t_1, t_2) &= \min_S \left[\sum_{v \in R_1} r_v + \sum_{u \in R_2} r_u + \sum_{(v,u) \in M} m_{vu} \right] \\ &= \min_S \left[\sum_{v \in V_1} r_v + \sum_{u \in V_2} r_u \right. \\ &\quad \left. - \sum_{(v,u) \in M} (r_v + r_u - m_{vu}) \right]. \end{aligned} \quad (2)$$

Since $\sum_{v \in V_1} r_v$ and $\sum_{u \in V_2} r_u$ are constant and independent from S , the edit-distance is completely determined by the set of matches that maximize the utility

$$\mathcal{U}(M) = \sum_{(v,u) \in M} (r_v + r_u - m_{vu}). \quad (3)$$

Here we are interested in weighted graphs. Accordingly we assume that there is a weight w_i^t assigned to each node i of tree t . The cost of matching a node i of tree t_1 to a node j of tree t_2 is equal to the modulus of the difference in node weight, i.e. $|w_i^{t_1} - w_j^{t_2}|$. In this case the edit-distance is given by

$$\begin{aligned} d(t_1, t_2) &= \min_S \left[\sum_{v \in V_1} w_v^{t_1} + \sum_{u \in V_2} w_u^{t_2} \right. \\ &\quad \left. - 2 \sum_{(v,u) \in M} \min(w_v^{t_1}, w_u^{t_2}) \right] \end{aligned} \quad (4)$$

and the *utility* of the match M becomes

$$\mathcal{U}(M) = \sum_{(v,u) \in M} \min(w_v^{t_1}, w_u^{t_2}).$$

We would like to extend the concept of edit-distance to more than two trees so that we can compare a tree t^* to a set of trees T . Moreover, we would like to determine how a new sample relates to a previous distribution of tree structures. Formally, we would like to locate the match that minimizes the sum of the edit-distances between the new tree t^* and each tree $t \in T$, with the added constraint that if node a in the new tree t^* is matched to node b in a tree $t_1 \in T$ and to node c in another tree $t_2 \in T$, then b must be matched to c , i.e.

$$(a, b) \in M \wedge (a, c) \in M \implies (b, c) \in M, \quad (5)$$

where $M \subseteq V_1 \times V_2$ is the set of Cartesian pairs representing correspondence matches between the nodes of tree t_1 and those of tree t_2 . One way of locating the set of matches would be to search for the maximum substructure that can be obtained from any tree in a set by removing appropriate nodes. Unfortunately, by discarding unmatched nodes, we lose information concerning the structure of the pattern

space for the trees. The main problem is that the set of common nodes becomes marginal and we lose information about how the nodes distribute in the various structures. To use Bunke’s [27] analogy, the maximum common substructure gives us information about the mean of the set of trees, but it completely discards any information about how sample trees distribute around this mean. To overcome this limitation we can calculate a union of the nodes. This is a structure from which we can obtain any tree in our set by removing appropriate nodes, as opposed to the common substructure, which provides the intersection of the nodes.

3. Tree-union

In this section we describe our procedure for constructing the tree-union from a set of trees \mathcal{D} . The tree-union is a superstructure of the original trees, that is, a directed acyclic graph such that each tree $t \in \mathcal{D}$ can be obtained from it with a sequence of node removal operations. More formally, let $G = (V, E)$ be a superstructure of \mathcal{D} , for each $t \in \mathcal{D}$ there is an edit sequence \mathcal{S}_t , composed only of node relabeling and removal operations, such that $\mathcal{S}_t(G) \approx t$. The tree-union is a superstructure that minimizes the total distance

$$d_{\text{tot}} = \sum_{t \in \mathcal{D}} \text{cost}(\mathcal{S}_t). \quad (6)$$

We commence by noting that as with edit-distance, the edit-union of two trees is completely determined by the set of matched nodes. Hence, we can form the union by iteratively merging nodes that are matched. Upon completion of this procedure, the result is a directed acyclic graph with multiple paths connecting various nodes (see Fig. 1). This structure,

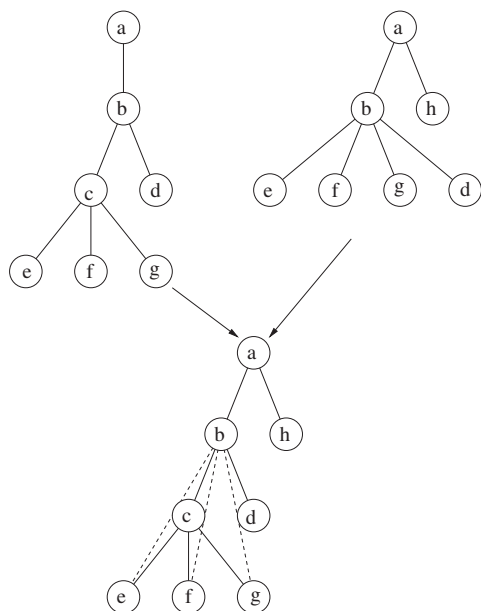


Fig. 1. Edit-union of two trees.

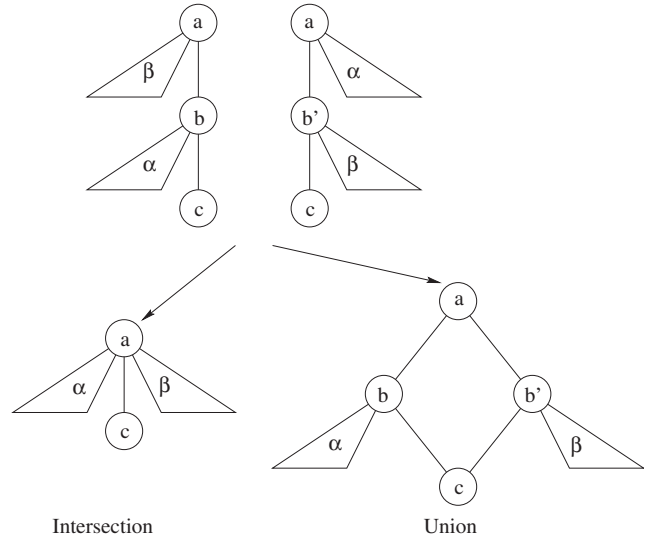


Fig. 2. Edit-union is not always a tree.

thus, has more links than necessary and in order to obtain the original trees using node removal operations alone, the superfluous edges need to be removed. If in Fig. 2 we eliminate the edges connecting node b to nodes e, f and g , we obtain a tree. Hence commencing from this tree, we can obtain either one of the original trees by node removal operations alone. Furthermore, the order relation defined by this tree and the one defined by the unaltered structure are identical.

We would hope that such a structure would always be a tree, so that we can use the matching technique already described to compare a tree to a group of trees. Unfortunately, it is not always possible to locate a tree such that we can edit it to obtain the original trees. An example is provided in Fig. 2. In this figure α and β are subtrees. Because of the constraints posed by matching subtree α to the subtree β , nodes b and b' cannot be matched and neither one can be a child of the other. The only alternative is to keep the two paths separate. In this way we can obtain the first tree by removing the node b' and the second tree by removing node b . Actually, removing the nodes is not sufficient. The reason for this is that shortcutting edges need to be removed. However, once again, the transitive closure of the union minus node b' is identical to the closure of the first tree.

3.1. Matching a tree to a union

As shown above, the union of two trees is, in general, a directed acyclic graph. Our approach can only match trees, and would fail on structures with multiple paths from one node a to a descendent node b , since it would count any match in the subtree rooted at b twice. Hence, we cannot directly use our approach to compare a tree to a tree-set or a distribution of trees.

Fortunately, we do not need to perform a match between two generic directed acyclic graphs. The reason for this is

that in an edit-union we have multiple paths between node a and node b , but each tree can have only one; hence multiple paths are mutually exclusive. If we constrain our search to matching nodes in one path only, we can match any tree to the union, being still guaranteed not to count the same subtree multiple times. Interestingly, this constraint can be merged with the *obtainability* constraint. We say that a match is *obtainable* if for each node v there cannot be two children a and b and a node c such that there is one path, possibly of length 0, from a to c and another from b to c . This constraint is reduced to the previously defined *obtainability* for trees when $c = b$, but it also makes it impossible for a and b to belong to two separate paths joining at c . Hence, from a node where multiple paths fork, we can extract matches from one path only.

We wish to locate the match consistent with the *obtainability* constraint that minimizes the sum of the edit-distances between the new tree and each tree in the set. For this purpose we can maximize the sum of the utilities

$$\mathcal{U}(M) = \sum_{t \in T} \sum_{(u,v) \in M, u \in V_t} (r_u^t + r_v^{t'} - m_{uv}^{tt'}). \quad (7)$$

Here $M \subset V_t \times \mathcal{N}^{\mathcal{T}}$ is the set of matches between the nodes V_t of the tree t and the nodes $\mathcal{N}^{\mathcal{T}}$ of the union structure of the set of trees T . Furthermore, r_u^t is the cost of removing node u from tree t , $r_v^{t'}$ is the cost of removing node v from t' , and $m_{uv}^{tt'}$ is the cost of matching node u of t to node v of t' . When the edit-distance is uniform (all weights are unity), the utility produced by a single match is equal to the number of trees that have an instance of that node (see Fig. 3). On the other hand, when we use

the weights associated to the nodes of the trees, we have $r_u^t + r_v^{t'} - m_{uv}^{tt'} = w_u + w_v^{t'} - |w_u - w_v^{t'}| = \min(w_u, w_v^{t'})$, where w_u is the weight associated to node u and $w_v^{t'}$ is the weight associated to node v in t' . By solving the modified weighted clique problems, we obtain the correspondence between the nodes in the new tree and the nodes in each tree in the set. Moreover, the edit-distance obtained is the sum of the distances from the new tree to each tree in the set T .

It is worth noting that this approach can be extended to match two union structures, as long as no more than one of them has multiple paths to a node. To do this we iterate through each pair of weights drawn from the two sets. To do this, we define the utility as

$$\mathcal{U}(M) = \sum_{t \in T_1, t' \in T_2} \sum_{(i,j) \in M} (r_u^t + r_v^{t'} - m_{uv}^{tt'}), \quad (8)$$

where $M \subset \mathcal{N}^{\mathcal{T}_1} \times \mathcal{N}^{\mathcal{T}_2}$ is the set of matches between the nodes of the union structures \mathcal{T}_1 and \mathcal{T}_2 , r_u^t is the cost of removing node v from tree t , and $m_{uv}^{tt'}$ is the cost of matching node u of tree t in \mathcal{T}_1 to node v of t' in \mathcal{T}_2 . The requirement that no more than one union has multiple paths to a node is necessary to avoid double counting.

3.2. Joining multiple trees

In the previous section we have seen that it is possible to construct the edit-union of a set of trees, and we have considered how a tree can be compared to this superstructure. We now wish to show how to construct such a structure. Locating the super-structure that minimizes the total distance between the trees in a set is computationally infeasible, but we propose a suboptimal iterative approach which at each iteration extends the union by adding a new tree to it. This is done by matching the tree to the union and then using the matched nodes to construct the union of the two structures. That is, we select a new tree t^* and we match it against the current union $\mathcal{F}^{(t)}$, to obtain the updated union $\mathcal{F}^{(t+1)}$. We proceed in this way until we have joined every tree.

In order to increase the accuracy of the approximation, we wish to merge the trees with the smaller distance first. The reason for this is that the smaller the distance between two trees, the higher is our confidence regarding the extracted correspondences. We start with the full set of trees, merge them and replace them with the union. We reiterate this procedure until we obtain a single structure. At each iteration we select two trees t_1 and t_2 such that the distance $d(t_1, t_2)$ is minimal, merge the two tree and reinsert the union structure $\mathcal{F}_{1,2}$ in the set of trees to be merged. Unfortunately, since we have no guarantee that the edit-union is a tree, we might attempt to merge two graphs with multiple paths to a node, and this is something that our matching algorithm cannot cope with. For this reason, we discard any union that is not a tree and attempt to merge the next-best match. When no trees can be merged without duplicating paths, we randomly select one union and merge the remaining structures to it in

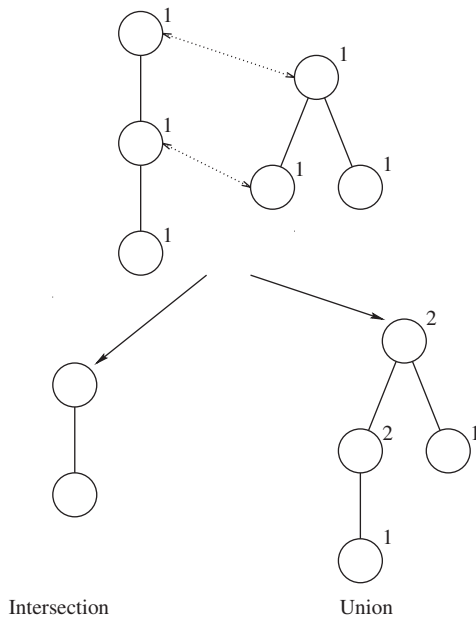


Fig. 3. The weight of a node in the union account for every node mapped to that node.

random order. In this way we are guaranteed to merge at most one multi-path structure at each step.

4. Pattern spaces from union trees

The tree-union provides a route to embedding trees of the same class in a pattern space. To do this we use the correspondences with the union tree to map each tree onto a pattern vector. The components of the vector are unity if the corresponding sample has a corresponding node, and zero otherwise. We perform principal components analysis for the sample trees assigned to each class. To do this we compute the covariance matrix for the pattern vectors, and project the pattern vectors onto the space spanned by the leading eigenvectors of the covariance matrix.

We place the nodes of the union tree \mathcal{T} in any arbitrary order. To each sample tree t we associate a pattern vector $\vec{x}_t = (x_1, \dots, x_n)^T \in \mathbb{R}^n$, where n is the number of nodes in the super-tree model \mathcal{T} . The component $x_t(i)$ of vector \vec{x} is

$$x_t(i) = \begin{cases} w_i^t & \text{if the tree has a node mapped to the} \\ & \text{ith node of the sample,} \\ 0 & \text{otherwise.} \end{cases}$$

In other words, we associate a pattern vector \vec{x}_t with the sample tree whose components are equal to the weight of the corresponding node in the union tree, if the node is present, and are zero otherwise. For the union tree \mathcal{T} we compute the mean pattern vector

$$\vec{\hat{x}} = \frac{1}{|T|} \sum_{t \in T} \vec{x}_t$$

and covariance matrix

$$\Sigma = \frac{1}{|T|} \sum_{t \in T} (\vec{x}_t - \vec{\hat{x}})(\vec{x}_t - \vec{\hat{x}})^T.$$

Suppose that the unit eigenvectors of the covariance matrix (ordered to decreasing eigenvalue) are $\vec{e}_1, \vec{e}_2, \dots, \vec{e}_n$. The leading l_{sig} eigenvectors are used to form the columns of the matrix $E = (\vec{e}_1 | \vec{e}_2 | \dots | \vec{e}_{l_{sig}})$. We perform PCA on the sample trees by projecting the pattern vectors onto the leading eigenvectors of the covariance matrix. The projection of the pattern vector for the sample tree indexed t is $\vec{y}_t = E^T \vec{x}_t$. The distance between the vectors in this space is $D^{PCA}(t, t') = (\vec{y}_t - \vec{y}_{t'})^T (\vec{y}_t - \vec{y}_{t'})$.

5. Experimental results

We evaluate the application of the new graph embedding approach on the problem of shock-tree matching. The shock-tree is a graph-based representation of the differential structure of the boundary of a 2D shape. It is obtained by locating the shape skeleton, and examining the differential behavior of the radius of the bitangent circle from the skeleton to the

object boundary, as the skeleton is traversed [28]. The so-called shocks distinguish between the cases where the local bitangent circle has maximum radius, minimum radius, constant radius or a radius that is strictly increasing or decreasing. We abstract the skeletons as trees in which the level in the tree is determined by their time of formation [22,28]. The later the time of formation, and hence their proximity to the center of the shape, the higher the shock in the hierarchy. While this temporal notion of relevance can work well with isolated shocks (maxima and minima of the radius function), it fails on monotonically increasing or decreasing shock groups. To give an example, a protrusion that ends on a vertex will always have the earliest time of creation, regardless of its relative relevance to the shape. To overcome this drawback, we augment the structural information given by the skeleton topology and the relative time of shock formation, with a measure of feature importance. We opt to use a shape-measure based on the rate of change of boundary length with distance along the skeleton as outlined in Ref. [29]. The nodes in our trees are the skeletal branches which are assigned a weight which is the length of the boundaries associated with the relevant skeletal branch. The edges indicate connectivity of the skeletal branches, while the level of a node in the tree is determined by the time of formation of the corresponding skeletal branch. At the top level of the structure is a “dummy” root node which does not correspond to any branch, but rather to the barycenter of the shape.

We compare the embedding obtained using the union approach with 2D multi-dimensional scaling of the pairwise edit-distances. Multi-dimensional scaling is a well-known statistical technique for visualizing data which exists in the form of pairwise similarities rather than ordinal values [30]. Stated simply, the method involves embedding the objects associated with the pairwise distances in a low-dimensional space. This is done by performing principal components analysis on a centered and sphered matrix of pairwise similarities, and projecting the original objects into the resulting eigenspace. The objects are visualized by displaying their positions in the space spanned by the leading eigenvectors. The method has been widely exploited for data analysis in the psychology literature. A comprehensive review can be found in the recent book by Cox and Cox [30]. Then we compare the clusters obtained using the L_1 norm defined on the union with those obtained using pairwise edit-distances. The clusters are extracted using the algorithm presented in Ref. [14].

5.1. Example pattern spaces

Fig. 4 displays the first two principal components of the sample-tree distribution for the embedding spaces extracted from six shape classes and the corresponding multi-dimensional scaling of the edit-distances of the same shock-trees. In most cases the union approach appears to create a tightly packed central cluster with a few shapes

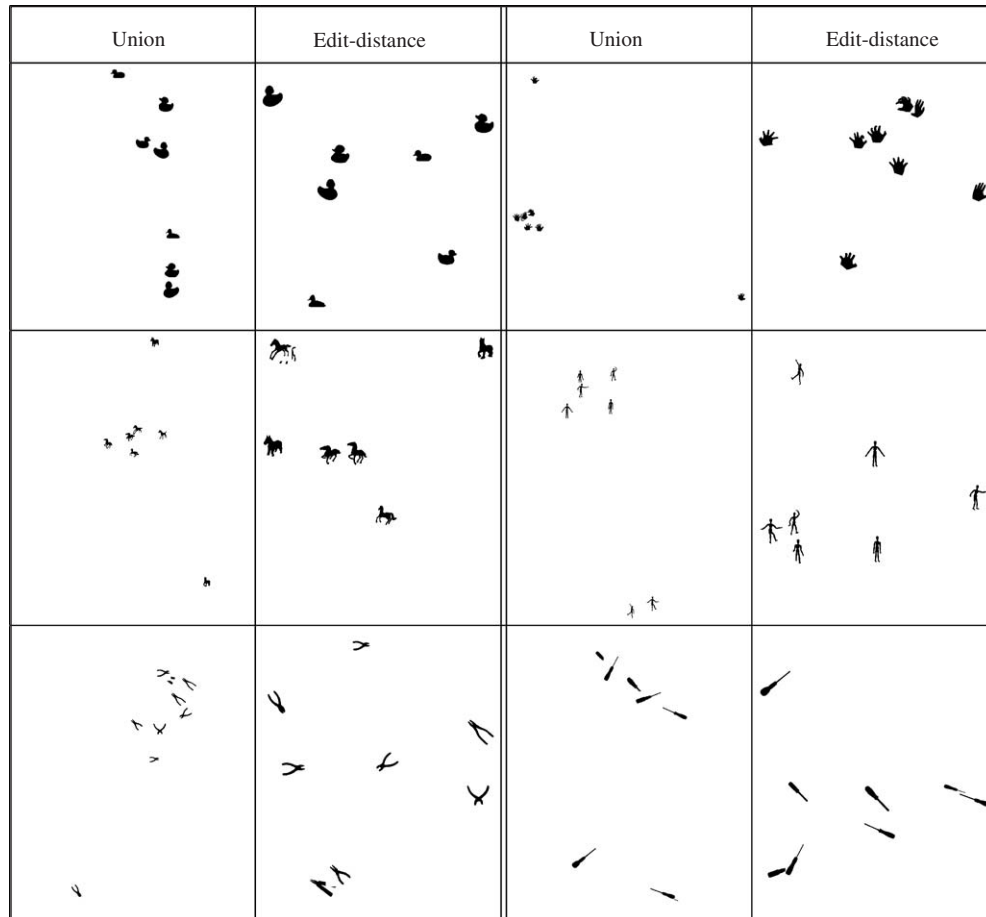


Fig. 4. Comparison of shape embedding through edit-union and edit-distance. Single class case.

scattered further away than the rest. This separation is linked to substantial variations in the structure of the shock-trees. For example, in the shape-space formed by the class of pliers the outlier is the only pair-of-pliers with the nose closed. In the case of shape-space for the horse-class, the outliers appear to be the cart-horses while the inliers are the ponies. On the other hand, multi-dimensional scaling of the edit-distances appears more compact, since it does not separate outliers as strongly. The spatial distribution obtained is not linked to any evident property of the shape.

By using the embedding to visualise multiple classes, the difference in quality of the embedding is more evident. Fig. 5 shows a clear example where the embedding obtained through edit-union is better than that obtained through multi-dimensional scaling of the pair-wise distances. In this case the pairwise distance algorithm consistently underestimates the distance between shapes belonging to different clusters. This is a general problem of pairwise matching. The method works very well when the shapes are close and the extracted correspondence is reliable, but as the shapes move further apart the advantage the correct correspondence has over alternative ones diminishes, until, eventually, another match is selected, which reports a lower distance. The result of this

is a consistent underestimation of the distance as the shapes move farther apart in shape-space. Figs. 6 and 7 show examples where the distance in shape-space is not large enough to allow us to observe the described behavior, yet the embedding obtained through union fares well against the embedding obtained through multi-dimensional scaling of the pairwise edit-distances. In particular, Fig. 7 shows a better ordering of the shapes, with brushes being so tightly packed that they overlap. It is interesting to note how the union embedding places the monkey wrench (at the top) somewhere in between the pliers and the wrenches. The algorithm is able to consistently match the head to the heads of the wrenches, and the handles to the handles of the pliers. This fact shows that, by forcing the matches to be consistent across trees, we enable the embedding to better capture the structural information present in the trees, yielding better spatial distribution than that obtained with multi-dimensional scaling.

We now perform a more quantitative evaluation of the method. Fig. 8 plots the distances obtained through edit-union of weighted shock-trees (x -axis) versus the corresponding pairwise edit-distances (y -axis). The solid line in the plot corresponds to the case when the two distances take on the same value. Most of the points fall below the

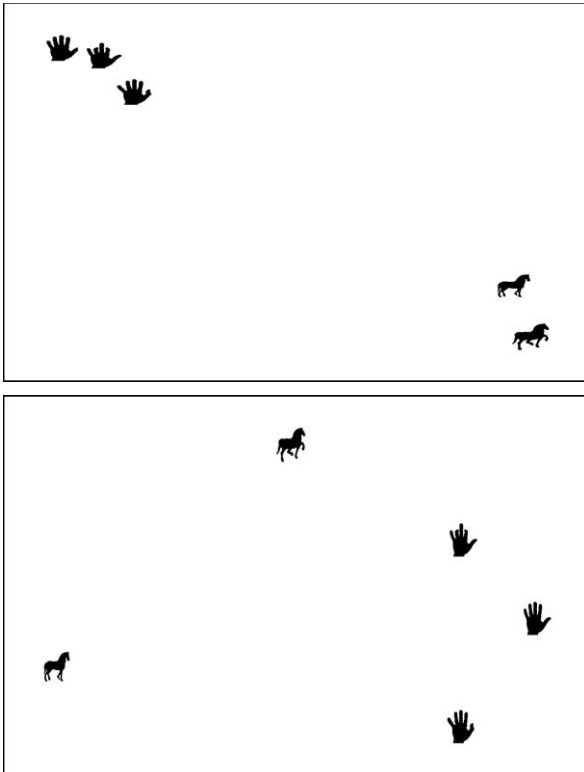


Fig. 5. Top: embedding through union. Bottom: multi-dimensional scaling of pairwise distances.

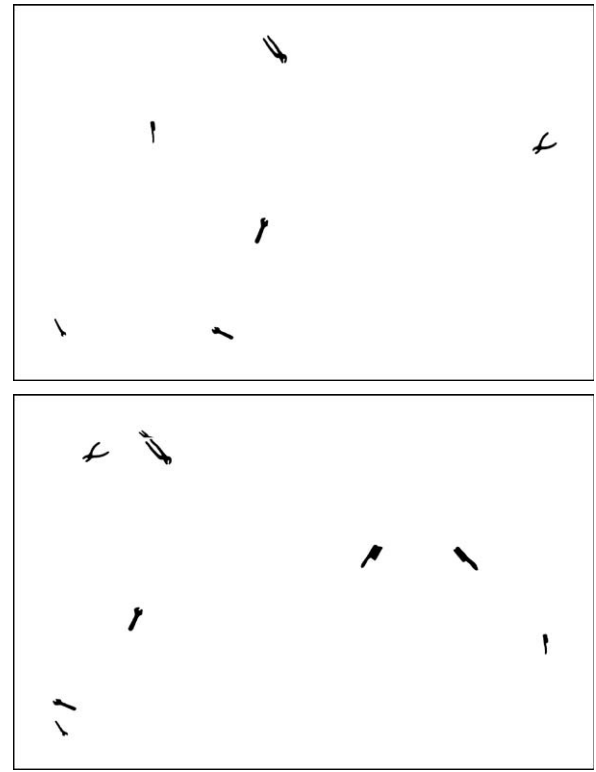


Fig. 7. Top: embedding through union. Bottom: multi-dimensional scaling of pairwise distances.

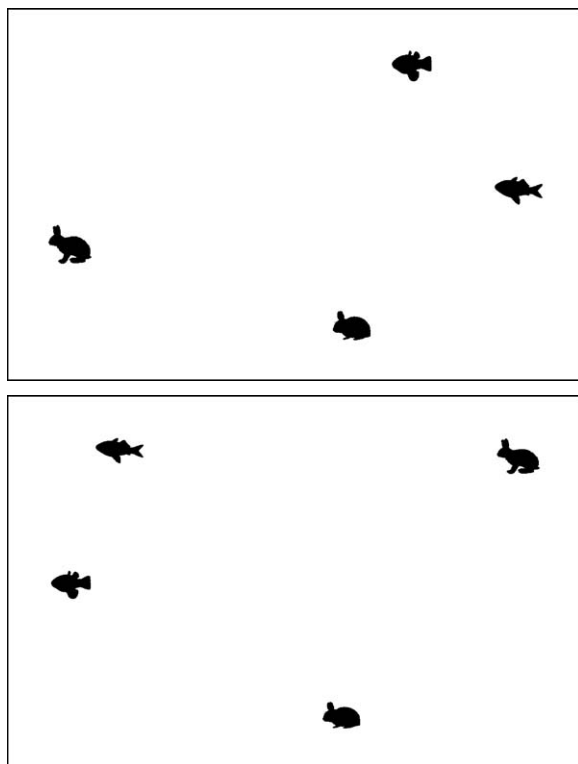


Fig. 6. Top: embedding through union. Bottom: multi-dimensional scaling of pairwise distances.

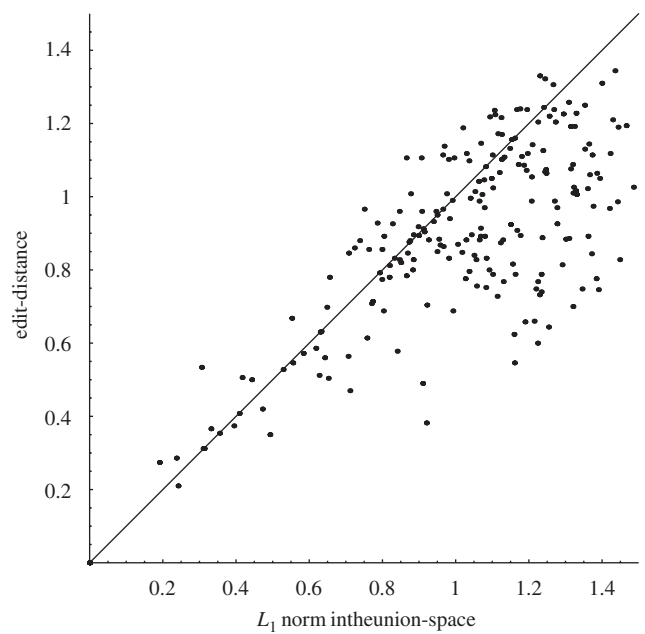


Fig. 8. Edit-union versus pairwise distances.

line. Hence, the plot demonstrates that the pairwise edit-distance approach tends to underestimate the distances between shapes.

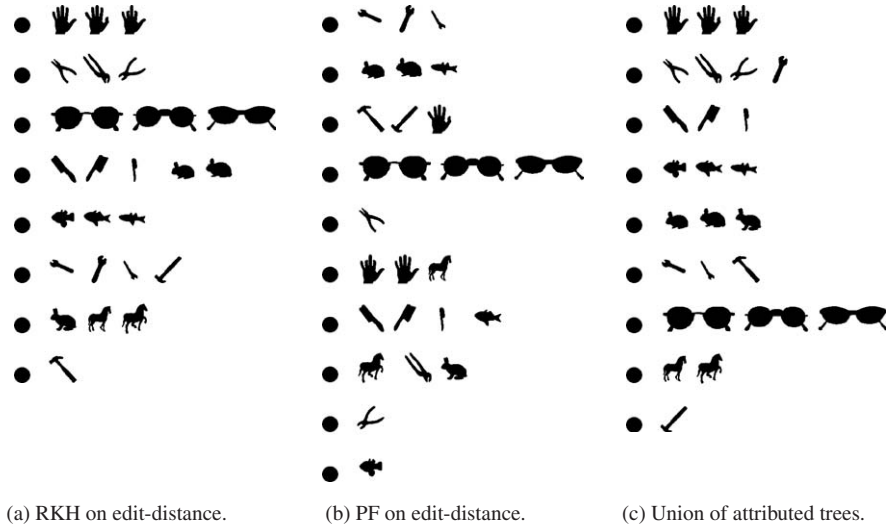


Fig. 9. Clusters extracted from edit-distances versus those obtained from the L_1 norm defined on the union: (a) RKH on edit-distance; (b) PF on edit-distance; and (c) Union of attributed trees.

We now turn our attention to the problem of clustering the shapes into classes using the computed distances. Here we compare the L_1 norm defined on the tree-union to two distance measures. The first of these is the approximate edit-distance described in Ref. [26], while the second is a tree distance metric that can be computed in polynomial time [31]. The distance metric between trees t_1 and t_2 is computed using the function $\sigma(u, v)$ that gauges the similarity between node u in t and node v in t' . The similarity measure for the two trees is $W(t, t') = \max_{\phi} \sum_{(u,v) \in \phi} \sigma(u, v)$, where ϕ is a subtree isomorphism between t and t' . Hence, the distance metric is defined as:

$$D^{metric}(t, t') = 1 - \frac{W(t, t')}{\max(|t|, |t'|)}.$$

Additionally, we explore the use of two graph-spectral algorithms for clustering the different distance measures. The first is the pairwise clustering method described in Ref. [14] (RHK). This is a graph spectral method that assigns objects to clusters using the leading eigenvector of an affinity matrix found by taking the negative exponential of the distance between shapes. The second is the matrix factorization algorithm developed by Perona and Freeman in Ref. [32] (PF).

In Fig. 9 we compare the results of applying the clustering methods to the weighted tree edit-distances and the distances obtained through edit-union. Here the shapes collected together after the bullet points correspond to distinct clusters. Fig. 10 shows the results obtained by clustering the metric described in Ref. [31].

In order to assess the quality of the groupings, we have used two well-known cluster-validation measures [33]. The first is the standard classification rate. To compute the measure, for each cluster we note the predominant shape class.



Fig. 10. Clusters extracted from D^{metric} : (a) RKH algorithm; and (b) PF algorithm.

Those graphs assigned to the cluster which do not belong to the predominant shape class are deemed to be misclassified. The classification rate is fraction of graphs belonging to the predominant cluster shape classes divided by the total number of graphs. This measure exhibits a well-known bias towards large number of classes. To overcome this we also used the Rand index. The Rand index is defined as $R_I = A/(A + D)$. Here A is the number of “agreements”, that is the number of pairs of graphs that belong to the same class and that are assigned to the same cluster, and D is the number of “disagreements”, that is, the number of pairs of graphs that belong to different shape classes and

Table 1
Validity measures of the extracted clusters

	Classification rate (%)	Rand index (%)
RKH on edit-distance	84	88
PF on edit-distance	76	85.23
Union	92	97.54
RHK on D^{metric}	68	83.08
PF on D^{metric}	84	86.46

that are assigned to different clusters. The index is hence the fraction of graphs of a particular shape class that are closer to an graph of the same class than to one of another class.

Table 1 lists the classification rate and the Rand index for the extracted clusters. When applied to a database of 25 shapes, edit-union clearly outperforms the pairwise distance approach. Unfortunately the union approach does not scale, and fails as the database becomes larger and the number of shape classes present increases.

5.2. Synthetic data

To augment these real-world experiments, we have computed the union using synthetic data. The aim of the experiments is to characterize the ability of the approach to generate an embedding space for tree structures. To meet this goal we have randomly generated some prototype trees and, from each tree, we have generated five or 10 structurally perturbed copies. The procedure for generating the random trees is as follows. We commenced with an empty tree (i.e. one with no nodes) and we iteratively added the required number of nodes. At each iteration nodes were added as children of one of the existing nodes. The parent was randomly selected with uniform probability from among the existing nodes. The weight of the newly added nodes was selected at random from an exponential distribution with mean 1. This procedure tends to generate trees in which the branch ratio is highest closest to the root. This is quite realistic in real-world situations, since shock-trees tend to have this property. To perturb the trees we simply added nodes using the same approach.

In our experiments the size of the prototype trees varied from five to 20 nodes. As we can see from Fig. 11, the algorithm was able to clearly separate the clusters of trees generated by the same prototype. Fig. 11 shows three experiments with synthetic data. The top and middle images are produced by embedding five structurally perturbed trees per prototype. Hence, trees 1 to 5 are perturbed copies of the first prototype, 6 to 10 of the second. The bottom image shows the result of the experiment with 10 structurally perturbed trees per prototype. Hence, trees 1 to 10 belong to one cluster and trees 11 to 20 to the other. In each image the clusters are well separated.

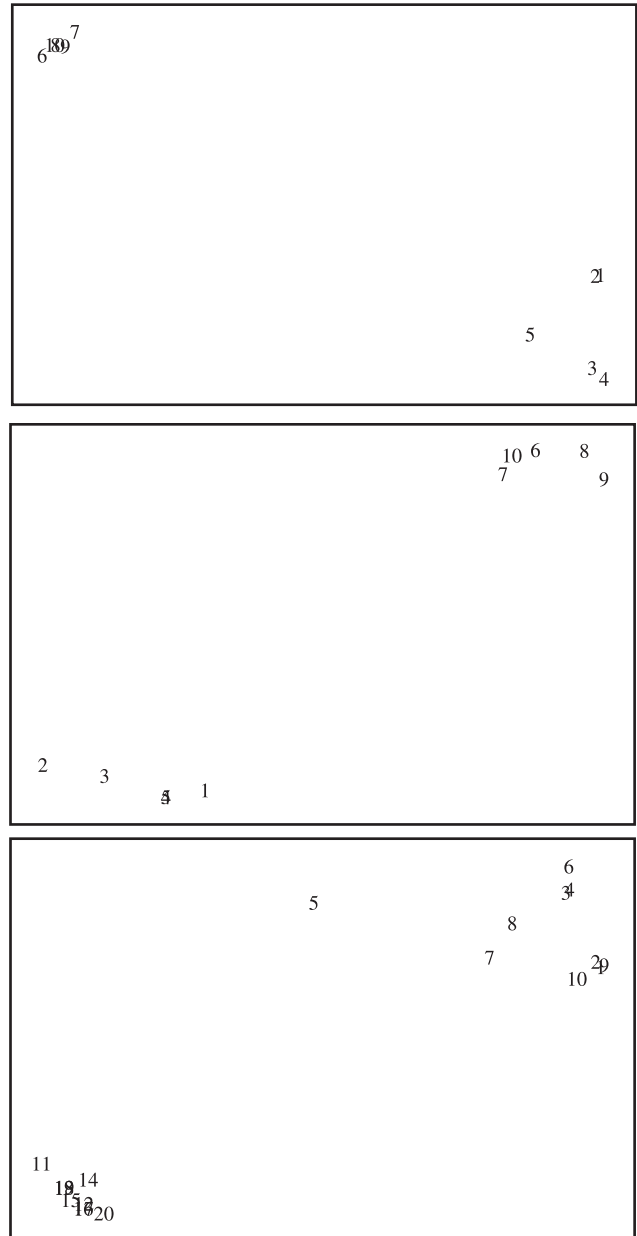


Fig. 11. Synthetic clusters.

6. Conclusions

In this paper we investigated a technique to extend the tree edit-distance framework to allow for the simultaneous matching of multiple tree structures. With this approach we can impose a consistency of node correspondences between matches, avoiding the underestimation of the distance typical of pairwise edit-distance approaches. Furthermore, through this method we obtain a “natural” embedding space of tree structures that can be used to analyze how tree representations vary in our problem domain.

In a set of experiments we apply this algorithm to match shock graphs. The results of these experiments are very

encouraging, as they show that the algorithm is able to group similar shapes together in the generated embedding space. Yet the approach fails when confronted with larger databases containing several shape classes. This is due to the fact that the union is capable of capturing the structural variation present in a class, but has problems with multiple classes. There is a clear direction in which this work can be extended. We can cast the matching problem as one of learning the structural representations that best describe a set of trees and use the union is used to various single classes. This extension has been recently presented in Ref. [34].

References

- [1] I.T. Jolliffe, *Principal Components Analysis*, Springer, Berlin, 1986.
- [2] S. Roweis, L. Saul, Non-linear dimensionality reduction by locally linear embedding, *Science* 299 (2002) 2323–2326.
- [3] J.B. Tenenbaum, V.D. Silva, J.C. Langford, A global geometric framework for non-linear dimensionality reduction, *Science* 290 (2000) 586–591.
- [4] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, *Neural Comput.* 15 (6) (2003) 1373–1396.
- [5] S. Rizzi, Genetic operators for hierarchical graph clustering, *Pattern Recognition Lett.* 19 (1998) 1293–1300.
- [6] J. Segen, Learning graph models of shape, in: J. Laird (Ed.), *Proceedings of the Fifth International Conference on Machine Learning*, 1988, pp. 29–25.
- [7] A. Munger, H. Bunke, X. Jiang, Combinatorial search vs. genetic algorithms: a case study based on the generalized median graph problem, *Pattern Recognition Lett.* 20 (1999) 1271–1279.
- [8] A.K.C. Wong, J. Constant, M.L. You, *Random Graphs, Syntactic and Structural Pattern Recognition*, World Scientific, Singapore, 1990.
- [9] A.D. Bagdanov, M. Worring, First order Gaussian graphs for efficient structure classification, *Pattern Recognition* 36 (2003) 1311–1324.
- [10] D. Heckerman, D. Geiger, D.M. Chickering, Learning Bayesian networks: the combination of knowledge and statistical data, *Mach. Learn.* 20 (1995) 197–243.
- [11] M.A. Lozano, F. Escolano, ACM attributed graph clustering for learning classes of images, in: *Graph Based Representations in Pattern Recognition*, Lecture Notes in Computer Science, vol. 2726, 2003, pp. 247–258.
- [12] H. Bunke, et al., Graph clustering using the weighted minimum common supergraph, in: *Graph Based Representations in Pattern Recognition*, Lecture Notes in Computer Science, vol. 2726, 2003, pp. 235–246.
- [13] B.J. Jain, F. Wysotzki, Central clustering of attributed graphs, *Mach. Learn.* 56 (2004) 169–207.
- [14] B. Luo, A. Torsello, A. RoblesKelly, R.C. Wilson, E.R. Hancock, A probabilistic framework for graph clustering, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001, pp. 912–919.
- [15] X. Bai, E.R. Hancock, Heat kernels manifolds and graph embedding, in: *Structural Syntactic and Statistical Pattern Recognition*, Lecture Notes in Computer Science, vol. 3138, Springer, Berlin, 2004, pp. 198–206.
- [16] J. Shi, J. Malik, Normalized cuts and image segmentation, *IEEE Trans. Pattern Anal. Mach. Intell.* 22 (8) (2000) 888–905.
- [17] R. Kannan, et al., On clusterings: good, bad and spectral, *FOCS* 41, pp. 367–377.
- [18] M. Pavan, M. Pelillo, Dominant sets and hierarchical clustering, in: *Proceedings of the Ninth IEEE International Conference on Computer Vision*, vol. I, 2003, pp. 362–369.
- [19] B. Luo, R.C. Wilson, E.R. Hancock, Eigenspaces for graphs, *Int. J. Image Graph.* 2 (2002) 247–268.
- [20] B. Luo, R.C. Wilson, E.R. Hancock, Spectral embedding of graphs, *Pattern Recognition* 36 (2003) 2213–2223.
- [21] R.C. Wilson, B. Luo, E.R. Hancock, Pattern vectors from algebraic graph theory, *IEEE Trans. Pattern Anal. Mach. Intell.* 27 (7) (2005) 1112–1124.
- [22] A. Shokoufandeh, S. Dickinson, K. Siddiqi, S. Zucker, Indexing using a spectral coding of topological structure, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1999, pp. 491–497.
- [23] K. Sengupta, K.L. Boyer, Organizing large structural modelbases, *IEEE Trans. Pattern Anal. Mach. Intell.* 17 (4) (1995) 321–332.
- [24] T.B. Sebastian, P.N. Klein, B.B. Kimia, Recognition of shapes by editing their shock graphs, *IEEE Trans. Pattern Anal. Mach. Intell.* 26 (5) (2004) 550–571.
- [25] K. Zhang, R. Statman, D. Shasha, On the editing distance between unordered labeled trees, *Inform. Process. Lett.* 42 (3) (1992) 133–139.
- [26] A. Torsello, E.R. Hancock, Efficiently computing weighted tree edit distance using relaxation labeling, in: *Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2001, pp. 438–453.
- [27] H. Bunke, A. Kandel, Mean and maximum common subgraph of two graphs, *Pattern Recognition Lett.* 21 (2000) 163–168.
- [28] K. Siddiqi, A. Shokoufandeh, S.J. Dickinson, S.W. Zucker, Shock graphs and shape matching, *Int. J. Comput. Vision* 35 (1999) 13–32.
- [29] A. Torsello, E.R. Hancock, A skeletal measure of 2D shape similarity, *Comput. Vision Image Understanding* 95 (1) (2004) 1–29.
- [30] T. Cox, M. Cox, *Multidimensional Scaling*, Chapman & Hall, London, 1994.
- [31] A. Torsello, D. Hidović-Rowe, M. Pelillo, Polynomial-time metrics for attributed trees, *IEEE Trans. Pattern Anal. Mach. Intell.* 27 (7) (2005) 1087–1099.
- [32] P. Perona, W.T. Freeman, A factorization approach to grouping, in: *European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science, vol. 1406, 1998, pp. 655–670.
- [33] A.K. Jain, R.C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
- [34] A. Torsello, E.R. Hancock, Learning shape-classes using a mixture of tree-unions, *IEEE Trans. Pattern Anal. Mach. Intell.* 28 (6) (2006) 954–967.
- [35] K. Zhang, D. Shasha, Simple fast algorithms for the editing distance between trees and related problems, *SIAM J. Comput.* 18 (1989) 1245–1262.

About the Author—ANDREA TORSELLO received the “Laurea” degree with honors in computer science from Ca’ Foscari University of Venice, Italy, in 1997. In 2004 he received his PhD in computer science at the University of York, UK. Currently he is an Assistant Professor at “Ca’ Foscari” University of Venice, Italy.

His research interests are in the area of computer vision and pattern recognition, in particular, the application of stochastic and structural approaches to shape analysis. Recently, he co-edited a special issue of *Pattern Recognition* on “Similarity-based pattern recognition.”

About the Author—EDWIN HANCOCK studied physics as an undergraduate at the University of Durham and graduated with honours in 1977. He remained at Durham to complete a PhD in the area of high-energy physics in 1981. Following this he worked for 10 years as a researcher in the fields of high-energy nuclear physics and pattern recognition at the Rutherford-Appleton Laboratory (now the Central Research Laboratory of the Research

Councils). During this period he also held adjunct teaching posts at the University of Surrey and the Open University. In 1991 he moved to the University of York as a lecturer in the Department of Computer Science. He was promoted to Senior Lecturer in 1997 and to Reader in 1998. In 1998 he was appointed to a Chair in Computer Vision.

Professor Hancock now leads a group of some 15 faculty, research staff and PhD students working in the areas of computer vision and pattern recognition. His main research interests are in the use of optimization and probabilistic methods for high and intermediate level vision. He is also interested in the methodology of structural and statistical pattern recognition. He is currently working on graph-matching, shape-from-X, image data-bases and statistical learning theory. His work has found applications in areas such as radar terrain analysis, seismic section analysis, remote sensing and medical imaging. Professor Hancock has published some 90 journal papers and 350 refereed conference publications. He was awarded the Pattern Recognition Society medal in 1991 and an outstanding paper award in 1997 by the journal *Pattern Recognition*. In 1998 he became a fellow of the International Association for Pattern Recognition.

Professor Hancock has been a member of the Editorial Boards of the journals *IEEE Transactions on Pattern Analysis and Machine Intelligence*, and, *Pattern Recognition*. He has also been a guest editor for special editions of the journals *Image and Vision Computing* and *Pattern Recognition*. He has been on the programme committees for numerous national and international meetings. In 1997 with Marcello Pelillo, he established a new series of international meetings on energy minimization methods in computer vision and pattern recognition.