International Journal of Computer Vision 72(3), 259–285, 2007

DOI: 10.1007/s11263-006-8929-y

Discovering Shape Classes using Tree Edit-Distance and Pairwise Clustering

ANDREA TORSELLO
Dipartimento di Informatica, University "Ca' Foscari" of Venice,
Via Torino 155, 30172 Venezia Mestre, Italy
torsello@dsi.unive.it

ANTONIO ROBLES-KELLY
NICTA, RSISE Bldg. 115, Australian National University, Canberra, ACT 0200, Australia*

EDWIN R. HANCOCK
Department of Computer Science, University of York, York Y01 5DD, UK
erh@cs.york.ac.uk

Received July 29, 2005; Revised April 19, 2006; Accepted April 19, 2006

First online version published in June, 2006

Abstract. This paper describes work aimed at the unsupervised learning of shape-classes from shock trees. We commence by considering how to compute the edit distance between weighted trees. We show how to transform the tree edit distance problem into a series of maximum weight clique problems, and show how to use relaxation labeling to find an approximate solution. This allows us to compute a set of pairwise distances between graph-structures. We show how the edit distances can be used to compute a matrix of pairwise affinities using χ^2 statistics. We present a maximum likelihood method for clustering the graphs by iteratively updating the elements of the affinity matrix. This involves interleaved steps for updating the affinity matrix using an eigendecomposition method and updating the cluster membership indicators. We illustrate the new tree clustering framework on shock-graphs extracted from the silhouettes of 2D shapes.

Keywords: shock trees, shape recognition, pairwise clustering, edit-distance

1. Introduction

The analysis of skeletal abstractions of 2D shapes has been a topic of sustained activity for over 30 years in the computer vision literature. Some of the earliest work in the area drew its inspiration from biometrics where it has lead to the study of the so-called Blum skeleton (Blum, 1973). Recently, there has been a renewed research interest in the topic which as been aimed at deriving a richer description of the differential structure of the object boundary. This literature has focused on the so-called shock-structure of the reaction-diffusion equation for object boundaries.

The idea of characterizing boundary shape using the differential singularities of the reaction equation was first introduced into the computer vision literature by Kimia et al. (1995). The idea is to evolve the boundary of an object to a canonical skeletal form using the reaction-diffusion equation. The skeleton represents the singularities in the curve evolution, where inward

*National ICT Australia is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

moving boundaries collide. The reaction component of the boundary motion corresponds to morphological erosion of the boundary, while the diffusion component introduces curvature dependent boundary smoothing. In practice, the skeleton can be computed in a number of ways (Arcelli and di Baja, 1992; Ogniewicz, 1994). Recently, Siddiqi et al. have shown how the eikonal equation which underpins the reaction-diffusion analysis can be solved using the Hamilton-Jacobi formalism of classical mechanics (Bouix and Siddiqi, 2000; Siddiqi et al., 1999a).

With the skeleton to hand, then the next step is to devise ways of using it to characterize the shape of the original object boundary. Most of the approaches reported in the literature opt to use a structural characterization. For instance, Zucker, Siddiqi and others have labeled points on the skeleton using so-called shock-labels (Siddiqi et al., 1999b). According to this taxonomy of local differential structure, there are different classes associated with the behavior of the radius of the bitangent circle from the skeleton to the nearest pair of boundary points. The so-called shocks distinguish between the cases where the local bitangent circle has maximum radius, minimum radius, constant radius or a radius which is strictly increasing or decreasing. Kimia and Giblin opt for a simpler representation which is based just on the junctions and terminations of the skeleton (Giblin and Kimia, 1999).

Once the skeletal representation is to hand then shapes may be matched by comparing their skeletons. Most of the work reported in the literature adopts a structural approach to the matching problem. For instance, Pelillo, Siddiqi et al. (1999) use a sub-tree matching method. Tirthapura, Kimia and Klein have a potentially more robust method which matches by minimizing graph-edit distance (Klein et al., 1999; Tirthapura et al., 1998). However, this approach requires the order of the branches to be the same for all the observed shapes. This makes the method robust against random deformation, but not against shape articulation or variations of 3D viewpoint.

Stability with respect to change in viewpoint can be achieved using aspect-graph based approaches (Bowyer and Dyer, 1990; Dickinson et al., 1992). For instance, Cyr and Kimia (2004) compute edit-distance to a set of prototypical views of the object to be recognized. With the aspect-graph representation the matching problem becomes one of view selection. Denton et al. (2004a, b) select canonical views by merging similar features in a set of images of the same object. To achieve stability against articulation requires the that calculation of edit-distance allows for a potential change in branch order.

The observation underpinning this paper is that although considerable effort has gone into the extraction and matching of shock trees, the topic of how to use shock trees to learn shape-classes has not received significant attention. The aim in this paper is therefore to develop a framework for learning shape classes by clustering shock trees. We pose the problem as one of graph-clustering. We make two contributions. First, we show how to compute approximate distances between weighted trees using relaxation labeling. Second, we develop a pairwise clustering algorithm algorithm that can be applied to the set of distances between trees.

1.1. Graph Clustering

Graph clustering is an important, yet relatively under-researched topic in machine learning (Rizzi, 1998; Segen, 1988). The importance of the topic stems from the fact that it can be used as a tool for learning the class-structure of data abstracted in terms of relational graphs. Problems of this sort are posed by a multitude of unsupervised learning tasks in knowledge engineering, pattern recognition and computer vision. The process can be used to structure large data-bases of relational models (Sengupta and Boyer, 1988) or to learn equivalence classes. One of the reasons for limited progress in this area has been the lack of algorithms suitable for clustering relational structures. Broadly speaking, there are two approaches to the problem.

The first of these is to use central clustering methods. Here a class prototype is maintained, and the aim is to find clusters of graphs that are close to the prototype. This prototype may be either an explicit representation of full graph-structure or a proxy representation based on features that succinctly express the graph characteristics. The learning of explicit cluster prototypes is a difficult task. For instance, in related work we have developed a greedy algorithm that minimizes a description length criterion (Torsello and Hancock, 2006). Lozano and Escolano (2003) use the EM algorithm to learn the prototype. However, these algorithms are computationally intensive and have hence only been demonstrated on rather small graphs. Moreover, one of the difficulties is that of defining what is meant by the mean or representative graph for each cluster. However, Munger et al. (1999) have recently taken some important steps in

this direction by developing a genetic algorithm for searching for median graphs. This problem may also be overcome by using a proxy representation of the prototype. For instance, Luo et al. (2003) have used vectors of graph-spectral features, including the vector of leading eigenvectors of the adjacency matrix, for this purpose.

The second approach to the problem is to pose it as one of pairwise clustering. This avoids the need for a class prototype, and requires only that a set of pairwise distances between graphs be supplied. The clusters are located by identifying sets of graphs that have strong mutual pairwise affinities. There is therefore no need to explicitly identify an representative (mean, mode or median) graph for each cluster. Unfortunately, the literature on pairwise clustering is much less developed than that on central clustering. However, one of the most powerful pairwise clustering algorithm is that of Hofmann and Buhmann (1997) which use mean field theory to iteratively perform pairwise clustering. Shi and Malik (2000) have developed a graph spectral method that uses the Fiedler vector to find clusters that minimize a normalized cut measure. Pavan and Pelillo have recently introduced a new graph-based measure of pairwise cluster affinity based on dominant sets (Pavan and Pelillo, 2003a) and have developed a hierarchical clustering method that uses the measure to cluster graphs (Pavan and Pelillo, 2003b).

When posed in a pairwise setting, the graph-clustering problem requires two computational ingredients. The first of these is a distance measure between relational structures. We address this issue in more detail in the next subsection. The second ingredient is a means of performing pairwise clustering on the distance measure. There are several possible routes available. The simplest is to transform the problem into a central clustering problem. For instance, it is possible to embed the set of pairwise distances in a Euclidean space using a technique such as multi-dimensional scaling and to apply central clustering to the resulting embedding. The second approach is to use a graph-based method (Sengupta and Boyer, 1988) to induce a classification tree on the data. Finally, there are mean-field methods which can be used to iteratively compute cluster-membership weights (Hofmann and Buhmann, 1997). These methods require that the number of pairwise clusters be known *a priori*.

1.2. Tree Edit Distance

To apply the graph clustering to shock-trees, we require a means of comparing their pairwise similarity. The problem of how to measure the similarity of pictorial information which has been abstracted using graph-structures has been the focus of sustained research activity for over twenty years in the computer vision literature. Early work on the topic included Barrow and Burstall's idea (1976) of locating matches by searching for maximum common subgraphs using the association graph, and the extension of the concept of string edit distance to graph-matching by Fu and et al. (Eshera and Fu, 1986). The idea behind edit distance (Tsai and Fu, 1979) 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. 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.

Unfortunately, the task of calculating edit distance is a computationally hard problem and most early efforts can be regarded as being goal-directed. However, in an important series of recent papers, Bunke has demonstrated the intimate relationship between the size of the maximum common subgraph and the edit distance (Bunke and Kandel, 2000). In particular, he showed that, under certain assumptions concerning the edit-costs, computing the maximum common subgraph (MCS) and the graph edit distance are equivalent. The restriction imposed on the edit-costs is that the deletions and re-insertions of nodes and edges are no more expensive than the corresponding node or edge relabeling operations. In other words, there is no incentive to use relabeling operations, and as a result the edit operations can be reduced to those of insertion and deletion.

While trees are a special case of graphs, because of the connectivity and partial order constraints which apply to them, the methods used to compare and match them require significant specific adaptation. Consequently, Bunke's result (2000) linking the computation of edit distance to the size of the maximum common subgraph does not translate in a simple way to trees.

1.3. Contribution and Paper Outline

To develop a framework for tree-clustering, we make two contributions in this paper.

The first of these is to develop an energy minimization method for efficiently computing the weighted tree edit distance. We use the graph-theoretic notion of tree closure to show how the problem of locating the maximum edited common subtree can be transformed into that of searching for maximal cliques of an association graph. We follow Bomze et al. (2000) and use a variant of the Motzkin Straus theorem to convert the maximum weighted clique problem into a quadratic programming problem which can be solved by relaxation labeling. By re-casting the search for the maximum common subgraph as a *max-clique* problem (Barrow and Burstall, 1976), then we can extend Bunke's work (2000) from graphs, and efficiently compute tree edit distance.

The second contribution is to develop a maximum likelihood framework for graph clustering. The problem is posed as one of pairwise clustering which is parameterized using two sets of indicator variables. The first of these are cluster membership variables which indicate to which cluster a graph belongs. The second set of variables are affinity weights which convey the strength of the similarity relations between pairs of graphs belonging to the same cluster. Our clustering algorithm is an iterative one in which both sets of indicator variables are updated so as to maximize a likelihood criterion.

The outline of this paper is as follows. In Section 2 we outline our algorithm for computing weighted tree edit distance. Section 3 presents the clustering algorithm. In Section 4 we provide experiments on shock graphs. Finally, Section 5 presents conclusions and suggests directions for further investigation.

2. Tree Edit Distance

The problem we aim to solve is the automatic extraction of correspondences between tree representations. Formally, given two trees $t_1 = (V_1, E_1)$ and $t_2 = (V_2, E_2)$, where V_1 and V_2 are set of nodes and E_1 and E_2 set of edges, we wish to find the edit-distance between the two trees. That is, 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.
- *node relabel:* this operation changes the weight of a node.

Since a node insertion on the data tree is dual to a node removal on the model tree, we can reduce the number of operations to be performed to only node removal and node relabeling, as long as we perform the operations on both trees. Clearly, the cost of removing a node in the model tree must be equal to the cost of inserting it in the data tree. We assign a cost r_v to the operation of removing node v and a cost m_{vu} to matching node v to node u (that is, the minimum cost of relabeling nodes v and v to a common label).

With these definitions, the edit-distance between trees t_1 and t_2 is:

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

where $\mathcal{S}$ is sequence of edit operations, R_1 and R_2 are the sets of nodes of t_1 and t_2, respectively, that are removed by $\mathcal{S}$, and $M \in V_1 \times V_2$ is the set of matches between nodes of t_1 and t_2 induced by $\mathcal{S}$.

The edit-distance approach is general in the sense that, by applying different costs to the edit operations, it can be equally applied to unattributed trees and to trees with either symbolic or continuous-valued attributes. In particular, to solve the correspondence problem for unattributed trees, we set $r_v = 1$ and $m_{vu} = 0$ for each node v and u of the two trees. On the other hand, later in the paper, to solve the correspondence problem for shock trees attributed with the weight described in Torsello and Hancock (2004), we set $r_v = w_v$ and $m_{vu} = |w_v - w_u|$, where w_v is the weight assigned to node v. Obviously, other cost assignments are possible.

It is easy to see that the cost of the edit-sequence is completely determined by the nodes in the two trees that are matched to one-another. In fact, given the optimal edit-sequence $\mathcal{S}$, we have:

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

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

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

The edit-distance is related to the maximum value of the utility $\max_M \mathcal{U}(M)$ in the following manner

$$d(t_1, t_2) = \sum_{v\in R_1} r_v + \sum_{u\in R_2} r_u - \max_M[\mathcal{U}(M)],$$

where the maximization is over all the matches that are compatible with the hierarchical constraints imposed by the trees.

At this point we introduce the concept of an *edited isomorphism*. Let us assume that we have two trees $t_1 = (V_1, E_1)$ and $t_2 = (V_2, E_2)$. Furthermore, let $t' = (V', E')$ be a tree that can be obtained from both t_1 and t_2 with node removal and relabel operations. The correspondences M_1 and M_2 between the nodes of t' and the nodes of t_1 and t_2, respectively, will induce an isomorphism $M' = M_1^{-1} \circ M_2$ between nodes in t_1 and t_2. This isomorphism places two nodes in correspondence with each other if and only if they are mapped to the same node in t'. This isomorphism is referred to as an edited isomorphism induced by t'. Furthermore, we say that the isomorphism induced by this tree is a *maximum edited isomorphism* if it maximizes the total utility $\mathcal{U}(M')$. Clearly, finding the maximum edited isomorphism is equivalent to solving the tree edit-distance problem.

In the remainder of this section, we present the steps necessary for computing the maximum edited isomorphism using relaxation labeling. We commence by explaining the relationship between the maximum common subtree and the association graph, in the case of exact tree matching. We show how to extend this to the case of inexact tree matching by exploiting the equivalence of transitive closures and node removal operations on trees. Next, we show how to decompose the process into a series of maximum weighted clique finding subproblems. Finally, we map the max-weighted clique problems onto a relaxation labeling process.

2.1. Association Graphs and the Maximum Common Subtree

To commence, we describe a polynomial-time algorithm for the subtree isomorphism problem. This allows us to formalize some concepts and provide a starting point to extend the approach to the minimum tree edit-distance problem.

Let $G = (V, E)$ be a graph, where V is the set of nodes (or vertices) and $E \subseteq V \times V$ is the set of directed edges. With the notation $v \rightsquigarrow u$ we shall mean that there is a directed edge going from node v to node u. If there is a directed path from v to u, we shall write $v \dashrightarrow u$. Hierarchical trees have a canonical order relation $\mathcal{O}$ induced by paths: given two nodes v and u, we have $(v, u) \in \mathcal{O} \Leftrightarrow v \dashrightarrow u$. That is, two nodes are in the canonical relation if and only if there is a path connecting them. This relation can be shown to be an (irreflexive) order relation.

The phase-space we use to represent the matching of nodes is the directed association graph. This is a variant of the association graph, a structure that is frequently used in graph matching problems (Barrow and Burstall, 1976; Pelillo et al., 1999). The association graph $G_A = (V_A, E_A)$ of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ has node set $V_A = V_1 \times V_2$ equal to the Cartesian products of nodes of the graphs to be matched. Hence, each node represents a possible association, or match, of a node in one graph to a node in the other. The edges represent the pairwise constraints of the problem. In particular, they represent both connectivity on the original graphs and the feasibility of a solution having both associations linked by an edge. The use of directed arcs in the association graph allows us to make use of the order provided by the tree hierarchies. For the exact isomorphism problem (maximum common subgraph) the edges of the association graph $G_A = (V_1 \times V_2, E_A)$ of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are:

$$(v, v') \rightsquigarrow (u, u') \quad \text{iff } v \rightsquigarrow u \text{ and } v' \rightsquigarrow u', \tag{4}$$

where $v, u \in V_1$ are nodes of graph G_1 and $v', u' \in V_2$ are nodes of graph G_2. The graph obtained can be shown to be still be ordered. Specifically, an association graph for the tree isomorphism problem can be shown to be a forest.

Proposition 1. *The directed association graph of two directed acyclic graphs (DAGs) G and G' is acyclic.*

Proof: Let us assume that $(u_1, v_1) \rightsquigarrow \cdots \rightsquigarrow (u_n, v_n)$ is a cycle. Then, since an arc $(v, v') \rightsquigarrow (u, u')$ in the association graph exists only if the arcs $v \rightsquigarrow u$ and $v' \rightsquigarrow u'$ exist in G and G' respectively, we have that

$u_1 \rightsquigarrow \cdots \rightsquigarrow u_n$ is a cycle in G and $v_1 \rightsquigarrow \cdots \rightsquigarrow v_n$ is a cycle in G' against the hypothesis that they are DAGs. □

Proposition 2. *The directed association graph of two trees t and t' is a forest.*

Proof: Since we already know that the association graph is a DAG, we have to show that for each node (u, u') there is at most one node (v, v') such that $(v, v') \rightsquigarrow (u, u')$. Due to the way in which the association graph is constructed this means that either u or u' must have at most one incoming edge. However, both t and t' are trees, so in consequence both u and u' have at most one incoming edge, namely the one that originates from the parent. □

The directed association graph can be used to reduce a tree matching problem into subproblems using a divide-and-conquer approach. We call the maximum (weight) common subtree rooted at (v, v') a solution to the maximum (weight) common subtree problem applied to two subtrees of t and t'. In particular, the solution is constrained to the subtrees of t and t' rooted at v and v' respectively. This solution is further constrained by the condition that v and v' are roots of the matched subtrees.

With the maximum rooted common subtree problem for each child of (v, v') at hand, the maximum isomorphism rooted at (v, v') can be reduced to a maximum weight bipartite match problem between the set D of the children of v and the set D' of the children of v'.

To proceed, let $B = (D \cup D', E)$ be a bipartite graph with partitions D and D' and $w : E \to \mathbb{R}$ be a weight function on the edges of B. A bipartite match is a set of non-adjacent edges of B. The maximum weight bipartite match is the set of non-adjacent edges with maximum total weight. The search for a maximum weight bipartite match is a well known linear programming problem with several very efficient polynomial time algorithms to solve it (Papadimitriou and Steiglitz, 1982).

The two partitions V and V' of the bipartite match consist of the children of v and v' respectively. The weight of the match between $u \in V$ and $u' \in V'$ is the sum of the matched weights of the maximum isomorphism rooted at (u, u'). In the case of an un-weighted tree this is the cardinality of the isomorphism. This structure provides us with a one-to-one relationship between matches in the bipartite graph and the children of (v, v') in the association graph. The solution of the bipartite matching problem identifies a set of children of (v, v') that satisfy the constraint of matching one node of t to no more than one node of t'. Furthermore, among such sets is the one that guarantees the maximum total weight of the isomorphism rooted at (v, v'). In Reyner (1977) a similar approach is applied to the subtree problem.

The maximum isomorphism between t and t' is a maximum isomorphism rooted at (v, v'), where either v or v' is the root of t or t' respectively. This reduces the isomorphism problem to $n+m$ rooted isomorphism problems, where n and m are the cardinalities of t and t'. Furthermore, since there are $n \times m$ nodes in the association graph, the problem is reduced to a set of $n \times m$ maximum bipartite matching problems, each of which can be solved with known polynomial time algorithms. In what follows, we will extend this approach to the minimum weighted tree-edit problem and present an evolutionary method to conquer the subproblems.

2.2. *Inexact Tree Matching: Node Removal and Transitive Closure on Trees*

We would like to extend the algorithm described in the previous section to develop an error-tolerant method for locating tree isomorphisms. As noted earlier, there is a strong connection between the computation of the maximum common subtree and the tree edit-distance. Bunke and Kandel (2000) showed that, under certain constraints applied to the edit-costs, locating the maximum common subgraph problem and computing the minimum graph edit-distance are computationally equivalent to one another.

This is not directly true for trees, because of the added constraint that a tree must be connected. However, extending the concept to the common edited subtree, we can use common substructures to find the minimum-cost edited tree-isomorphism.

Given a tree $t = (V, E)$, we define the *closure* $\Omega(t) = (V, E_\Omega)$ to be a directed acyclic graph with the same node set and with edges satisfying

$$u \rightsquigarrow v \text{ in } \Omega(t) \iff u \dashrightarrow v \text{ in } t. \tag{5}$$

Clearly, t and $\Omega(t)$ are subject to the same order relation $\mathcal{O}$ between their nodes. Furthermore, we have $u \rightsquigarrow v$ in $\Omega(t) \iff u \mathcal{O} v$, i.e. the edge set of $\Omega(t)$ is a complete description of $\mathcal{O}$.

For each node v of t, we can define an edit operation E_v on the tree and an edit operation $\mathcal{E}_v$ on the closure $\Omega(t)$ of the tree t (see Fig. 1). In both cases the edit

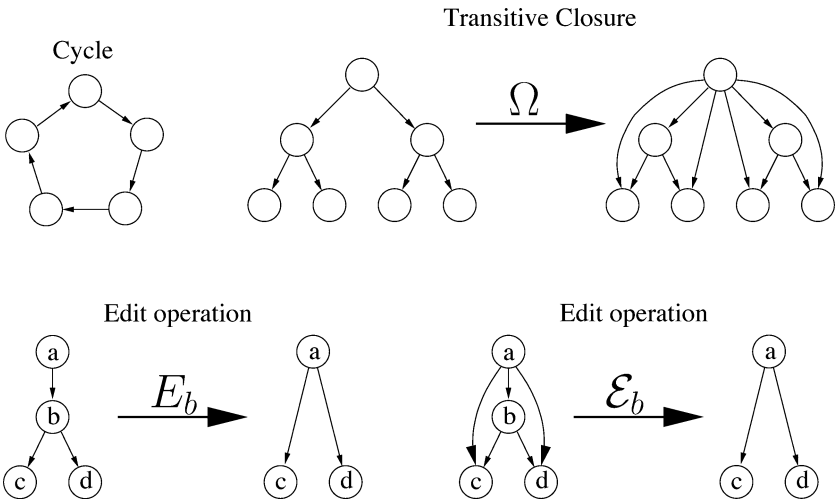


Figure 1. Terminology on directed graphs.

operation removes the node v, all the incoming edges, and all the outgoing edges.

In a previous work (Torsello and Hancock, 2003), we have shown that the transitive closure operation and the node removal operation commute. That is, we have

Lemma 1. $\mathcal{E}_v(\Omega(t)) = \Omega(E_v(t))$.

Furthermore, the transitive closure operation commutes with node relabeling as well, since one acts only on weights and the other acts only on node connectivity.

To take our discussion one step further, we need to define the concept of obtainability. To commence, we say that two nodes a and b of tree t are *independent* if there is no path from a to b or from b to a, that is if neither is a descendent of the other in t. A subtree s of $\Omega(t)$ is said to be *obtainable* if for each node v of s there cannot be two children a and b so that the edge (a, b) is in $\Omega(t)$. In other words, s is *obtainable* if and only if every pair of siblings in s are independent.

These definitions lead us to the following theorem which is also proved in Torsello and Hancock (2003):

Theorem 1. *A tree $\hat{t}$ can be obtained from a tree t with an edit sequence composed of only node removal and node relabeling operations if and only if $\hat{t}$ is an* obtainable *subtree of* $\Omega(t)$.

By virtue of Theorem 1, every common edited isomorphism between tree t and tree t' induces a consistent subtree of both $\Omega(t)$ and $\Omega(t')$. Since minimizing the edit-cost and maximizing the utility are equivalent, the set of correspondences of the minimum-cost edited tree-isomorphism can be found by searching for the consistent subtree with maximum utility. As a consequence, finding a minimum-cost edit-sequence is equivalent to finding a maximum utility *common obtainable subtree* of $\Omega(t)$ and $\Omega(t')$.

2.3. Cliques and Common Obtainable Subtrees

In this section we show that the directed association graph allows us to decompose the tree matching task into a series of subproblems, that can be solved using a divide-and-conquer strategy. Given two trees t and t' to be matched, we create the directed association graph of the transitive closures $\Omega(t)$ and $\Omega(t')$ and we search for an *obtainable* matching tree in the graph. That is, we seek a tree in the graph that corresponds to two *obtainable* trees in the transitive closures $\Omega(t)$ and $\Omega(t')$. Any such tree having maximum utility induces the optimal set of node-correspondence between t and t'.

By analogy to what we did for the exact matching case, we divide the problem into a *maximum common obtainable subtree* rooted at (v, w), for each node (v, w) of the association graph. We show that, given the utility of the maximum common consistent subtree rooted at each child of (v, w) in the association graph, we can transform the rooted subtree problem into a *maximum weighted clique problem*. A *clique* of a graph $G = (V, E)$ is a complete, or fully connected, subgraph

of G. A *maximum (unweighted) clique* is a clique with maximum node cardinality among all cliques of G, while a *maximum weighted clique* of a weighted graph G is a clique with maximum total weight among all cliques of G. The search for a clique with maximum weight is a well-known NP-hard problem. Solving this problem for each node in the association graph and searching for the one with maximum utility, we can find the solution to the minimum-cost edit-sequence problem and hence find the edit-distance.

Let us assume that we know the utility of the subtree for every child of the node (v, w) in the association graph. We wish to find the set of *independent* siblings with greatest total utility. Let us construct an undirected graph whose nodes consist of the children of (v, w) in the association graph. We connect the two nodes (p, q) and (r, s) if and only if p and r are *independent* in t, and q and s are *independent* in t'. Furthermore, we assign to each association node (a, b) a weight equal to the utility of the *maximum common obtainable subtree* rooted at (a, b). The maximum weight clique of this graph will be the set of mutually independent siblings with maximum total weight. Let $W^{(u,v)}$ be the weight of this clique. The utility of the *maximum common obtainable subtree* rooted at (v, w) will be

$$U^{(u,w)} = W^{(u,v)} + r_v + r_w - m_{vw}, \tag{6}$$

where r_v and r_w are the costs of removing nodes v and w respectively, while m_{vw} is the cost of matching v to w. Furthermore, the nodes of the clique will be the children of (v, w) in the maximum common consistent subtree. With the set of rooted utilities to hand, the expression for the tree edit distance becomes

$$d(t_1, t_2) = \sum_{v \in R_1} r_v + \sum_{u \in R_2} r_u - \max_{u \in V_1} \max_{w \in V_2} U^{(u,w)}$$

2.4. Heuristics for Maximum Weighted Clique

We have transformed an inexact tree-matching problem into a series of maximum weighted clique problems. That is, we transformed one NP-hard problem into multiple NP-hard problems. The observation underlying this approach is the fact that there are a large number of approaches and very powerful heuristics exist to solve or approximate the *max-clique* problem. Furthermore, since the seminal paper by Barrow and Burstall (1976), transforming matching problems into max-clique problems has become a standard technique.

The method adopted here to solve each instance of the maximum weight clique problem is an evolutionary one introduced by Bomze et al. (2000). This method is based on a continuous formulation of the discrete combinatorial problem. This is achieved by transforming the discrete problem into one of symmetric quadratic programming. This approach has proven to be powerful, and is competitive in terms of performance with the best clique finding algorithms in the literature (Pelillo, 1999; Pelillo and Torsello, 2006).

In 1965, Motzkin and Straus (1965) showed that the (unweighted) maximum clique problem can be reduced to a quadratic programming problem on the n-dimensional simplex $\Delta = \{\mathbf{x} \in \mathbb{R}^n | x_i \geq 0 \text{ for all } i = 1 \ldots n, \sum_i x_i = 1\}$, where x_i are the components of vector $\mathbf{x}$. More precisely, let $G = (V, E)$ be a graph where V is the node set and E is the edge set, and let $C \subseteq V$ be a maximum clique of G, then the vector $\mathbf{x}^* = \{x_i^* = 1/\#C \text{ if } i \in C,\ 0 \text{ otherwise}\}$ maximizes in Δ the function $g(\mathbf{x}) = \mathbf{x}^T \mathcal{A}_G \mathbf{x}$, where $\mathcal{A}_G$ is the adjacency matrix of G. Furthermore, given a set $\mathcal{S} \subseteq V$, we define the *characteristic* vector $\mathbf{x}^{\mathcal{S}}$ has components

$$x_i^{\mathcal{S}} = \begin{cases} 1/\#\mathcal{S} & \text{if } i \in \mathcal{S} \\ 0 & \text{otherwise.} \end{cases}$$

With this definition, $\mathcal{S}$ is a maximum (maximal) clique if and only if $g(\mathbf{x}^{\mathcal{S}})$ is a global (local) maximum for the function g.

Gibbons et al. (1997) generalized this result to the weighted clique case. In their formulation the association graph is substituted with a matrix $\bar{\mathcal{A}}_G = (\bar{a}_{ij})_{i,j \in V}$ related to the weights and connectivity of the graph by the relation

$$\bar{a}_{ij} = \begin{cases} 1/w_i & \text{if } i = j \\ k_{ij} \geq \dfrac{\bar{a}_{ii} + \bar{a}_{jj}}{2} & \text{if } (i, j) \notin E \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Let us consider a weighted graph $G = (V, E, w)$, where V is the set of nodes, E the set of edges, and $w : V \to \mathbb{R}$ a weight function that assigns a weight to each node. Gibbons et al. proved that, given a set $\mathcal{S} \subseteq V$ and its *characteristic* vector $\mathbf{x}^{\mathcal{S}}$ defined as

$$x_i^{\mathcal{S}} = \begin{cases} \dfrac{w(i)}{\sum_{j \in \mathcal{S}} w(j)} & \text{if } i \in \mathcal{S}, \\ 0 & \text{otherwise,} \end{cases}$$

then $\mathcal{S}$ is a maximum (maximal) weight clique only if $\mathbf{x}^{\mathcal{S}}$ is a global (local) minimizer for the quadratic

form $\mathbf{x}^T \bar{A}_G \mathbf{x}$. Furthermore, the weight of the clique $\mathcal{S}$ is $w(\mathcal{S}) = \frac{1}{\mathbf{x}^{S^T} \bar{A}_G \mathbf{x}^S}$.

Unfortunately, under this formulation, the minima are not necessarily isolated. As a result, when we have more than one clique with the same maximal weight, any convex linear combinations of their characteristic vectors will give the same maximal value. This implies that, if we find a minimizer $\mathbf{x}^*$ we can derive the weight of the clique. However, we might not be able to determine the nodes that constitute the clique.

To overcome this problem, Bomze et al. (2000) introduce a regularization factor to the quadratic programming method that generates an equivalent problem with isolated solutions. The new quadratic program minimizes $\mathbf{x}^T C \mathbf{x}$ in the simplex, where the matrix $C = (c_{ij})_{i,j \in V}$ is defined as

$$c_{ij} = \begin{cases} \dfrac{1}{2w_i} & \text{if } i = j \\ k_{ij} \geq c_{ii} + c_{jj} & \text{if } (i, j) \notin E, i \neq j \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

With this regularization factor, $\mathcal{S}$ is a maximum (maximal) weighted clique if and only if $\mathbf{x}^{\mathcal{S}}$ is a global (local) minimizer for the quadratic program.

To solve the quadratic problem we transform it into the equivalent problem of maximizing $\mathbf{x}^T(\gamma \mathbf{e}\mathbf{e}^T - C)\mathbf{x}$, where $\mathbf{e} = (1, \ldots, 1)^T$ is the vector with every component equal to 1 and γ is a positive scaling constant.

To approximate the quadratic programming problem, we use relaxation labeling. Relaxation labeling is an evidence-combining process developed in the framework of constraint satisfaction. Its goal is to find a classification that assigns a label from a set $\Lambda = \{\lambda_1, \ldots, \lambda_m\}$ to a set of objects $O = \{o_1, \ldots, o_n\}$ that satisfies pairwise constraints and interactions between the objects and labels. The discrete assignment space is relaxed into a probability space $\Theta = (\Delta_m)^n$, where Δ_m is an m-dimensional simplex. Given a relaxed assignment $\mathbf{p}$, $p_i(\lambda)$ represents the probability that object $o_i \in O$ is classified with label $\lambda \in \Lambda$. The constraints to the possible assignments are given in the form of mutual compatibility between pair of assignments. We indicate with $R_{ij}(\lambda, \mu)$ the degree of compatibility of assigning label λ to object o_i, given that object o_j is labeled μ. A relaxation labeling process takes as input the initial labeling assignment $\mathbf{p}^0$ and iteratively updates it taking into account the compatibility model. The evolution of the assignment is determined by the update rule

$$p_i^{t+1}(\lambda) = \frac{p_i^t(\lambda) q_i^t(\lambda)}{\sum_\mu p_i^t(\mu) q_i^t(\mu)}, \tag{9}$$

where the compatibility coefficient is $q_i(\lambda) = \sum_{j=1}^n \sum_{\mu=1}^m R_{ij}(\lambda, \mu) p_j(\mu)$.

Pelillo (1997) showed that, when the compatibilities are symmetric, that is $R_{ij}(\lambda, \mu) = R_{ji}(\mu, \lambda)$, the function $\mathcal{A}(\mathbf{p}) = \sum_{i,\lambda} p_i(\lambda) q_i(\lambda)$ is a Lyapunov function for the process, i.e. $\mathcal{A}(\mathbf{p}^{t+1}) \geq \mathcal{A}(\mathbf{p}^t)$, with equality if and only if $\mathbf{p}^t$ is a stationary point. Therefore, this process can be used to find local optima of a quadratic programming problem defined on Θ. By setting the number of objects equal to 1, the quadratic problem solved by the relaxation labeling process is

$$\begin{aligned} &\max \sum_\lambda \sum_\mu p(\lambda) R(\lambda, \mu) p(\mu) \\ &\text{subject to } p \in \Delta_m. \end{aligned} \tag{10}$$

By setting $R(\lambda, \mu) = \gamma - c_{\lambda\mu}$, this problem is equivalent to that stated in Eq. (8) above. Therefore, relaxation labeling can be used to approximate the maximum weighted clique problem and hence the maximum set of *independent* children of nodes v and v'. Each label λ_i of the labeling problem is in relation with a node (u_i, u_i') which is a child of (v, v') in the association graph. Upon convergence, the non-zero components of $\mathbf{p}^\infty$ are in correspondence with nodes of the children of (v, v') that from a clique of *independent* siblings with maximal weight. That is, we find a set $\mathcal{S}$ such that

$$(u_i, u_i') \in \mathcal{S} \iff p^\infty(\lambda_i) > 0. \tag{11}$$

This set of correspondences is optimal subject to the fact that v is matched to v'. Hence, the weight of the match rooted at (v, v') is

$$W^{(v,v')} = m_{vv'} + \sum_{(u,u') \in \mathcal{S}} W^{(u,u')}. \tag{12}$$

In this way we can propagate the associations from the leaves of the directed association graph upwards, using the weight of the extracted cliques to initialize the compatibility matrix of every parent association. For a subproblem rooted at (u, v) the compatibility coefficients can be calculated knowing the weight of every isomorphism rooted at the descendants of u and v. Specifically, the compatibility coefficients between matches λ and μ which link node a to node a'

and node b to node b' respectively, are initialized as $R^{(u,v)}(\lambda, \mu) = \gamma - c^{(u,v)}_{(\lambda,\mu)}$, where

$$c^{(u,v)}_{(\lambda,\mu)} = \begin{cases} \dfrac{1}{2W^{(a,a')}} & \text{if } (a, a') = (b, b') \\ c^{(u,v)}_{(\lambda,\lambda)} + c^{(u,v)}_{(\mu,\mu)} & \text{if } (a, a') \text{ and } (b, b') \text{ are } \textit{independent} \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

We can now compute the approximate weighted tree edit distance using the formula

$$d(t_1, t_2) = \sum_{v \in R_1} r_v + \sum_{u \in R_2} r_u - \max_{u \in V_1} \max_{w \in V_2} U^{(u,w)}$$

where

$$U^{(u,w)} = W^{(u,v)} + r_v + r_w - m_{vw}$$

and the weight of the maximum clique is obtained from the label probabilities at convergence of the relaxation process using the formula

$$W^{(u,v)} = \frac{1}{(\mathbf{p}^\infty)^T \bar{A}_G \mathbf{p}^\infty}.$$

Once all the optimal associations have been calculated for all possible nodes of the association graph, we can determine the topmost match in the edited isomorphism by searching for the association with the maximum weight. Once the topmost match is found, we can construct the complete isomorphism by following the optimal matches obtained with relaxation labeling.

This optimization approach allows us to make use of problem specific information about the solutions. Usually, when solving the maximum clique problem, the label assignment probabilities are initialized with a uniform distribution so that the relaxation labeling process starts from a point close to the barycenter of the simplex. A problem with this approach is that the dimension of the basin of attraction of one maximal clique grows with the number of nodes in the clique, regardless of their weights. While in general there is a strong correlation between clique size and clique weight, this is not the case for the cliques generated by our algorithm. With the problem decomposition adopted here, the wider cliques tend to that map nodes to lower levels. Matches at higher level, on the other hand, yield smaller cliques that are likely to weigh more. As a result the solution will be biased towards matches that are very low on the graph, even if these matches require cutting a large number of nodes and are, thus, less likely to give an optimum solution. Due to the nature of our problem decomposition, matches higher up in the hierarchy are more likely than matches lower down. For this reason, we initialize the assignment probabilities as $p^0(\lambda_i) = \frac{\hat{p}(\lambda_i)}{\sum_j \hat{p}(\lambda_j)}$, where

$$\hat{p}(\lambda_i) = \exp[-k(\text{depth}(u_i) + \text{depth}(u_i'))]. \tag{14}$$

Here, k is a constant and $\text{depth}(u_i)$ is the relative depth in the original tree t of node u_i with respect to node v. A value of $k = 1.2$ was empirically found to yield good results on shock-graphs.

3. Pairwise Clustering

The process of pairwise clustering is somewhat different to the more familiar one of central clustering. Whereas central clustering aims to characterize cluster-membership using the cluster mean and variance, in pairwise clustering it is the relational similarity of pairs of objects which is used to establish cluster membership. Although less well studied than central clustering, there has recently been renewed interest in pairwise clustering aimed at placing the method on a more principled footing using techniques such as mean-field annealing (Hofmann and Buhmann, 1997), spectral graph theory (Shi and Malik, 2000), and relaxation labeling (Pavan and Pelillo, 2003a).

To commence, we require some formalism. We are interested in grouping a set of trees $\mathcal{G} = \{G_1, \ldots, G_{|K|}\}$ whose index set is K. The set of trees is characterized using a matrix of pairwise similarity weights. The elements of this weight matrix are computed using the approximate tree edit distance $d(t_i, t_j)$ between the shock trees t_i and t_j.

3.1. Similarity Matrix

In order to compute the matrix of similarities among the trees in $\mathcal{G}$, we adopt a picture of the graph clustering process in which the trees can be embedded in an n-dimensional space $\mathcal{R}^n$. Here we treat the embedding space as a latent representation. Hence we are not concerned with a specific embedding procedure. However, a number of concrete possibilities exist. For instance, features could be extracted from the graph adjacency structure and subjected to principal

components analysis, or the pattern of pairwise distances could be subjected to multi-dimensional scaling. In this way each graph would become a point in the embedding space. We assume that for each distinct cluster of trees the embedded position vectors follow a spherically symmetric Gaussian distribution. For the cluster with index ω, the covariance matrix is $\sigma_\omega I_n$ where n is the $n \times n$ identity matrix. Suppose that $x_{i\omega}$ and $x_{j\omega}$ represent the embedded position vectors for the trees G_i and G_j, and that the trees both belong to the cluster indexed ω. The difference in position between the trees, i.e. $x_{i\omega} - x_{j\omega}$ will be drawn from the normal distribution $\mathcal{N}(0, 4\sigma_\omega^2 I)$. As a result the distance measure

$$\left\| \frac{x_{i\omega} - x_{j\omega}}{2\sigma_\omega} \right\| = \frac{d(t_i, t_j)^2}{4\sigma_\omega^2} \approx \chi_n^2 \tag{15}$$

will follow a χ^2 distribution with n degrees of freedom.

Given a distance $d(t_i, t_j)$ between two points i and j, we can estimate the probability that the two points belong to the same cluster ω' using the χ^2 distribution, provided that we know the cluster variance $\sigma_{\omega'}^2$. The estimated probability is:

$$P\{i \in \omega' \text{ and } j \in \omega'\} = P\left\{ \chi_n^2 > \frac{d(t_i, t_j)^2}{4\sigma_{\omega'}^2} \right\}. \tag{16}$$

Using this simple model, we can define the similarity matrix A setting its coefficients $A_{i,j}$ to the probability that the trees i and j belong to the same cluster. In other words:

$$A_{i,j} = P\left\{ \chi_n^2 > \frac{d(t_i, t_j)^2}{4\sigma_{\omega'}^2} \right\}. \tag{17}$$

The element $A_{i,j}$ of the affinity weight matrix represents the strength of association between the trees i and j. We will work with affinity-weights which are constructed to fall in the interval [0, 1]. When the affinity weight is close to one, then there is a strong association between the pair of nodes; when it is close to zero then there is a weak association.

3.2. *Maximum Likelihood Clustering*

The aim in clustering is to partition the set of graphs into disjoint subsets. If T_ω represents one of these subsets and Ω is the index-set of different partitions (i.e. the different pairwise clusters), then $T = \bigcup_{\omega \in \Omega} T_\omega$ and $T_{\omega'} \cap T_{\omega''} = \emptyset$ if $\omega' \neq \omega''$.

To represent the assignment of nodes to clusters, we introduce a cluster membership indicator $s_{i\omega}$. This quantity measures the degree of affinity of the tree indexed i to the cluster $\omega \in \Omega$ and is in the interval [0, 1]. When the cluster membership is close to 1 then there is a strong association of the node to the cluster; when the value is close to 0 then the association is weak.

Later on, it will be convenient to work with a matrix representation of the cluster membership indicators. Hence, we introduce a vector of indicator variables for the cluster indexed ω which we denote by $\underline{s}_\omega = (s_{1\omega}, s_{2\omega}, \ldots)^T$. The vectors are used as the columns of the $|V| \times |\Omega|$ cluster membership matrix $S = (\underline{s}_1 | \underline{s}_2 | \ldots | s_{|\Omega|})$ whose rows are indexed by the set of nodes and whose columns are indexed by the set of clusters.

We adopt a maximum likelihood approach to the joint recovery of the cluster indicators S and a revised estimate of the affinity matrix A with an improved block structure. Both problems are posed in terms of the joint log-likelihood function $\mathcal{L}(\mathcal{S}, \mathcal{A})$. To recover the cluster indicator variables, we maximize $\mathcal{L}(\mathcal{S}, \mathcal{A})$, with respect to S keeping A fixed. The re-estimation of A is posed as maximum likelihood estimation, with the cluster membership indicators playing the role of fixed data.

We assume that the the observed affinity structure of the pairwise clusters arises as the outcome of a series of Bernoulli trials, where the probability of success is the affinity weight. To be more formal, let us consider the pair of trees i and j. We are concerned with whether or not this pair of trees both belong to the cluster indexed ω. The random variable that governs the outcome of the Bernoulli trial is the product of indicator variables $\zeta_{i,j,\omega} = s_{i\omega} s_{j\omega}$. If $s_{i\omega} = s_{j\omega} = 1$, then the two trees have a pairwise association to the cluster indexed ω, and $\zeta_{i,j\omega} = 1$. When either $s_{i\omega} = 0$ or $s_{j\omega} = 0$, then the pair of trees do not both associate to the cluster ω and $\zeta_{i,j,\omega} = 0$. According to our Bernoulli model of the cluster formation process, success is the event that both nodes belong to the same cluster, while failure is the event that they do not. The parameter of the Bernoulli trial is the affinity-weight $A_{i,j}$. This simple model is captured by the distribution rule

$$p(\mathcal{S}_{i\omega}, \mathcal{S}_{j\omega} | A_{i,j}) = \begin{cases} A_{i,j} & \text{if } \mathcal{S}_{i\omega} = 1 \text{ and } \mathcal{S}_{j\omega} = 1 \\ (1 - A_{i,j}) & \text{if } \mathcal{S}_{i\omega} = 0 \text{ or } \mathcal{S}_{j\omega} = 0 \end{cases} \tag{18}$$

This rule can be written in the more compact form,

$$p(\mathcal{S}_{i\omega}, \mathcal{S}_{j\omega}|A_{i,j}) = A_{i,j}^{\mathcal{S}_{i\omega}\mathcal{S}_{j\omega}}(1 - A_{i,j})^{1-\mathcal{S}_{i\omega}\mathcal{S}_{j\omega}} \quad (19)$$

The joint log-likelihood function for the indicator variables S and the affinity matrix A is given by

$$\mathcal{L}(A, S) = \sum_{\omega\in\Omega}\sum_{(i,j)\in\Phi} \ln p(s_{i\omega}, s_{j\omega}|A_{i,j}) \quad (20)$$

After substituting the Bernoulli distribution into the log-likelihood function, we find that

$$\mathcal{L}(A, S) = \sum_{\omega\in\Omega}\sum_{(i,j)\in\Phi} \{s_{i\omega}s_{j\omega}\ln A_{i,j} + (1 - s_{i\omega}s_{j\omega})\ln(1 - A_{i,j})\} \quad (21)$$

Collecting terms this simplifies to

$$\mathcal{L}(A, S) = \sum_{\omega\in\Omega}\sum_{(i,j)\in\Phi} \left\{\mathcal{S}_{i\omega}\mathcal{S}_{j\omega}\ln\frac{A_{i,j}}{1 - A_{i,j}} + \ln(1 - A_{i,j})\right\} \quad (22)$$

The log-likelihood function can be maximized with respect to the affinity-weights and the cluster membership variables. This is interleaves two iteration steps. First, the maximum likelihood affinity-weight is located by taking the outer-product of the vectors of cluster membership indicators. Second, we update the cluster membership indicators by applying a naive mean field method to the likelihood function.

3.2.1. Updating the Link-Weight Matrix. Our aim is to explore how the log-likelihood function can be maximized with respect to the affinity-weights and the cluster membership indicators. In this section, we turn our attention to the first of these. To do this we compute the derivatives of the log-likelihood function with respect to the elements of the affinity-weight matrix

$$\frac{\partial\mathcal{L}}{\partial A_{i,j}} = \sum_{\omega\in\Omega}\left\{s_{i\omega}s_{j\omega}\frac{1}{A_{i,j}(1 - A_{i,j})} - \frac{1}{1 - A_{i,j}}\right\} \quad (23)$$

The matrix of updated affinity-weights $\hat{A}$ may be found by setting the derivatives to zero and solving the equation $\frac{\partial\mathcal{L}}{\partial A_{i,j}} = 0$. The derivative vanishes when

$$\hat{A}_{i,j} = \frac{1}{|\Omega|}\sum_{\omega\in\Omega}\mathcal{S}_{i\omega}\mathcal{S}_{j\omega} \quad (24)$$

In other words, the affinity-weight for the pair of nodes (i, j) is simply the average of the product of individual node cluster memberships over the different perceptual clusters. We can make the structure of the updated affinity-weight matrix clearer if we make use of the vector of membership variables for the cluster indexed ω, i.e. $\underline{s}_\omega = (\mathcal{S}_{1\omega}, \mathcal{S}_{2\omega}, \ldots)^T$. With this notation the updated affinity-weight matrix is $\hat{A} = \frac{1}{|\Omega|}\sum_{\omega\in\Omega}\underline{s}_\omega\underline{s}_\omega^T$. Hence, the updated affinity-weight matrix is simply the average of the outer-products of the vectors of cluster membership indicators.

3.2.2. Updating Cluster Membership Variables. We can repeat the gradient-based analysis of the log-likelihood function to develop update equations for the cluster-membership variables. We commence by computing the derivatives of the log-likelihood function with respect to the cluster-membership variable

$$\frac{\partial\mathcal{L}(A, S)}{\partial\mathcal{S}_{i\omega}} = \sum_{j\in V}\mathcal{S}_{j\omega}\ln\frac{A_{i,j}}{1 - A_{i,j}} \quad (25)$$

Since the associated saddle-point equations are not tractable in closed form, we use the soft-assign ansatz of Bridle (1990) to update the cluster membership assignment variables. This is a form of naive mean field theory (Ghahramani and Jordan, 1997). According to mean field theory the cluster memberships should be updated by replacing them with their expected values (Hofmann and Buhmann, 1997). Rather than performing the detailed expectation analysis, soft-assign allows the cluster memberships to be approximated by exponentiating the partial derivatives of the log-likelihood function. The updated cluster memberships are given by

$$\hat{\mathcal{S}}_{i\omega} = \frac{\exp\left[\frac{\partial\mathcal{L}(A,S)}{\partial s_{i\omega}}\right]}{\sum_{i\in V}\exp\left[\frac{\partial\mathcal{L}(A,S)}{\partial\mathcal{S}_{i\omega}}\right]} = \frac{\exp\left[\sum_{j\in V}\mathcal{S}_{j\omega}\ln\frac{A_{i,j}}{1-A_{i,j}}\right]}{\sum_{i\in V}\exp\left[\sum_{j\in V}\mathcal{S}_{j\omega}\ln\frac{A_{i,j}}{1-A_{i,j}}\right]} \quad (26)$$

After simplifying the argument of the exponential, the update formula reduces to

$$\hat{s}_{i\omega} = \frac{\prod_{j\in V}\left\{\frac{A_{i,j}}{1-A_{i,j}}\right\}^{\mathcal{S}_{j\omega}}}{\sum_{i\in V}\prod_{j\in V}\left\{\frac{A_{i,j}}{1-A_{i,j}}\right\}^{\mathcal{S}_{j\omega}}} \tag{27}$$

It is worth pausing to consider the structure of this update equation. First, the updated affinity-weights are an exponential function of the current ones. Second, the exponential constant is greater than unity, i.e. there is re-enforcement of the cluster memberships, provided that $A_{i,j} > \frac{1}{4}$.

The update process requires and initialization for the cluster membership indicators.

Here we make use of the method of Sarkar and Boyer (1998) who have shown how the same-sign eigenvectors of the matrix of similarity-weights can be used to define disjoint clusters. for clustering. The eigenvalues $l_1, l_2 \ldots$ of A are the solutions of the equation $|A - lI| = 0$ where I is the $|K| \times |K|$ identity matrix. The corresponding eigenvectors $\mathbf{v}_{l_1}, \mathbf{v}_{l_2}, \ldots$ are found by solving the equation $A\mathbf{v}_{l_i} = l_i\mathbf{v}_{l_i}$. Let the set of same-sign eigenvectors be represented by $\Omega = \{\omega | l_\omega > 0 \wedge [(\mathbf{v}^*_\omega(i) \geq 0 \forall i) \vee \mathbf{v}^*_\omega(i) \leq 0 \forall i])\}$. Since the same-sign eigenvectors are orthogonal, this means that there is only one value of ω for which $\mathbf{v}^*_\omega(i) \neq 0$. In other words, each node i is associated with a unique cluster. We therefore initialize the cluster membership variables using the same sign eigenvectors and set

$$\mathcal{S}^{(0)}_{iw} = \frac{|\mathbf{v}^*_{\omega_0}(i)|}{\sum_{i\in T_{\omega_0}}|\mathbf{v}^*_{\omega_0}(i)|}. \tag{28}$$

Since each node is associated with a unique cluster, this means that the updated affinity matrix is composed of non-overlapping blocks. Moreover, the link-weights are guaranteed to be in the interval [0, 1]. Finally, it is important to note that the updating of the link-weights is a unique feature of our algorithm which distinguishes it from the pairwise clustering methods of Hofmann and Buhmann (1997) and Shi and Malik (2000).

3.3. Algorithm Description

Finally, to summarize, the iterative steps of the pairwise clustering algorithm are as follows:

1. *Initialization*: Compute the initial current affinity-weight matrix $A^{(0)}$ using (17). The same-sign eigenvectors of $A^{(0)}$ are used to initialize the cluster-membership variables using Eq. (28).
2. Compute the updated cluster-membership variables using Eq. (27).
3. Update affinity-weight matrix $\hat{A}$ using Eq. (24).
4. Repeat steps (2) and (3) until convergence is reached.

Once the cluster membership has been determined, the relevant objects are extracted and the algorithm is iterated again on the remaining nodes. This procedure is repeated untill all the objects are assigned to a class. In this way the algorithm does not need require a priori information concerning the number of classes, but instead infers the class structure from the available data in an unsupervised manner.

At present we have no formal proof of convergence or analysis of the rate of convergence of the clustering algorithm. However empirical evidence show that it locates stable clusters in 100 iterations, and appears to monotonically increase the log-likelihood function.

4. Experimental Results

We illustrate the utility of the tree-clustering algorithm on sets of shock trees. The organization of this section is as follows. We commence by discussing the shock-tree and its use as a structural representation of 2D shape. We then provide our experimental results. The experimental evaluation of the clustering method is divided into three parts. We commence by evaluating the effectiveness of the weighted tree edit distance as a means of distinguishing and clustering different shapes. Next, we provide a quantitative analysis of the proportion of correct classifications with the clustering algorithm. Finally, we present a sensitivity analysis to explore the effect of structural differences between trees on the computed edit distance.

4.1. Shock Trees

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. The idea of characterizing boundary shape using the differential singularities of the reaction equation was first introduced into the computer

vision literature by Kimia et al. (1995). The idea is to evolve the boundary of an object to a canonical skeletal form using the reaction-diffusion equation. The skeleton represents the singularities in the curve evolution, where inward moving boundaries collide. The reaction component of the boundary motion corresponds to morphological erosion of the boundary, while the diffusion component introduces curvature dependent boundary smoothing. In practice, the skeleton can be computed in a number of ways (Arcelli and di Baja, 1992; Ogniewicz, 1994). Recently, Siddiqi, Tannenbaum and Zucker have shown how the eikonal equation which underpins the reaction-diffusion analysis can be solved using the Hamilton-Jacobi formalism of classical mechanics (Bouix and Siddiqi, 2000; Siddiqi et al., 1999a).

With the skeleton to hand, then the next step is to devise ways of using it to characterize the shape of the original object boundary. Here we follow Zucker, Siddiqi, and others, by labeling points on the skeleton using so-called shock-labels (Siddiqi et al., 1999b). According to this taxonomy of local differential structure, there are different classes associated with the behavior of the radius of the bitangent circle from the skeleton to the nearest pair of boundary points. 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 (Shokoufandeh et al., 1999; Siddiqi et al., 1999b). 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. We consider two variants of the shock tree matching problem. In the first variant we use a purely structural approach. The nodes in the shock trees are given uniform weight and we match only their structure. The second variant is weighted. Here we assign to each shock group a weight that is determined by the ratio of the length of the associated skeleton branch to the mean length of the associated "left" and "right" boundary segments as outlined in Torsello and Hancock (2004). Suppose that the ratio is denoted by $dl/d\mathcal{S}$. This quantity is related to the rate of change of the bitangent circle radius along the skeleton, i.e. $dr/d\mathcal{S}$, by the formula $\frac{dl}{d\mathcal{S}} = \sqrt{1 - (\frac{dr}{ds})^2}$. The nodes in our graphs are the skeletal branches which are assigned a weight which is by the average value of $dl/d\mathcal{S}$ along the relevant skeletal branch. The edges indicate connectivity of the skeletal branches. The level of a node in the tree is determined by the time of formation of the corresponding skeletal branch (Shokoufandeh et al., 1999; Siddiqi et al., 1999b). The later the time of formation of the shock, the higher the corresponding node in the hierarchy.

Figure 6 shows an example of shape matching using the shock tree abstraction. The top row displays the silhouettes with the corresponding skeleton superimposed. The skeleton is already separated into shock-branches. the bottom row shows the corresponding tree abstractions and the matching nodes. Note that the root is a dummy node added to maintain connectivity, while the direct children of the root are points of local maxima of the radius, and have zero boundary to shock ratio and, hence, weight. The matching took an average of 146 milliseconds over 10 runs on a 2 GHz PC.

4.2. *Qualitative Results*

The silhouettes used to generate the shock graphs at the basis of our experiments are shown in Fig. 2. There are 25 different shapes. These include brushes, tools, spectacles, various animals and human hands. The figure is annotated with the pairwise similarity of the shapes. For the shapes indexed i and j, the similarity measure is defined as

$$S_{i,j} = 1 - \frac{1}{2} d(t_i, t_j), \tag{29}$$

where $d(t_i, t_j)$ is the edit-distance between shapes i and j. In order to make the comparison independent of the size of the picture or of the shock-tree representation, the weight on each tree was previously normalized by dividing it by the sum of all the weights on every node of the tree. In this way the maximum possible edit-distance between two trees was 2 units.

For comparison purposes, Fig. 3 reports the similarities between the unweighted shock trees. In this case

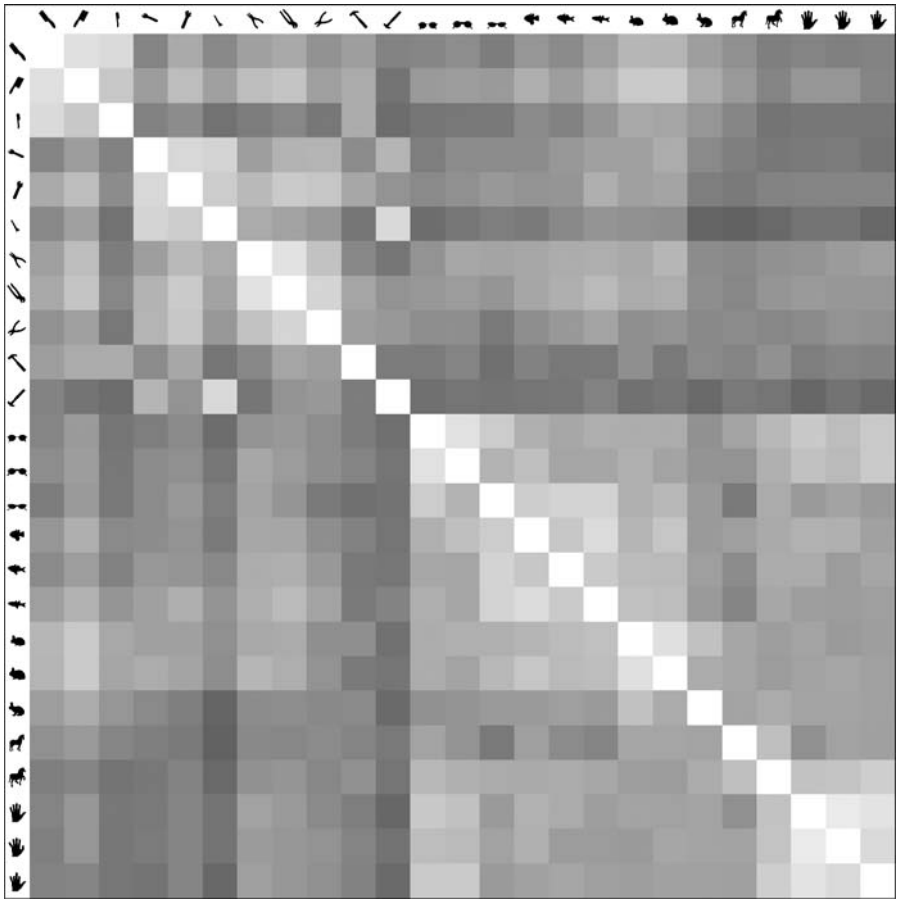


Figure 2. Pairwise similarities between shapes for the weighted shock trees.

the similarity between trees t_i and t_j is

$$T_{i,j} = \frac{1}{2}\frac{|V_i| + |V_j| - d(t_i, t_j)}{2}\left(\frac{1}{|V_i|} + \frac{1}{|V_j|}\right), \quad (30)$$

where V_i is the node set of tree t_i and $d(t_i, t_j)$ is the unattributed edit-distance between tree t_i and tree t_j.

In Figs. 4 and 5 we show the six best matched shapes for each object from the database. The top row of the figures shows the shapes considered. The remaining rows, from top to bottom, show the six best matched shapes ordered according to similarity. Hence, the further down we go in each column, the poorer the match to the shape in the top position. Figure 4 shows the matches obtained when we associate the shape measure to the shock trees. In each case the first matched shape is the object under study. From the third row down errors begin to emerge. For instance, a monkey wrench (object 6) matches to a hammer (object 11), and a horse (object 22) matches to a hand (object 25). Although there are 6 such errors in the third row (objects 6, 10, 11, 14, 16, 22), several of these are associated with small differences in the similarity values. This is the case with object 6, where a monkey wrench is matched to a hammer. In both objects the dominant feature is the long handle. Additionally, for four of the objects the correct matches appear in the fourth (object 6, 16), fifth (object 14), or sixth (object 22) position. It is only the two hammers that pose a real problem. This is due to the fact that the handle, the main feature on both objects, shows variation in its differential properties. Specifically, object 10 bulges at the grip, creating two separate shock segments for the handle, whereas the handle of object 11 generates a single shock segment. The problem could be solved by allowing the edit-distance calculation to merge segments, as

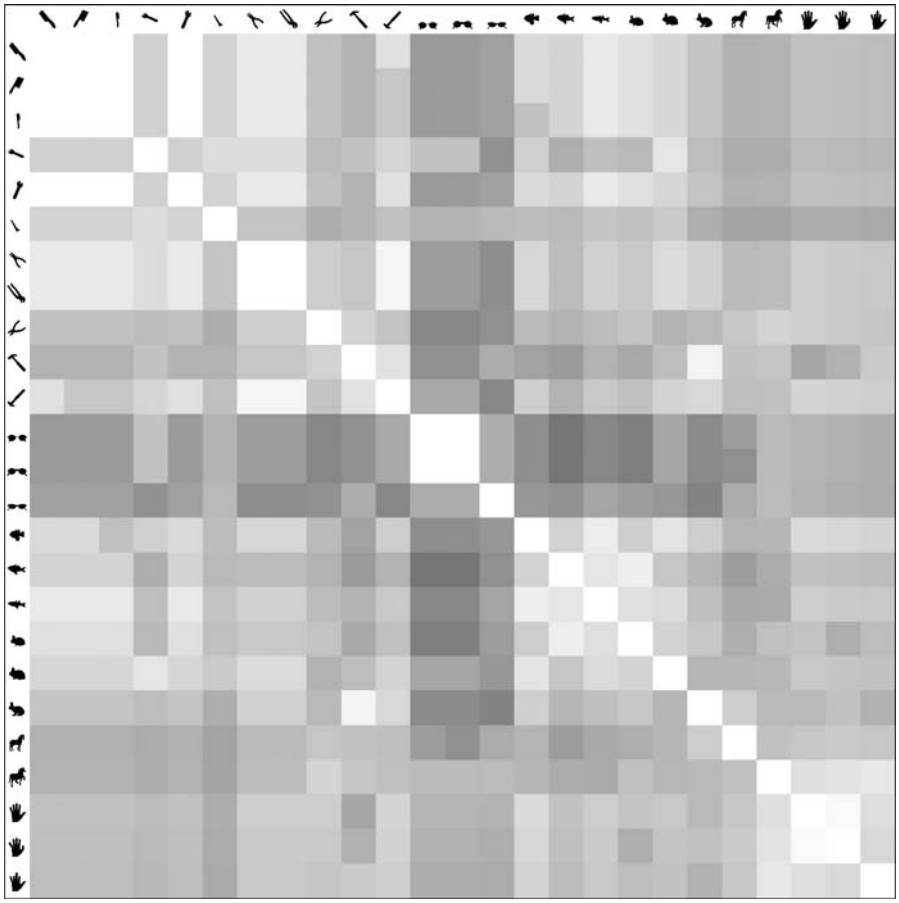


Figure 3. Pairwise similarities between shapes for the unweighted shock trees.

do Sebastian, Sebastian et al. (2001, 2004) but this is beyond the scope of the present study.

Figure 5 displays the top matches obtained using unattributed shock trees. Here again the top match is a perfect fit in each case. However, the performance degrades more quickly as we go down the columns of the table. In fact, the first error emerges in the second row of the figure.

To visualize the pairwise relationships between the different shapes, we have performed multi-dimensional scaling on the set of pairwise similarities. Multi-dimensional scaling is a well known statistical technique for visualizing data which exists in the form of pairwise similarities rather than ordinal values. 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 the 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 (1994).

The projections of the edit-distances onto the 2D space spanned by the two leading eigenvectors is shown in Fig. 7 (where the skeleton is weighted with the border-length to shock length ratio) and Fig. 8 (where it is not). When the skeleton is weighted with this ratio the MDS projection reveals the emergence of some class structure. Unfortunately, the full shape-structure is not captured by the two leading eigenvectors. For instance, the hands, the fish, the tools and the brushes all appear close to each other. In addition, there is no clear delineation of the shape-classes. When the skeleton is not weighted using the above-mentioned measure the grouping of the shapes is even poorer,

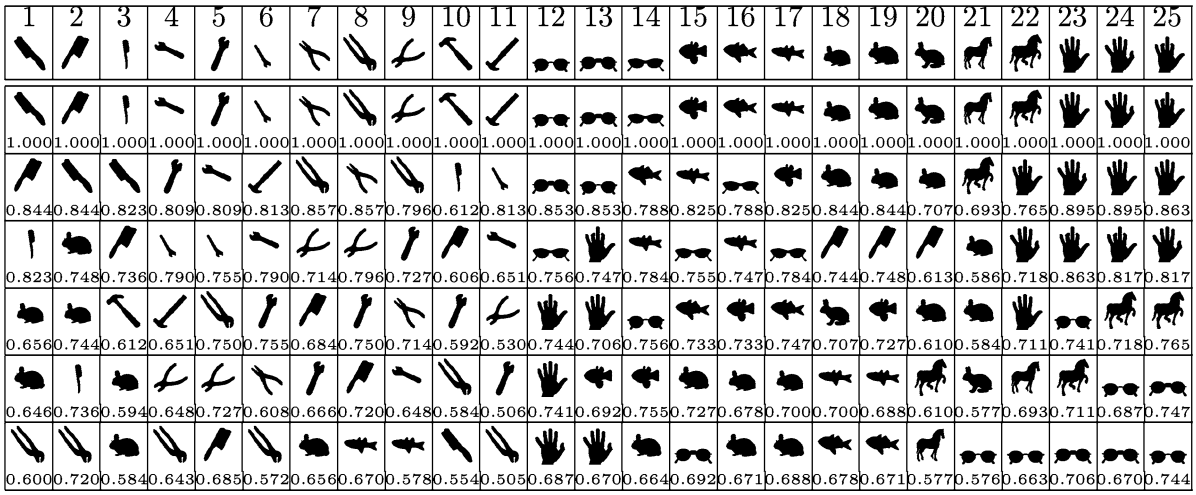


Figure 4. Top six matches for each shape for the weighted shock trees.

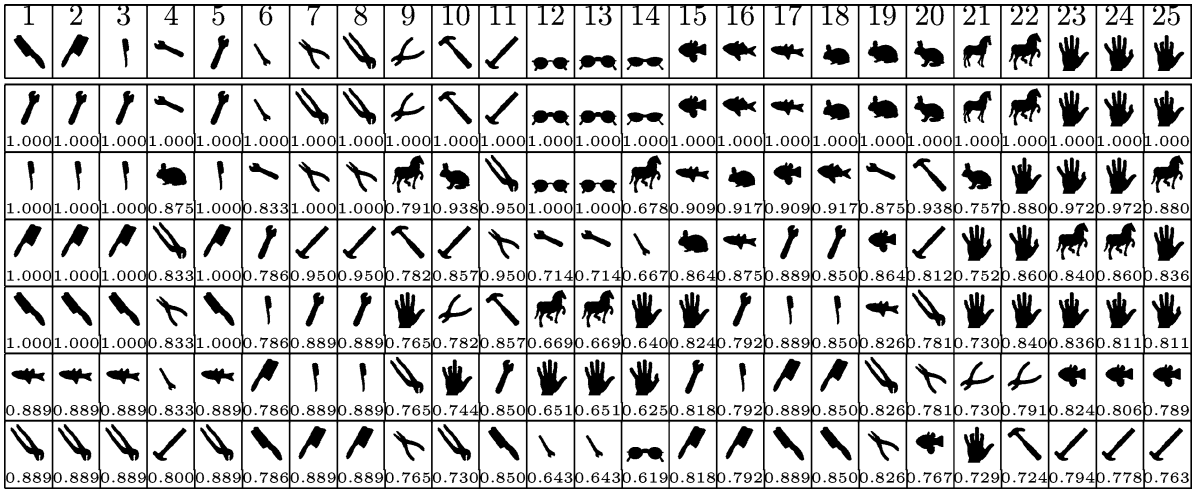


Figure 5. Top six matches for each shape for the unweighted shock trees.

with only the spectacles forming a well separated group.

Encouraged by these results, we have performed a detailed pairwise clustering of the pattern of similarities. Here we use the method detailed in Section 3 of the paper. The initial and final matrices of pairwise distance are shown in Fig. 9 for the measure-weighted skeleton and in Fig. 10 for the unweighted skeleton. In the case of the weighted skeleton the initial pairwise similarity matrix shows a strong separation of the shape-groups, which is further re-enforced by the iterative clustering method. On the basis of the block structure of the final matrix of pairwise distances, we identify eight clusters. Figure 11(a) presents the clusters in order of extraction.

In other words, the hands, tools, spectacles and animals form clusters. However, there are shapes which leak between these clusters. The problems encountered above are due to the fact that certain shapes straddle the true shape-classes and cause cluster-merging. Figure 11 (b) shows the result of applying the clustering algorithm to a pruned set of 16 shapes.

This is a much better set of clusters, which reflects the true shape-classes in the data. We have repeated these clustering experiments with the unweighted skeletons.

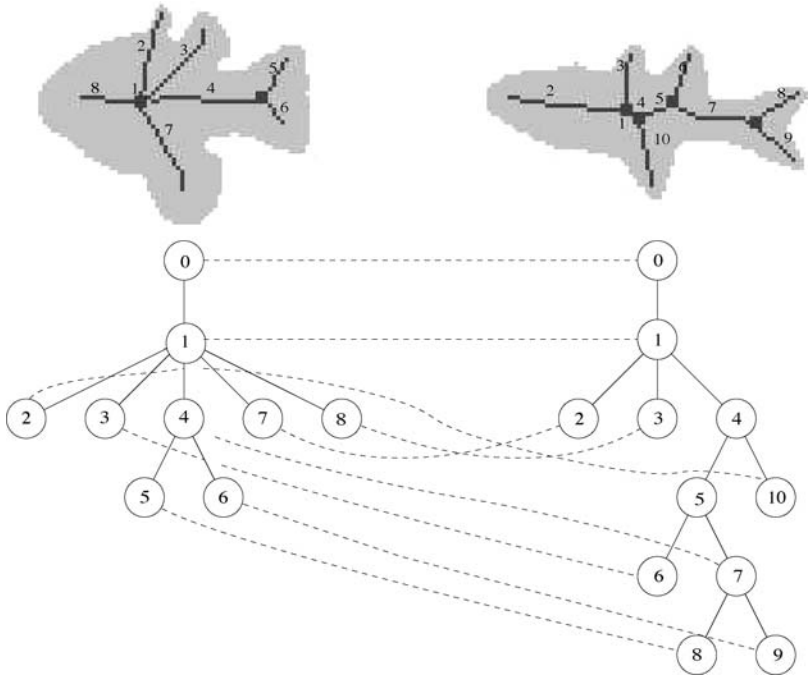


Figure 6. Example of shape matching using shock trees.

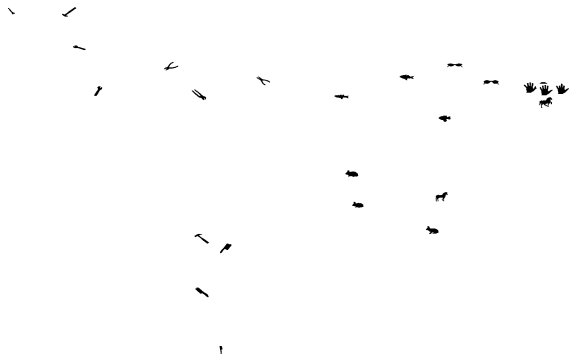


Figure 7. First and second principal components of the edit-distances of the shapes for the weighted shock trees.

Figure 8. First and second principal components of the edit-distances of the shapes for the unweighted shock trees.

Here the initial pairwise similarity matrix contains less structure than in the weighted case, and iteration of the clustering algorithm results in a noisier set of final cluster membership indicators (Fig. 10(b)). In particular, the clusters extracted from unweighted shock trees do not appear to correlate well with the shape classes in the database (Fig. 12(a)).

Clearly there is considerable merging and leakage between clusters. As illustrated in Fig. 12 (b), the classification does not improve when the algorithm is applied to the reduced database.

4.3. Quantitative Analysis

We now turn our attention to the properties of the weighted variant of our edit-distance approach when applied to larger databases. The first database consists of 150 shapes divided into 10 shape classes containing 15 shapes each. The silhouettes in this database are segmented quite roughly and the resulting skeletons

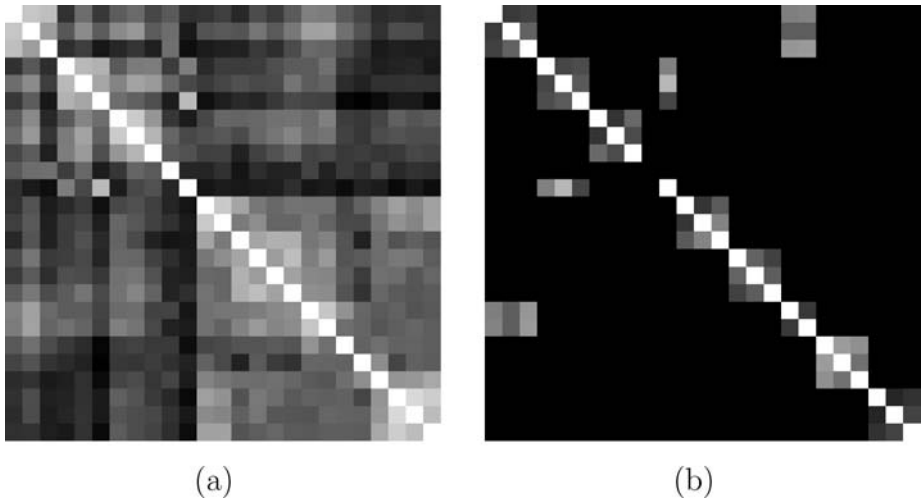


Figure 9. (a) Initial similarity matrix for the weighted tree edit-distances; (b) Final similarity matrix for the weighted tree edit-distances.

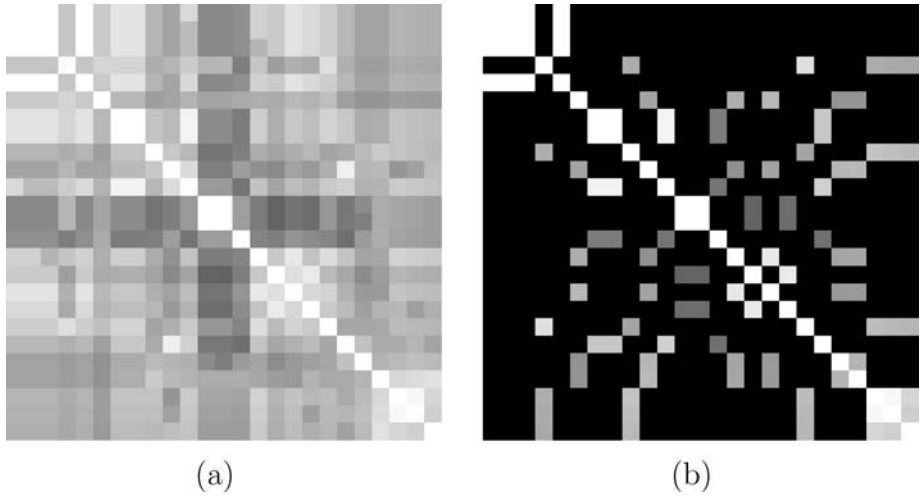
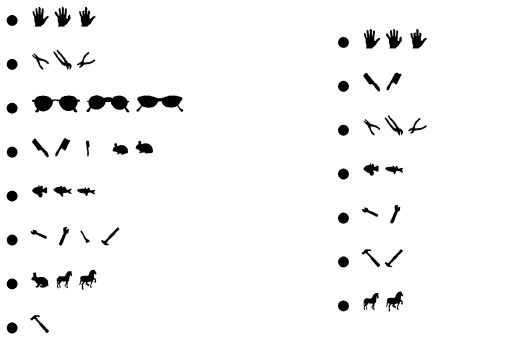


Figure 10. (a) Initial similarity matrix for the unweighted tree edit-distances; (b) Final similarity matrix for the unweighted tree edit-distances.

are very noisy. The second database consists of a subset of the MPEG7 shape database used in Sebastian et al. (2001). This database is composed of 420 shapes divided into 35 shape classes of 12 shapes each.

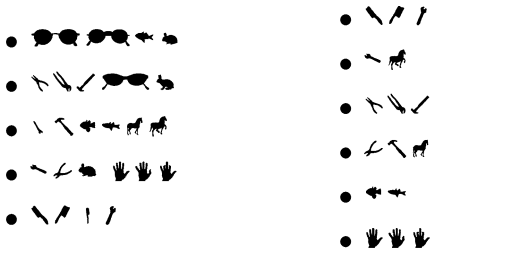
In Fig. 13 we show the results of the application of multi-dimensional scaling to the edit-distances between the trees in the first database. Each label in the plot corresponds to a particular shape class. Label 1 identifies cars, label 2 dogs, 3 ducks, 4 fishes, 5 hands, 6 horses, 7 leaves, 8 men, 9 pliers, and, finally, label 10 is associated with screwdrivers. The plot clearly shows the difficulty of this clustering problem, which clearly cannot be linearly separated at least in with a 2D embedding. The shape-groups are not well separated. Rather, there is a good deal of overlap between them. Furthermore, there are a considerable number of outliers. Note, however, that while edit-distance satisfies metric properties, it is unlikely that the distances can be embedded in a 2D subspace without distortion.

In Fig. 14 the results of the application of multi-dimensional scaling to the distances extracted from the MPEG7 database is displayed. As is the case for the first database, here too the shape-groups are not well separated. Instead, they are significantly overlapped and there are a considerable number of



a) Full database. b) Partial database.

Figure 11. Clusters extracted from weighted edit-distance.



a) Full database. b) Partial database.

Figure 12. Clusters extracted from un-weighted edit-distance.

outliers.

To assess the ability of the clustering algorithm to separate the shape-classes, we have performed experiments on an increasing number of shapes. With the first database we commenced with the 30 shapes from two shape-classes, and then increased the number of shape-classes under consideration untill there were 120 shapes, i.e. 8 classes. With the MPEG7 database we commenced with 24 shapes from two shape-classes, and then increased the number of shape-classes under consideration to 32, i.e. to 384 shapes in total. Each experiment was performed four times with different choices of shape-groups in order to allow us to perform an error analysis.

In order to evaluate the approach, we compare the classes obtained by clustering edit-distance with those obtained using the tree distance metric $d1$ which is computable in polynomial time and which is presented in Torsello et al. (2005). The similarity measure for the two trees is $S(t_1, t_2) = \max_\phi \sum_{(u,v)\in\phi} \sigma(u, v)$, where ϕ is a subtree isomorphism between t_1 and t_2. Hence, the distance metric is defined as:

$$d1(t_1, t_2) = 1 - \frac{S(t_1, t_2)}{\max(|t_1|, |t_2|)}.$$

Additionally, we explore the use of two alternative pairwise clustering algorithms for grouping the different distance measures. In order to provide a fair comparison, we restricted the analysis to clustering algorithms that do not require the number of classes to be known *a priori*. The selected algorithms are the matrix factorization algorithm developed by Perona and Freeman (1998) and the dominant-set framework developed by Pavan and Pelillo (2003a). In order to assess the quality of the clusterings, we have used two well-known cluster-validation measures (Jain and Dubes, 1988). The first is the standard classification rate. To compute the measure, for each cluster, we note the predominant shape class. Those graphs assigned to the cluster which do not belong to the predominant shape class are deemed to be misclassified. The classification rate is the total number of graphs belonging to the predominant cluster shape classes, divided by the total number of graphs. This measure exhibits a well known bias towards a large number of classes. To overcome this we also used the Rand index. The Rand index is defined as $R_I = \frac{A}{A+B}$. 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. The quantity B is the number of "disagreements," that is, the number of pairs of graphs that belong to different shape classes and that are assigned to different clusters. The index is hence the fraction of graphs of a particular shape class that are closer to a graph of the same class than to one of another class. Note that the classification rate does not penalize oversegmentation of the clusters, while the Rand index is specifically designed to do so.

Figure 15 plots the proportion of shapes from the first database correctly classified as the number of shapes is increased, while Fig. 16 plots the Rand index of the extracted groups. From these two plots we can draw several conclusions. First, the dominant set framework is the clear winner, providing the best results regardless of the distance measure used. Note, however, that the large difference in classification rate is misleading. In fact, the dominant set framework is known to extract very compact clusters. It hence has the tendency to oversegment the data into many clusters. On the other hand, the classification rate does not penalize oversegmentation. Instead, it is biased in favor of algorithms that tend to oversegmenting the data. The

Figure 13. Multi-dimensional scaling of the pairwise distances of the shock graphs. The numbers correspond to the shape-classes.

Rand index does not exhibit this bias. Although the dominant set method is still the best performer, the performance gap is significantly reduced. The Perona and Freeman algorithm has the worst performance of the three clustering algorithms. The clusters extracted from edit-distance where better than those extracted from the dl metric using the same clustering algorithm. In general, as the number of shapes increases then so the classification rate decreases, while the Rand index increases. This means that while the clusters tend to incorporate more noise as the number of shapes is increased, the overall shape class-structure remains intact and do not fragment into unrelated clusters.

Figure 17 plots the classification rate of the groups extracted from the much larger MPEG7 database as the number of shapes is increased, while Fig. 18 plots the corresponding Rand index. We can see that the values of the Rand index are generally higher than those obtained from the previous database. This is probably do to the fact that the silhouettes in this database are better segmented and the resulting shock-graphs are less noisy. In this database the proposed approach is the best performer, however the dominant set framework confirms it quality, providing very similar results. Similarly to what observed with the first database, the dominant set framework starts as the best performer, but it is overcome by the proposed approach when clustering more than 100 shapes. Finally, the experiments confirms that the Perona Freeman approach yields the lowest values for the Rand index. The difference in performance between edit-distance and dl metric is marginal for the proposed approach, but it is significant for the other algorithms.

In conclusion, the experiments show that the combination of edit-distance and the proposed approach can

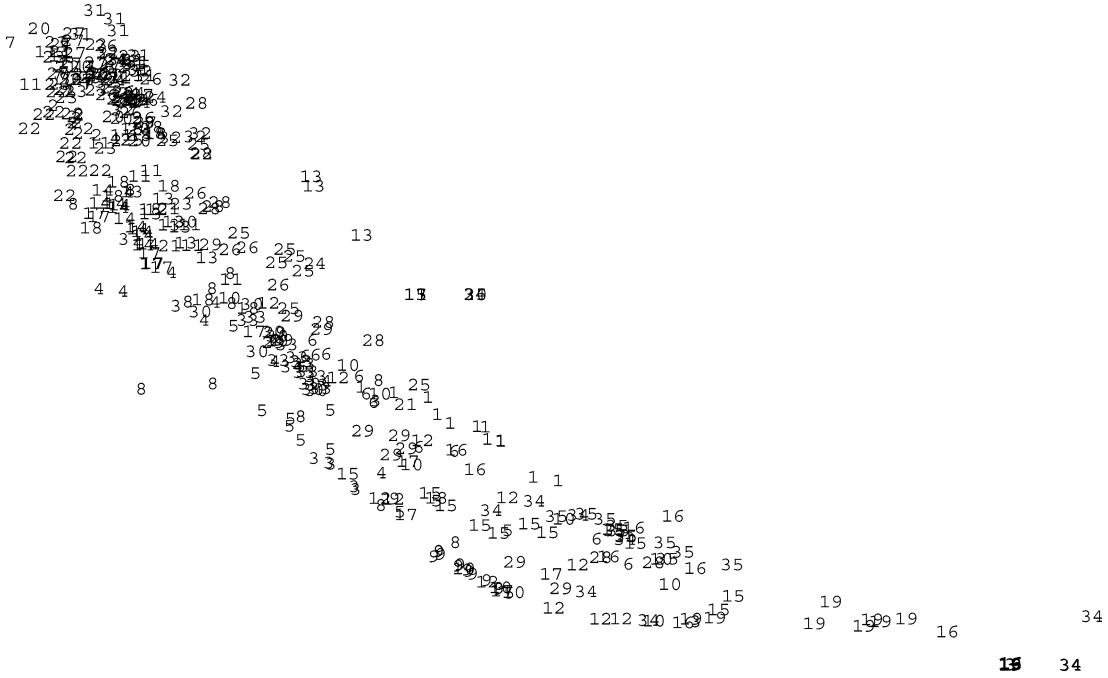


Figure 14. Multi-dimensional scaling of the pairwise distances of the shock graphs extracted from the MPEG database. The numbers correspond to the shape-classes.

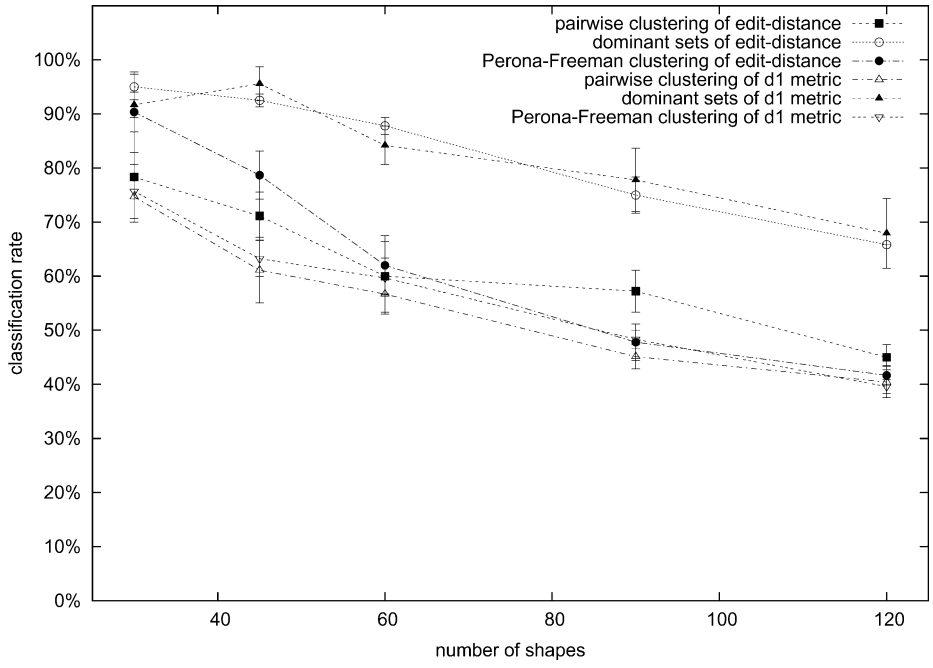


Figure 15. Proportion of correct classifications obtained with pairwise clustering of the distances of the shapes in the first database.

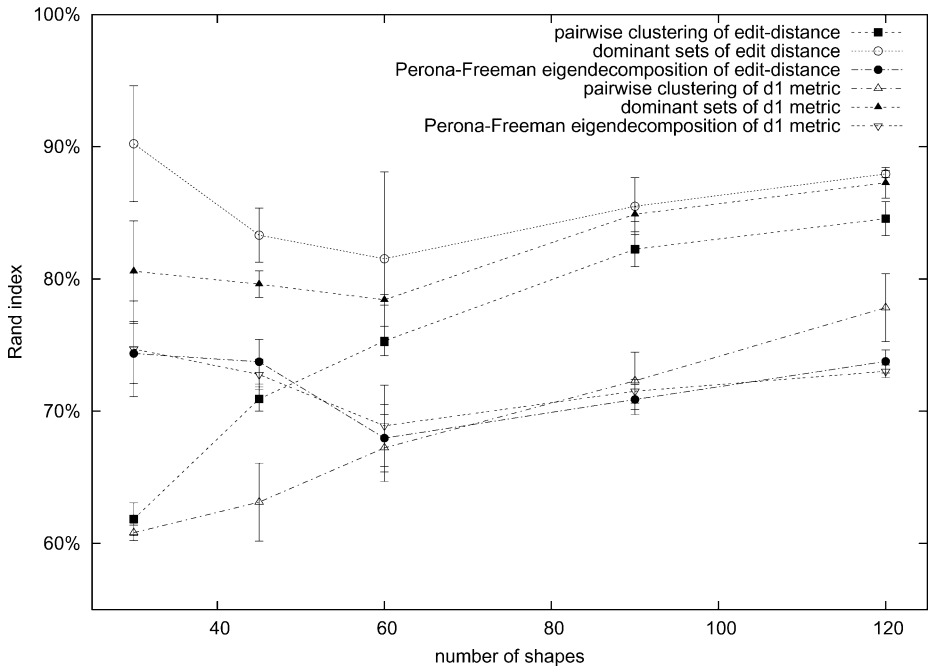


Figure 16. Rand index of the groups obtained with pairwise clustering of the distances of the shapes in the first database.

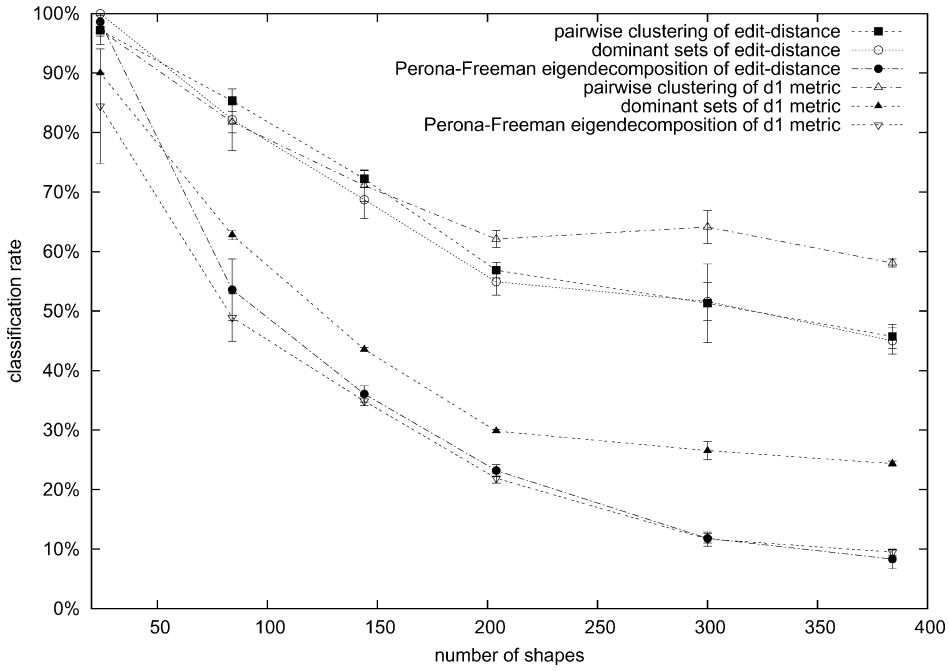


Figure 17. Proportion of correct classifications obtained with pairwise clustering of the distances of the shapes in the MPEG7 database.

extract the class structure present in a shape database with good accuracy, Comparative evaluation show that the dominant-set framework is a valid alternative to the proposed clustering approach, while the Perona-Freeman matrix factorization approach performs much worse. On the other hand, edit-distance constantly yields better clusters than the dl metric, although the difference can be marginal in some cases, making

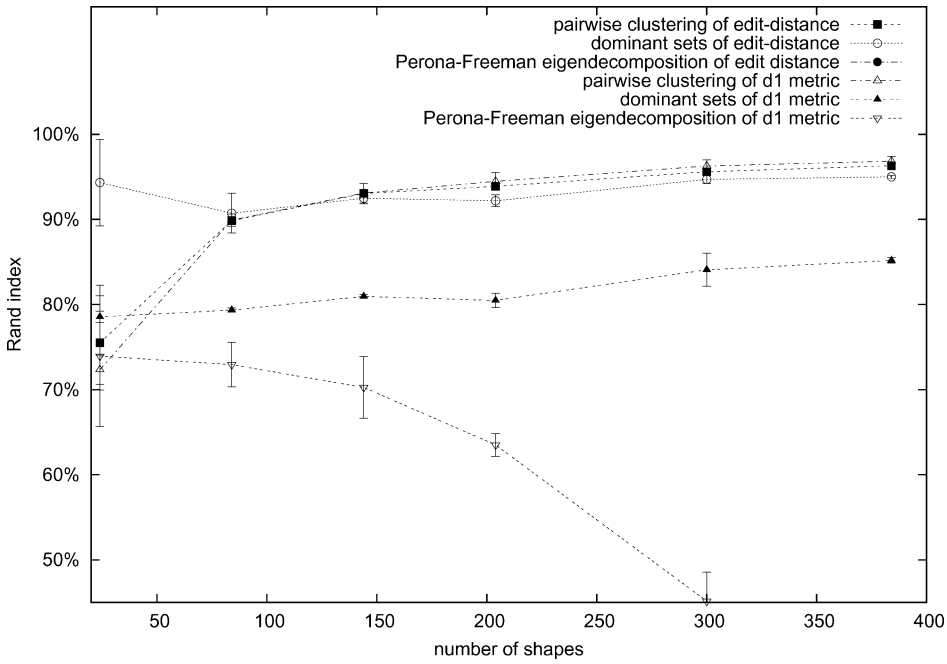


Figure 18. Rand index of the groups obtained with pairwise clustering of the distances of the shapes in the MPEG7 database.

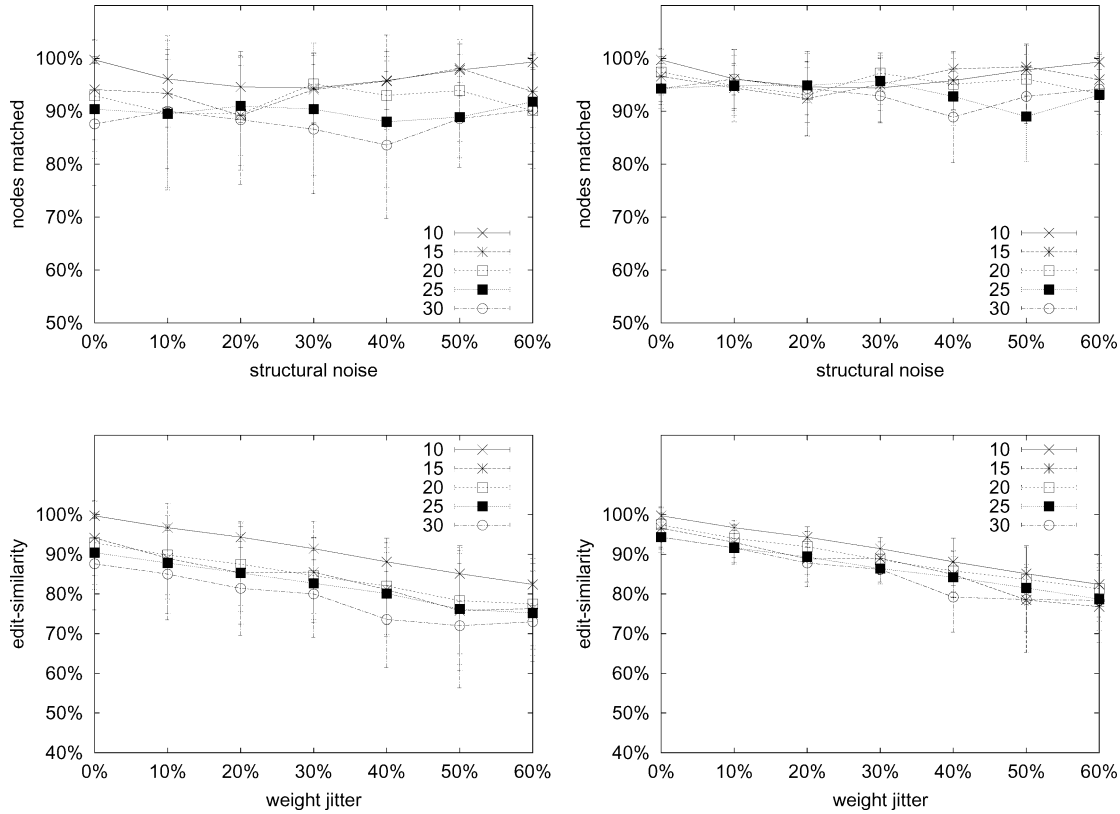


Figure 19. Sensitivity analysis: top-left node removal, top-right node removal without outliers, bottom-left weight jitter, bottom-right weight jitter without outliers.

dl distance preferable for its lower computational cost.

4.4. Sensitivity Study

To augment these real world experiments, we have performed a sensitivity analysis. The aim here is to characterize the behavior of our edit-distance algorithm when confronted with measurement errors resulting from noise, or jitter, on the weights and with structural errors resulting from node removal.

To test how well the method copes with structural modification we use it to match a randomly generated tree with modified versions of itself. To create these modified versions we removed an increasing fraction of nodes. Since we remove nodes only from one tree, the edited tree will have an exact match against the unedited version. Hence, we know the optimum value of the weight that should be attained by the maximum edited isomorphism. This is equal to the total weight of the edited tree.

By adding measurement errors or jitter to the weights, we test how well the method copes with a modification in the weight distribution. The measurement errors are distributed normally, with zero mean and controlled variance. Here we match the tree with noisy or jittered weights against its noise-free version. In this case we have no easy way of determining the optimal weight of the isomorphism, but we do expect a smooth drop in total weight with increasing noise variance.

We performed the experiments on trees with 10, 15, 20, 25, and 30 nodes. For each experimental run we used 11 randomly generated trees. The procedure for generating the random trees was 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 parents were 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 unit. 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.

The fraction of nodes removed was varied from 0% to 60%. In Fig. 19 top left we show the ratio of the computed weighted edit-distance to the optimal value of the maximum isomorphism. Interestingly, for certain trees the relaxation algorithm failed to converge within the allotted number of iterations. Furthermore, the algorithm also failed to converge on the noise corrupted variants of these trees. In other cases, the algorithm exhibited particularly rapid convergence. Again, the variants of these trees also showed rapid algorithm convergence. When the method fails to converge in an allocated number of iterations, we can still give a lower bound to the weight. However, this bound is substantially lower than the average value obtained when the algorithm does converge. The top right-hand graph of Fig. 19 shows the proportion of matched nodes when we eliminate these convergence failures. The main conclusions that can be drawn from these two plots are as follows. First, the effect of increasing structural error is a systematic underestimation of the weighted edit-distance. Second, the different curves exhibit a minimum within the plot-range. The reason for this is that the matching problem becomes trivial as the trees are decimated to extinction.

The bottom row of Fig. 19 shows the results obtained when measurement errors or jitter was added to the weights. Noise corrupted weights were obtained by adding random Gaussian noise with standard deviation ranging from 0 to 0.6. The bottom left-hand graph shows the result of this test. It is clear that the matched weight decreases almost linearly with the noise standard deviation. In these experiments, we encountered similar problems with algorithm convergence failure. Furthermore, the problematic trees were identical. This further supports the observation that the performance of the algorithm strongly depends on the randomized realization of the tree. The bottom right-hand plot shows the results of the jitter test with the convergence failures removed. Here we see a smaller variation in performance as the number of nodes increases.

5. Conclusions

This paper has presented a study of the problem of clustering shock tress. We commence by showing how to gauge the similarity of the trees using weighted edit distance. To compute the edit distance, we have adopted an optimization approach to tree matching.We show that any tree obtained with a sequence of cut operations is a subtree of the transitive closure of the original tree. Furthermore we show that the necessary condition for any subtree to be a solution can be reduced a clique problem in a derived structure. Using this idea we transform the tree edit distance problem

into a series of maximum weight cliques problems and then we use relaxation labeling to find an approximate solution.

To identify distinct groups of graphs, we developed a maximum likelihood algorithm for pairwise clustering. This takes as its input, a matrix of pairwise similarities between shock-trees computed from the edit distances. The algorithm interleaved iterative steps for computing cluster-memberships and for updating the pairwise similarity matrix. The number of clusters is controlled by the number of same-sign eigenvectors of the current similarity matrix. Experimental evaluation of the method shows that it is capable of extracting clusters of trees or graphs which correspond closely to the shape-categories present. Comparative studies show that both the edit-distance and the clustering algorithm are competitive with the state of the art.

References

Arcelli, C. and di Baja, G.S. 1992. Ridge points in euclidean distance maps. *Pattern Recognition Letters*, 13:237–243.

Barrow, H.G. and Burstall, R.M. 1976. Subgraph isomorphism, matching relational structures and maximal cliques. *Inf. Proc. Letter*, 4:83–84.

Blum, H. 1973. Biological shape and visual science (Part I). *Journal of theoretical Biology*, 38:205–287.

Bomze, I.M., Pelillo, M., and Stix, V. 2000. Approximating the maximum weight clique using replicator dynamics. *IEEE Trans. on Neural Networks*, vol. 11.

Bouix, S. and Siddiqi, K. 2000. Divergence-based medial surfaces. In *Computer Vision ECCV 2000*, Springer, LNCS 1842, pp. 603–618.

Bridle, J. S. 1990. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *Advances in Neural Information Processing Systems 2*, San Mateo, CA: Morgan Kaufmann, pp. 211–217.

Bunke, H. and Kandel, A. 2000. Mean and maximum common subgraph of two graphs. *Pattern Recognition Letters*, 21:163–168.

Cox, T.F. and Cox, M.A.A. 1994. *Multidimensional Scaling*. Chapman & Hall.

Bowyer, K.W. and Dyer, C.R. 1990. Aspect graphs: An introduction and survey of recent results. *International Journal of Imaging Systems and Technology*, 2:315–328.

Cyr, C.M. and Kimia, B.B. 2004. A similarity-based aspect-graph approach to 3D object recognition. *International Journal of Computer Vision*, 57(1):5–22.

Denton, T., Abrahamson, J., Shokoufandeh, A. 2004a. Approximation of canonical sets and their applications to 2D view simplification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)*, vol. 2, pp. 550–557.

Denton, T., Demirci, M., Abrahamson, J., Shokoufandeh, A., and Dickinson, S. 2004b. Selecting canonical views for view-based 3-D object recognition. In *Proceedings, International Conference on Pattern Recognition (ICPR)*, Cambridge, U.K.

Dickinson, S., Pentland, A., and Rosenfeld, A. 1992. 3D shape recovery using distributed aspect matching. *IEEE Trans. Pattern Anal. Machine Intell.*, 14(2):174–198.

Eshera, M.A. and Fu, K.-S. 1986. An image understanding system using attributed symbolic representation and inexact graph-matching. *IEEE Trans. Pattern Anal. Machine Intell.*, 8:604–618.

Ghahramani, Z. and Jordan, M.I. 1997. Factorial hidden markov models. *Machine Learning*, 29:245–273.

Gibbons, L.E. et al. 1997. Continuous characterizations of the maximum clique problem. *Math. Oper. Res.*, 22:754–768.

Giblin, P.J. and Kimia, B.B. 1999. On the local form and transitions of symmetry sets, medial axes, and shocks. In *Proc. Int. Conf. Computer Vision*, pp. 385–391.

Hofmann, T. and Buhmann, M. 1997. Pairwise data clustering by deterministic annealing. *IEEE Trans. Pattern Anal. Machine Intell.*, 19(1):1–14.

Jain, A.K. and Dubes, R.C. 1988. *Algorithms for Clustering Data*. Prentice Hall: Englewood Cliffs, NJ.

Kimia, B.B., Tannenbaum, A.R., and Zucker, S.W. 1995. Shapes, shocks, and deforamtions i. *Int. J. Computer Vision*, 15:189–224.

Klein, P. et al. 1999. A tree-edit-distance algorithm for comparing simple, closed shapes. In *ACM-SIAM Symp. on Disc.e Alg.*

Lozano, M. and Escolano, F. 2003. EM Algorithm for Clustering and ensemble of graphs with comb matching. *EMMVCPR*, *Lecture Notes in Computer Science*, vol. 2683, pp. 52–67.

Luo, B., Wilson, R.C., and Hancock, E.R. 2003. Spectral embedding of graphs, *Pattern Recognition*, 36:2212–2223.

Motzkin, T.S. and Straus, E.G. 1965. Maxima for graphs and a new proof of a theorem of Turán. *Canadian Journal of Mathematics*, 17:533–540.

Munger, A., Bunke, H., and Jiang, X. 1999. Combinatorial search vs. genetic algorithms: A case study based on the generalized median graph problem. *Pattern Recognition Letters*, 20(11–13):1271–1279.

Ogniewicz, R.L. 1994. A multiscale mat from voronoi diagrams: the skeleton-space and its application to shape description and decomposition. In *Aspects of Visual Form Processing*, pp. 430–439.

Papadimitriou, C.H. and Steiglitz, K.J. 1982. *Combinatorial optimization: Algorithms and Complexity*. Prentice-Hall: Englewood Cliffs, NJ.

Pavan, M. and Pelillo, M. 2003a. A new graph-theoretic approach to clustering and segmentation. In *Proc. IEEE Conf. Computer Vision Pattern Recognition*, vol. I, pp. 145–152.

Pavan, M. and Pelillo, M. 2003. Dominant sets and hierarchical clustering. In *Proc. Int. Conf. Computer Vision*, vol. I, pp. 362–369.

Pelillo, M. 1997. The dynamics of relaxation labeling process, *J. Math. Imaging Vision*, 7:309–323.

Pelillo, M. 1999. Replicator equations, maximal cliques, and graph isomorphism. *Neural Computation*, 11:1935–1955.

Pelillo, M., Siddiqi, K., and Zucker, S.W. 1999. Matching hierarchical structures using association graphs. *IEEE Trans. Pattern Anal. Machine Intell.*, 21(11):1105–1120.

Pelillo, M. and Torsello, A. 2006. Payoff-monotonic game dynamics and the maximum clique problem. *Neural Computation*, 18:1215–1258.

Perona, P. and Freeman, W.T. 1998. A factorization approach to grouping. In *European Conference on Computer Vision*, vol. 1, pp. 655–670.

Reyner, S.W. 1977. An analysis of a good algorithm for the subtree problem. *SIAM Journal on Computing*, 6:730–732.

Rizzi, S. 1998. Genetic operators for hierarchical graph clustering. *Pattern Recognition Letters*, 19:1293–1300.

Sarkar, S. and Boyer, K.L. 1998. Quantitative measures of change based on feature organization: Eigenvalues and eigenvectors. *Computer Vision and Image Understanding*, 71(1):110–136.

Sebastian, T.S., Klein, P.N., and Kimia, B.B. 2001. Recognition of shapes by editing shock graphs. In *Proc. Int. Conf. Computer Vision*, vol. 1, pp. 755–762.

Sebastian, T.B., Klein, P.N., and Kimia, B.B. 2004. Recognition of shapes by editing their shock graphs. *IEEE Trans. Pattern Anal. Machine Intell.*, 26(5):550–571.

Segen. J. 1988. Learning graph models of shape. In J. Laird (Ed.), *Proceedings of the Fifth International Conference on Machine Learning*, pp. 29–25.

Sengupta, K. and Boyer, K.L. 1995. Organizing large structural modelbases. *IEEE Trans. Pattern Anal. Machine Intell.*, 17(4).

Sengupta, K. and Boyer, K.L. 1998. Modelbase partitioning using property matris spectra. *Computer Vision and Image Understanding*, 70(2).

Shi, J. and Malik, j. 2000. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Machine Intell.*, 22(8):888–905.

Shokoufandeh, A. et al. 1999. Indexing using a spectral encoding of topological structure. In *Proc. IEEE Conf. Computer Vision Pattern Recognition*.

Siddiqi, K., Bouix, S., Tannenbaum, A., and Zucker, S.W. 1999a. The hamilton-jacobi skeleton. In *Proc. Int. Conf. Computer Vision*, pp. 828–834.

Siddiqi, K., Shokoufandeh, A., Dickinson, S.J., and Zucker, S.W. 1999b. Shock graphs and shape matching. *Int. J. Computer Vision*, 35(1):13–32.

Tirthapura, S. et al. 1998. Indexing based on edit-distance matching of shape graphs. In *SPIE Int. Symp. on Voice, Video, and Data Comm.*, pp. 25–36.

Torsello, A. and Hancock, E.R. 2003. Computing approximate tree edit distance using relaxation labelling. *Pattern Recognition Letters*, 24:1089–1097.

Torsello, A., and Hancock, E.R. 2004. A skeletal measure of 2D shape similarity. *Computer Vision and Image Understanding*, 95(1):1–29.

Torsello, A. and Hancock, E.R. 2006. Learning shape-classes using a mixture of tree-unions. *IEEE Trans. Pattern Anal. Machine Intell.*, 28(6):954–967.

Torsello, A., Hidovič-Rowe, D., and Pelillo, M. 2005. Polynomial-time metrics for attributed trees. *IEEE Trans. Pattern Anal. Machine Intell.*, 27(7):1087–1099.

Tsai, W.H. and Fu, K.S. 1979. Error-correcting isomorphism of attributed relational graphs for pattern analysis. *IEEE Transactions on Systems, Man and Cybernetics*, 9:757–768.