Efficiently Computing Weighted Tree Edit Distance Using Relaxation Labeling

Andrea Torsello and Edwin R. Hancock

Department of Computer Science University of York
York, YO10 5DD, UK
atorsell@cs.york.ac.uk

Abstract. This paper investigates an approach to tree edit distance problem with weighted nodes. We show that any tree obtained with a sequence of cut and relabel 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 to a clique problem in a derived structure. Using this idea we transform the tree edit distance problem into a series of maximum weight clique problems and then we use relaxation labeling to find an approximate solution.

1 Introduction

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. Moreover, the problem has recently acquired significant topicality with the need to develop ways of retrieving images from large data-bases. Stated succinctly, the problem is one of inexact or error-tolerant graph-matching. Early work on the topic included Barrow and Burstall's idea [1] 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 his co-workers [6]. The idea behind edit distance [18] 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 [4] . In particular, he showed that, under certain assumptions concerning the edit-costs, computing the 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 not 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.

The work reported in this paper builds on a simple observation which follows from Bunke's work. By re-casting the search for the maximum common subgraph as a max clique problem [1], then we can efficiently compute the edit distance. A diverse array of powerful heuristics and theoretical results are available for solving the max clique problem. In particular the Motzkin-Straus theorem [10] allows us to transform the max clique problem into a continuous quadratic programming problem. An important recent development is reported by Pelillo [11] who shows how probabilistic relaxation labeling can be used to find a (local) optimum of this quadratic programming problem.

In this paper we are interested in measuring the similarity of tree structures obtained from a skeletal representation of 2D shape. 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. For instance, Bartoli et al. [2], use the graph theoretic notion of a path string to transform the tree isomorphism problem into a single max weighted clique problem. This work uses a refinement of the Motzkin Strauss theorem to transform the max weighted clique problem into a quadratic programming problem on the simplex [3], the quadratic problem is then solved using relaxation labeling.

Because of the added connectivity and partial order constraints mentioned above, Bunke's result linking the computation of edit distance to the size of the maximum common subgraph does not translate in a simple way to trees. Furthermore, specific characteristics of trees suggest that posing the tree-matching problem as a variant on graph-matching is not the best approach. In particular, both the tree isomorphism and the subtree isomorphism problems have efficient polynomial time solutions. Moreover, Tai [16] has proposed a generalization of the string edit distance problem from the linear structure of a string to the non-linear structure of a tree. The resulting tree edit distance differs from the general graph edit distance in that edit operations are carried out only on nodes and never directly on edges. The edit operations thus defined are node deletion, node insertion and node relabeling. This simplified set of edit operations is guaranteed to preserve the connectivity of the tree structure. Zhang and Shasha [22] have investigated a special case which involves adding the constraint that the solution must maintain the order of the children of a node. With this order among siblings, they showed that the tree-matching problem is still in P and gave an algorithm to solve it. In subsequent work they showed that the unordered case was indeed an NP hard problem [23]. The NP-completeness, though, can be eliminated again by adding particular constraints to the edit operations. In particular, it can be shown that the problem returns to P when we add the constraint of strict hierarchy, that is when separate subtrees are constrained to be mapped to separate subtrees [21].

In this paper we propose an energy minimization method for efficiently computing the weighted tree edit distance. We follow Pelillo [11] by casting the problem into the Motzkin-Straus framework. To achieve n this goal we use the graph-theoretic notion of tree closure. We show that, given a tree T, then any tree obtained by cutting nodes from T is a subtree of the closure of T. Furthermore, we can eliminate subtrees that can not be obtained from T by solving a series of max-clique problems. In this way we provide a divide and conquer method for finding the maximum edited common subtree by searching for maximal cliques of an association graph formed from the closure of the two trees. With this representation to hand, we follow Bomze et al. [3] 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.

2 Exact Tree Matching

In this section we describe a polynomial time algorithm for the subtree isomorphism problem. This allows us to formalize some concepts and give a starting point to extend the approach to the minimum tree edit distance problem.

2.1 Association Graph

The phase space we use to represent the matching of nodes is the directed association graph, a variant of the association graph. The association graph is a structure that is frequently used in graph matching problems. The nodes of the association graph are 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 of the association graph represent the pairwise constraints of the problem: they represent both connectivity on the original graphs and the feasibility of a solution with the linked associations.

Hierarchical graphs have an order relation induced by paths: given two nodes a and b, (a, b) is in this relation if and only if there is a path from a to b. When the directed graph is acyclical, this relation can be shown to be an (irreflexive) order relation. The use of directed arcs in the association graph allows us to make use of this order. We connect nodes with directed arcs in a way that preserves the ordering of the associated graph. The graph obtained can be shown to be ordered still. Specifically, an association graph for the tree isomorphism problem can be shown to be a forest.

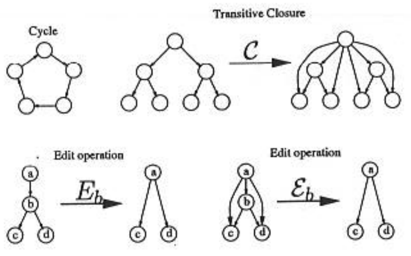

Fig. 1. Terminology on directed graphs

For the exact isomorphism problem (maximum common subgraph) the edges of the association graphs are:

$$(v, v') \to (u, u') \text{ iff } v \to u \text{ and } v' \to u' \tag{1}$$

where v and u are nodes on one graph and v' and u' are nodes on the other.

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

Proof. Let us assume that $(u_1, v_1) \to \cdots \to (u_n, v_n)$ is a cycle. Then, since an arc $(v, v') \to (u, u')$ in the association graph exists only if the arcs $v \to u$ and $v' \to u'$ exist in G and G' respectively, we have that $u_1 \to \cdots \to u_n$ is a cycle in G and $v_1 \to \cdots \to v_n$ 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. 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 $(v, v') \to (u, u')$. Due to the way the association graph is constructed this means that either u or u' must have at most one incoming edge. But t and t' are trees, so both u and u' have at most one incoming edge, namely the one from the parent.

The directed association graph can be used to reduce a tree matching problem into subproblems: the best match given the association of nodes v and v' can be found examining only descendents of v and v' This gives us a divide and conquer solution to the maximum common subtree problem: use the association graph to divide the problem and transform it into maximum bipartite match subproblems, the subproblems can then be efficiently conquered with known polynomial time algorithms. We then extend the approach to the minimum unlabeled tree edit problem and present an evolutionary method to conquer the subproblems. Finally, we present a method to convert the divide and conquer approach into a multi-population evolutionary approach.

2.2 Maximum Common Subtree

We present a divide and conquer approach to the exact maximum common subtree problem. We call the maximum common subtree rooted at (v, v') a solution to the maximum common subtree problem applied to two subtrees of t and t'. In particular, the solution is on the subtrees of t and t' rooted at v and v' respectively. This solution is further constrained with the condition that v and v' are roots of the matched subtrees.

With the maximum rooted isomorphism problem for each children of (v, v') at hand, the maximum isomorphism rooted at (v, v') can be reduced to a maximum bipartite match problem. 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 case of a non weighted tree this is the cardinality of the isomorphism. With this structure we have 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').

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 cardinality of t and t'. Furthermore, since there are nm nodes in the association graph, the problem is reduced to nm maximum bipartite match problems.

3 Inexact Tree Matching

We want to extend the algorithm to provide us with an error-tolerant tree isomorphism. There is a strong connection between the computation of maximum common subtree and the tree edit distance. In [4] Bunke showed that, under certain constraints applied to the edit-cost function, computing the maximum common subgraph problem and the minimum graph edit distance are equivalent to one-another.

This is not directly true for trees, because of the added constraint that a tree must be connected. But, extending the concept to the common edited subtree, we can use common substructures to find the minimum cost edited tree isomorphism. In particular, we want to match weighted trees. These are trees with weight associated to the nodes and with the property that the cost of an edit operation is a function of the weight of the nodes involved.

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.

In our model the cost node removal and insertion is equal to the weight of the node, while the cost of relabeling a node is equal to the difference in the weights. This approach identifies node removal to relabel to 0 weight and is a natural interpretation when the weight represents the "importance" of the 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, as long as we perform the operations on both trees.

At this point we introduce the concept of *edited isomorphism.* Assuming that we have two trees T_1 and T_2 and a tree T' that can be obtained from both with node removal and relabel operations, T' will induce an isomorphism between

nodes in T_1 and T_2 so that places two nodes in correspondence if and only if they get cut down to the same node in T'. We call such isomorphism an edited isomorphism induced by T'. From the definition it is clear that there is a tree T', obtained only with node removal and relabel operations, so that the sum of the edit distance from this tree to T_1 and T_2 is equal to the edit distance between T_1 and T_2, i.e. a median tree. We say that the isomorphism induced by this tree is a *maximum edited isomorphism* because it maximizes $W_m = \sum_{(i,j)} \min(w_i, w_j)$, where i and j are nodes matched by the isomorphism, and w_i and w_j are their weights. In fact, if we W_1 and W_2 be the weights in T_1 and T_2 respectively, the edit distance between T_1 and T' is $W_1 - W_m$, the distance between T_1 and T_2 is $W_1 + W_2 - 2W_m$. Clearly, finding the maximum edited isomorphism is equivalent to solving the tree edit distance problem.

3.1 Editing the Transitive Closure of a Tree

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 Ct of the tree t (see Figure 1). In both cases the edit operation removes the node v, all the incoming edges, and all the outgoing edges.

We show that the transitive closure operation and the node removal operation commute, that is we have:

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

Proof. If a node is in $\mathcal{E}_v(\mathcal{C}(t))$ it is clearly also in $\mathcal{C}(E_v(t))$. What is left is to show is that an edge (a, b) is in $\mathcal{E}_v(\mathcal{C}(t))$ if and only if it is in $\mathcal{C}(E_v(t))$.

If (a, b) is in $\mathcal{C}(E_v(t))$ then neither a nor b is v and there is a path from a to b in $E_v(t)$. Since the edit operation E_v preserves connectedness and the hierarchy, there must be a path from a to b in t as well. This implies that (a, b) is in $\mathcal{C}(t)$. Since neither a nor b is v, the operation $\mathcal{E}_v$ will not delete (a, b). Thus (a, b) is in $\mathcal{E}_v(\mathcal{C}(t))$.

If (a, b) is in $\mathcal{E}_v(\mathcal{C}(t))$, then it is also in $\mathcal{C}(t)$, because $\mathcal{E}_v(\mathcal{C}(t))$ is obtained from $\mathcal{C}(t)$ by simply removing a node and some edges. This implies that there is a path from a to b in t and, as long as neither a nor b are v, there is a path from a to b in $E_v(t)$ as well. Thus (a, b) is in $\mathcal{C}(E_v(t))$. Since (a, b) is in $\mathcal{E}_v(\mathcal{C}(t))$, both a and b must be nodes in $\mathcal{E}_v(\mathcal{C}(t))$ and, thus, neither can be v.

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

We call a subtree s of Ct *consistent* if for each node v of s there cannot be two children a and b so that (a, b) is in Ct. In other words, given two nodes a and b, siblings in s, s is consistent if and only if there is no path from a to b in t.

We can, now, prove the following:

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 a consistent subtree of the DAG Ct.*

Proof. Let us assume that there is an edit sequence $\{E_{v_i}\}$ that transforms t into $\hat{t}$, then, by virtue of the above lemma, the dual edit sequence $\{\mathcal{E}_{v_i}\}$ transforms $\mathcal{C}t$ into $\mathcal{C}\hat{t}$. By construction we have that $\hat{t}$ is a subtree of $\mathcal{C}\hat{t}$ and $\mathcal{C}\hat{t}$ is a subgraph of $\mathcal{C}t$, thus $\hat{t}$ is a subtree of $\mathcal{C}t$. Furthermore, since the node removal operations respect the hierarchy, $\hat{t}$ is a consistent subtree of $\mathcal{C}t$.

To prove the converse, assume that $\hat{t}$ is a consistent subtree of $\mathcal{C}t$. If (a,b) is an edge of $\hat{t}$, then it is an edge on $\mathcal{C}t$ as well, i.e. there is a path from a to b in t and we can define a sequence of edit operations $\{E_{v_i}\}$ that removes any node between a and b in such a path. Showing that the nodes $\{v_i\}$ deleted by the edit sequence cannot be in $\hat{t}$ we show that all the edit operations defined this way are orthogonal. As a result they can be combined to form a single edit sequence that solves the problem.

Let v in $\hat{t}$ be a node in the edited path and let p be the minimum common ancestor of v and a in $\hat{t}$. Furthermore, let w be the only child of p in $\hat{t}$ that is an ancestor of v in $\hat{t}$ and let q be the only child of p in $\hat{t}$ that is an ancestor of a in $\hat{t}$. Since a is an ancestor of v in t, an ancestor of v can be a descendant of a, an ancestor of a, or a itself. This means that w has to be in the edited path. Were it not so, then w had to be a or an ancestor of a against the hypothesis that p is the minimum common ancestor of v and a. Since q is an ancestor of a in t and a is an ancestor of w in t, q is an ancestor of w in t, but q and w are siblings in $\hat{t}$ against the hypothesis that $\hat{t}$ is consistent.

Using this result, we can show that the minimum cost edited tree isomorphism between two trees t and t' is a maximum common consistent subtree of the two DAGs $\mathcal{C}t$ and $\mathcal{C}t'$, provided that the cost of node removal and node matching depends only on the weights.

The minimum cost edited tree isomorphism is a tree that can be obtained from both model tree t and data tree t' with node removal and relabel operations. By virtue of the theorem above, this tree is a consistent subtree of both $\mathcal{C}t$ and $\mathcal{C}t'$. The tree must be obtained with minimum combined edit cost. Since node removal can be considered as matching to a node with 0 weight, the isomorphism that grants the minimum combined edit cost is the one that gives the maximum combined match, i.e. it must be the maximum common consistent subtree of the two DAGs.

3.2 Cliques and Common Consistent Subtrees

In this section we show that the directed association graph induces a divide and conquer approach to edited tree matching as well. Given two trees t and t' to be matched, we create the directed association graph of the transitive closures $\mathcal{C}t$ and $\mathcal{C}t'$ and we look for a consistent matching tree in the graph. That is we seek a tree in the graph that corresponds to two consistent trees in the transitive closures $\mathcal{C}t$ and $\mathcal{C}t'$. The maximum such tree corresponds to the maximum common consistent subtree of $\mathcal{C}t$ and $\mathcal{C}t'$.

In analogy to what we did for the exact matching case, we divide the problem into a maximum common consistent subtree rooted at (v,w), for each node

(v,w) of the association graph. We show that, given the weight of the maximum common consistent subtree rooted at each child of (v,w) in the association graph, then we can transform the rooted maximum common consistent subtree problem into a max weighted clique problem. Solving this problem for each node in the association graph and looking for the maximum weight rooted common consistent subtree, we can find the solution to the minimum cost edited tree isomorphism problem.

Let us assume that we know the weight of the isomorphism for every child of (v,w) in the association graph. We want to find the consistent set of siblings with greatest total weight. Let us construct an undirected graph whose nodes consist of the children of (v,w) in the association graph. We connect two nodes (p,q) and (r,s) if and only if there is no path connecting p and r in t and there is no path connecting q and s in t'. This means that we connect two matches (p,q) and (r,s) if and only if they match nodes that are consistent siblings in each tree. Furthermore, we assign to each association node (a,b) a weight equal to the weight of the maximum common consistent subtree rooted at (a,b). The maximum weight clique of this graph will be the set of consistent siblings with maximum total weight. The weight of the maximum common consistent subtree rooted at (v,w) will be this weight plus the minimum of the weights of v and w, i.e. the maximum weight that can be obtained by the match. Furthermore, the nodes of the clique will be the children of (v,w) in the maximum common consistent subtree.

3.3 Heuristics for the Maximum Weighted Clique

As we have seen, we have transformed an inexact tree matching problem into a series of maximum weighted clique problems. That is, we transformed one NP-complete problem into multiple NP-complete problems. The reason behind this approach lies in the fact that the max clique problem is, on average, a relatively easy problem. Furthermore, since the seminal paper of Barrow and Burstall [1], it is a standard technique for structural matching and a large number of approaches and very powerful heuristics exist to solve it or approximate it.

The approach we will adopt to solve each single instance of the max weight clique problem is an evolutionary one introduced by Bomze, Pelillo and Stix [3]. This approach is based on a continuous formulation of the combinatorial problem and transforms it into a symmetric quadratic programming problem in the simplex Δ. For more detail we refer to the appendix.

Relaxation labeling is a evidence combining process developed in the framework of constraint satisfaction problems. Its goal is to find a classification p that satisfies pairwise constraints and interactions between its elements. The process is determined by the update rule

$$p_i^{t+1}(\lambda) = \frac{p_i^t(\lambda) q_i^t(\lambda)}{\sum_\mu p_i^t(\mu) q_i^t(\mu)}, \tag{2}$$

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

In [12] Pelillo showed that the function $A(\mathbf{p}) = \sum_{i\lambda} p_i(\lambda)q_i(\lambda)$ is a Lyapunov function for the process, i.e. $A(\mathbf{p}^{t+1}) \geq A(\mathbf{p}^t)$, with equality if and only if $\mathbf{p}^t$ is stationary.

3.4 Putting It All Together

In the previous sections we proved that the maximum edited tree isomorphism problem can be reduced to nm maximum weight clique problem and we have given an iterative process that is guaranteed to find maximal weight cliques. In this section we will show how to use these ideas to develop a practical algorithm.

A direct way is to use the relaxation labeling dynamics starting from the leaves of the directed association graph and propagate the result upwards in the graph using the weight of the extracted clique 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 M of every isomorphism rooted at the descendants of u and v. Specifically, the compatibility coefficients are initialized as $R_{(u,v)} = \gamma \mathbf{e}\mathbf{e}^T - C$, or, equivalently, $r_{(u,v)}(a, a'b, b') = \gamma - c^{(u,v)}_{(a,a')(b,b')}$, where

$$c^{(u,v)}_{(a,a')(b,b')} = \begin{cases} \frac{1}{2M_{(a,a')}} & \text{if } (a,a') = (b,b') \\ c^{(u,v)}_{(a,a')(a,a')} + c^{(u,v)}_{(b,b')(b,b')} & \text{if } (a,a') \text{ and } (b,b') \text{ are consistent} \\ 0 & \text{otherwise.} \end{cases}$$

This approach imposes a sequentiality to an otherwise highly parallel algorithm. An alternative can be obtained transforming the problem into a single multi-object labeling process. With this approach we set up a labeling problem with one object per node in the association graph, and at each iteration we update the label distribution for each object. We, then, update the compatibility matrices according to the new weight estimate.

This multi-object approach uses the fact that the compatibility matrix for one rooted matching subproblem does not depend upon which nodes are matched below the root, but only on the number of matches. That is, to solve one subproblem we need to know only the weight of the cliques rooted at the children, not the nodes that form the clique.

Gibbons' result [7] guarantees that the weight of the clique is equal to $\frac{1}{\mathbf{x}^T B \mathbf{x}}$, where $\mathbf{x}$ is the characteristic vector of the clique and B is the weight matrix defined in (3). This allows us to generate an estimate of the clique at each iteration: given the current distribution of label probability $\mathbf{p}$ for the subproblem rooted at (u, v), we estimate the number of nodes matched under (u, v) as $\frac{1}{\mathbf{p}^T B \mathbf{p}}$, and thus we assign to (u, v) the weight $M_{(u,v)} = \frac{1}{\mathbf{p}^T B \mathbf{p}} + \min(w_u, w_v)$, that is the weight of the maximum set of consistent descendants plus the the weight that can be obtained matching node u with node v.

We obtain a two step update rule: at each iteration we update the label probability distribution according to equation (2), and then we use the updated distributions to generate new compatibility coefficients according to the rule $r_{u,v}(a, a', b, b') = \gamma - c^{(u,v)}_{(a,a')(b,b')}$, where

$$c^{(u,v)}_{(a,a')(b,b')} = \begin{cases} \frac{1}{2}\mathbf{p}_{a,a'} B^{(a,a')} \mathbf{p}_{a,a'} & \text{if } (a,a') = (b,b') \\ c^{(u,v)}_{(a,a')(a,a')} + c^{(u,v)}_{(b,b')(b,b')} & \text{if } (a,a') \text{ and } (b,b') \text{ are consistent} \\ 0 & \text{otherwise} \end{cases}$$

Another possible variation to the algorithm can be obtained using different initial assignments for the label distribution of each subproblem.

A common approach is to initialize the assignment with a uniform distribution so that we have an initial assignment close to the baricenter 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.

With our problem decomposition the wider cliques are the ones that map nodes at lower levels. As a result the solution will be biased towards matches that are very low on the graph, even if these matches require cutting a lot of nodes and are, thus, less likely to give an optimum solution.

A way around this problem is to choose an initialization that assigns a higher initial likelihood to matches that are higher up on the subtree. In our experiments we decided to initialize the probability of the association (a, b) for the subproblem rooted at (u, v) as $p_{(u,v)}(a, b) = e^{-(d_a + d_b + \epsilon)}$, where d_a is the depth of a with respect to u, d_b is the depth of b with respect to v, and ϵ is a small perturbation. Of course, we then renormalize $\mathbf{p}_{(u,v)}$ to ensure that it is still in the simplex.

4 Experimental Results

We evaluate the new tree-matching method on the problem of shock-tree matching. The idea of characterizing boundary shape using the differential singularities of the reaction equation was first introduced into the computer vision literature by Kimia, Tannenbaum and Zucker [8]. 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, here we use a variant of the method Siddiqi, Tannenbaum and Zucker, which solves the eikonal equation which underpins the reaction-diffusion analysis using the Hamilton-Jacobi formalism of classical mechanics [14]. Once the skeleton is to hand, the next step is to devise ways of using it to characterize the shape of the original boundary. Here we follow Zucker, Siddiqi, and others, by labeling points on the skeleton using so-called shock-labels [15]. According to this taxonomy of local differential structure, there are different classes associated with behavior of the radius of the osculating circle from the skeleton to the nearest pair of boundary points. The so-called shocks distinguish between the cases where the local osculating circle has maximum radius, minimum radius, constant radius or a radius which 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 [13, 15]. 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.

We generate two variants of this matching problem. In the first variant we use aq purely symbolic approach: Here the shock trees have uniform weight and we match only the structure. The second variant is weighted: we assign to each shock group a weight proportional to the length of the border that generates the shock; this value proves to be a good measure of skeletal similarity [17].

For our experiments we used a database consisting of 16 shapes. For each shape in the database, we computed the maximum edited isomorphism with the other shapes. In the unweighted version the "goodness" measure of the match is the average fraction of nodes matched,that is, $W(t_1, t_2) = \frac{1}{2}\left(\frac{\#\hat{t}}{\#t_1} + \frac{\#\hat{t}}{\#t_2}\right)$, where $\#t$ indicates the number of nodes in the tree t. Conversely, to calculate the goodness of the weighted match we weights so that the sum over all the nodes of a tree is 1. The way we use the total weight of the maximum common edited isomorphism as a measure for the match. In figure 2 we show the shapes and the goodness measure of their match. The top value of each cell is the result for the unweighted case, the bottom value is represents the weighted match.

To illustrate the usefullness of the set of similarity measures, we have used them as input to a pairwise clustering algorithm [9]. The aim here is see whether the clusters extracted from the weighted or the unweighted tree edit distance correspond more closely to the different perceptual shape categories in the database. In the unweighted case the process yielded six clusters (brush (1) + brush (2) + wrench (4); spanner (3) + horse (13) ; pliers (5) + pliers (6) + hammer (9) ;pliers (7) +hammer (8) + horse (12); fish (10) + fish (12); hand (14) + hand (15) + hand (16). Clearly there is merging and leakage between the different shape categories. Clustering on the weighted tree edit distances gives better results yielding seven clusters: brush (1) + brush (2) ; spanner (3) + spanner (4); pliers (5) + pliers (6) + pliers (7); hammer (8) + hammer (9); fish (10) + fish (11); horse (12) + horse (13); hand (14) + hand (15) + hand (16)). These correspond exactly to the shape categories in the data-base.

5 Sensitivity Study

To augment these real world experiments, we have performed a sensitivity analysis. The aim here is to characterise the effects measurement errors resulting from noise or jitter on the weights and the structural errors resulting from node removal.

Node removal tests the capability of the method to cope with structural modification. To do this we remove an increasing fraction of nodes from a randomly generated tree. We then we match the modified tree against its unedited version. 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

	1.000	1.000	0.774	1.000	0.889	0.889	0.706	0.643	0.850	0.818	0.889	0.635	0.640	0.706	0.694	0.684
	0.981	0.844	0.434	0.604	0.562	0.600	0.497	0.554	0.422	0.511	0.557	0.484	0.427	0.428	0.415	0.402
	1.000	1.000	0.774	1.000	0.889	0.889	0.706	0.643	0.850	0.818	0.889	0.635	0.640	0.706	0.694	0.684
	0.844	0.981	0.548	0.685	0.683	0.720	0.559	0.509	0.381	0.621	0.629	0.534	0.447	0.474	0.517	0.434
	0.774	0.774	1.000	0.774	0.833	0.833	0.676	0.714	0.800	0.773	0.694	0.615	0.620	0.676	0.667	0.658
	0.446	0.543	1.000	0.569	0.554	0.643	0.637	0.475	0.651	0.436	0.559	0.428	0.395	0.406	0.414	0.373
	1.000	1.000	0.774	1.000	0.889	0.889	0.706	0.643	0.850	0.818	0.889	0.635	0.640	0.706	0.694	0.684
	0.604	0.685	0.809	1.000	0.663	0.731	0.713	0.592	0.496	0.507	0.624	0.404	0.418	0.443	0.449	0.404
	0.889	0.889	0.694	0.889	1.000	1.000	0.765	0.730	0.950	0.808	0.778	0.673	0.680	0.765	0.750	0.737
	0.562	0.684	0.531	0.666	1.000	0.857	0.714	0.459	0.385	0.588	0.622	0.456	0.497	0.524	0.541	0.521
	0.889	0.889	0.833	0.889	1.000	1.000	0.765	0.730	0.950	0.808	0.778	0.673	0.680	0.765	0.750	0.737
	0.571	0.650	0.643	0.750	0.857	1.000	0.793	0.584	0.505	0.580	0.670	0.456	0.515	0.514	0.480	0.503
	0.706	0.706	0.676	0.706	0.765	0.765	1.000	0.782	0.794	0.674	0.680	0.730	0.692	0.765	0.801	0.669
	0.502	0.559	0.648	0.727	0.703	0.796	1.000	0.533	0.503	0.438	0.562	0.433	0.457	0.467	0.506	0.482
	0.643	0.643	0.714	0.643	0.730	0.730	0.847	1.000	0.771	0.649	0.639	0.769	0.724	0.651	0.698	0.682
	0.554	0.606	0.475	0.592	0.459	0.531	0.554	1.000	0.415	0.434	0.405	0.443	0.459	0.419	0.400	0.408
	0.850	0.850	0.800	0.850	0.950	0.950	0.794	0.857	1.000	0.764	0.739	0.692	0.700	0.794	0.778	0.763
	0.441	0.356	0.676	0.506	0.388	0.472	0.530	0.394	0.981	0.384	0.438	0.396	0.386	0.310	0.353	0.310
	0.818	0.818	0.773	0.818	0.808	0.808	0.599	0.731	0.764	1.000	0.909	0.647	0.720	0.749	0.806	0.789
	0.516	0.626	0.479	0.507	0.593	0.586	0.487	0.434	0.379	1.000	0.825	0.556	0.520	0.634	0.630	0.555
	0.889	0.889	0.694	0.889	0.778	0.778	0.680	0.548	0.739	0.909	0.778	0.598	0.680	0.595	0.667	0.737
	0.556	0.636	0.559	0.626	0.622	0.670	0.578	0.405	0.428	0.820	1.000	0.449	0.590	0.542	0.532	0.518
	0.635	0.635	0.615	0.635	0.673	0.673	0.778	0.769	0.692	0.647	0.598	1.000	0.785	0.730	0.752	0.729
	0.496	0.449	0.417	0.388	0.462	0.429	0.475	0.397	0.403	0.492	0.449	1.000	0.627	0.496	0.519	0.560
	0.640	0.640	0.620	0.640	0.680	0.680	0.741	0.724	0.700	0.655	0.680	0.824	1.000	0.840	0.764	0.834
	0.395	0.447	0.395	0.441	0.495	0.488	0.445	0.497	0.371	0.602	0.573	0.693	0.992	0.711	0.687	0.699
	0.706	0.706	0.676	0.706	0.765	0.765	0.706	0.651	0.715	0.824	0.765	0.730	0.840	1.000	0.972	0.947
	0.443	0.520	0.334	0.436	0.570	0.536	0.465	0.422	0.335	0.632	0.554	0.494	0.696	0.976	0.895	0.840
	0.694	0.694	0.667	0.694	0.750	0.750	0.801	0.698	0.778	0.659	0.583	0.752	0.812	0.915	1.000	0.920
	0.431	0.450	0.409	0.425	0.520	0.520	0.480	0.445	0.378	0.629	0.543	0.572	0.719	0.847	0.982	0.771
	0.684	0.684	0.658	0.684	0.737	0.737	0.669	0.682	0.763	0.718	0.573	0.729	0.741	0.947	0.920	0.947
	0.441	0.447	0.380	0.447	0.560	0.520	0.493	0.434	0.346	0.563	0.558	0.541	0.765	0.864	0.817	1.000

Fig. 2. Matching result for unweighted (top) and weighted (bottom) shock trees

weight that should be attained by the maximum edited isomorphism. This is equalt to the total weight of the edited tree.

By adding measurement errors or jitter to the weights, we test how the method copes with a modification in the weight distribution. The measurement errors are normally distributed with zero mean and controlled variance. Here we match the tree of noisy or jittered weights against its noise-free version. In this case we have no easy way to determine 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 commence with an empty tree (i.e. one with no nodes) and we iteratively add the required number of nodes. At each iteration nodes are added as children of one of the existing nodes. The

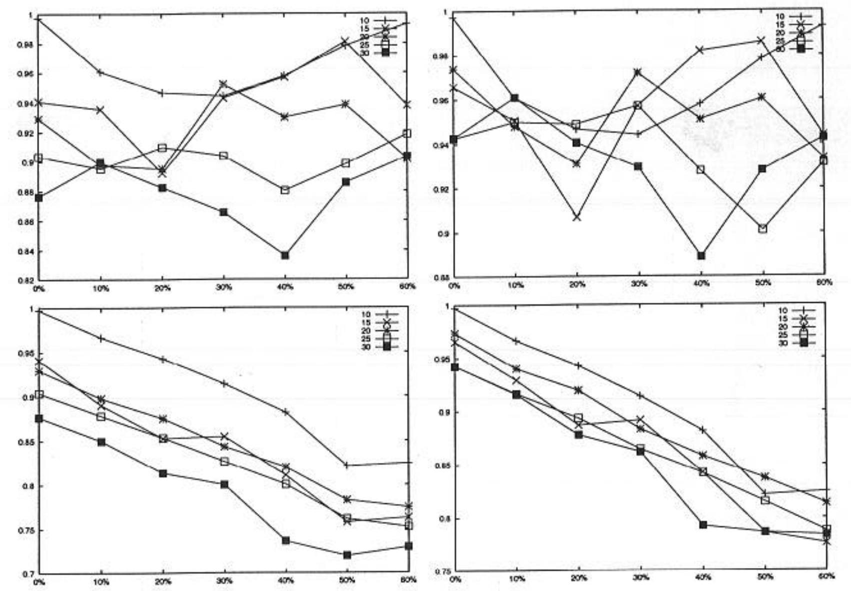

Fig. 3. Sensitivity analysis: top-left node removal, top-right node removal without outliers, bottom-left weight jitter, bottom-right weight jitter without outliers.

parents are randomly selected with uniform probability from among the existing nodes. The weight of the newly added nodes are selected at random from an exponential distribution with mean 1. This procedure will tend to generate trees in which the branch ratio is highest closest to the root. This is quite realistic of real-world situations, since shock trees tend to have the same characteristic.

The fraction of nodes removed was varied from 0% to 60%. In figure 3 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 figure 3 shows the ratio of weight matched when we eliminate these convergence failures. The main conclusion that can be drawn from these two plots are as follows. First, the effect of increasing structural error is to cause a systematic underestimation of the weighted edit distance. The different curves all exhibit a minimum value of the ratio. The reason for this is that the matching problem becomes trivial as the trees are decimated to extinction,

The bottom row of figure 3 shows the results obtained when we have added measurement errors or jitter to the weights. We noise corrupted weights were obtained with randomly added 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 non-convergence. Furthermore, the problematic instances were identical. This further supports the observation that the problem strongly depends on the instance. The bottom right-hand plot shows the results of the jitter test with the convergence failures removed.

6 Conclusions

In this paper we have investigated a optimization approach to tree matching. We based the work on to tree edit distance framework. We show that any tree obtained with a sequence of cut operation 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.

In a set of experiments we apply this algorithm to match shock graphs, a graph representation of the morphological skeleton. The results of these experiments are very encouraging, showing that the algorithm is able to match similar shapes together. Furthermore we provide some sensitivity analysis of the method.

A Motzkin-Strauss Heuristic

In 1965, Motzkin and Strauss [10] showed that the (unweighted) maximum clique problem can be reduced to a quadratic programming problem on the n-dimensional simplex $\Delta = \{\mathbf{x} \in \mathbb{R}^n | x_i \geq 0 \text{ for all } i = 1 \dots n, \sum_i x_i = 1\}$, here 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 $g(\mathbf{x}) = \mathbf{x}^T A \mathbf{x}$, where A is the adjacency matrix of G. Furthermore, given a set $S \subseteq V$, we define the *characteristic* vector $\mathbf{x}^S$

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

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

Gibbons *et al.* [7] generalized this result to the weighted clique case. In their formulation the association graph is substituted with a matrix $B = (b_{ij})_{i,j \in V}$ is related to the weights and connectivity of the graph by the relation

$$b_{ij} = \begin{cases} 1/w_i & \text{if } i = j \\ k_{ij} \geq \frac{b_{ii}+b_{jj}}{2} & \text{if } (i,j) \notin E \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

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 $S \subseteq V$ and its *characteristic* vector $\mathbf{x}^S$ defined as

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

S is a maximum (maximal) weight clique if and only if $\mathbf{x}^S$ is a global (local) minimizer for equation $\mathbf{x}^T B\mathbf{x}$. Furthermore, the weight of the clique S is $w(S) = \frac{1}{\mathbf{x}^{S^T} B\mathbf{x}^S}$.

Unfortunately, under this formulation, the minima are not necessarily isolated: 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. What this implies is that, if we find a minimizer $\mathbf{x}^*$ we can derive the weight of the clique, but we might not be able to tell the nodes that constitute it.

Bomze, Pelillo and Stix [3] 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

$$c_{ij} = \begin{cases} \frac{1}{2w_i} & \text{if } i = j \\ k_{ij} \geq c_{ii} + c_{jj} & \text{if } (i,j) \notin E, i \neq j \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Once again, S is a maximum (maximal) weighted clique if and only if $\mathbf{x}^S$ 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 $\mathbf{e} = (1, \cdots, 1)^T$ is the vector with every component equal to 1 and γ is a positive scaling constant.

References

1. H. G. Barrow and R. M. Burstall, Subgraph isomorphism, matching relational structures and maximal cliques, *Inf. Proc. Letter*, Vol. 4, pp.83, 84, 1976.
2. M. Bartoli et al., Attributed tree homomorphism using association graphs, In *ICPR*, 2000.
3. I. M. Bomze, M. Pelillo, and V. Stix, Approximating the maximum weight clique using replicator dynamics, *IEEE Trans. on Neural Networks*, Vol. 11, 2000.
4. H. Bunke and A. Kandel, Mean and maximum common subgraph of two graphs, *Pattern Recognition Letters*, Vol. 21, pp. 163-168, 2000.

5. W. J. Christmas and J. Kittler, Structural matching in computer vision using probabilistic relaxation, *PAMI*, Vol. 17, pp. 749-764, 1995.
6. M. A. Eshera and K-S Fu, An image understanding system using attributed symbolic representation and inexact graph-matching, *PAMI*, Vol 8, pp. 604-618, 1986.
7. L. E. Gibbons et al., Continuous characterizations of the maximum clique problem, *Math. Oper. Res.*, Vol. 22, pp. 754-768, 1997
8. B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker, Shapes, shocks, and deforamtions I, *International Journal of Computer Vision*, Vol. 15, pp. 189-224, 1995.
9. B. Luo, et al., Clustering shock trees, submitted 2001.
10. T. S. Motzkin and E. G. Straus, Maxima for graphs and a new proof of a theorem of Turán, *Canadian Journal of Mathematics*, Vol. 17, pp. 533-540, 1965.
11. M. Pelillo, Replicator equations, maximal cliques, and graph isomorphism, *Neural Computation*, Vol. 11, pp.1935-1955, 1999.
12. M. Pelillo, The dynamics of relaxation labeling process, *J. Math. Imaging Vision*, Vol. 7, pp. 309-323, 1997.
13. A. Shokoufandeh, S. J. Dickinson, K. Siddiqi, and S. W. Zucker, Indexing using a spectral encoding of topological structure, In *CVPR*, 1999.
14. K. Siddiqi, S. Bouix, A. Tannenbaum, and S. W. Zucker, The hamilton-jacobi skeleton, In *ICCV*, pp. 828-834, 1999.
15. K. Siddiqi et al., Shock graphs and shape matching, *Int. J. of Comp. Vision*, Vol. 35, pp. 13-32, 1999.
16. K-C Tai, The tree-to-tree correction problem, *J. of the ACM*, Vol. 26, pp. 422-433, 1979.
17. A. Torsello and E.R. Hancock, A skeletal measure of 2D shape similarity, *Int. Workshop on Visual Form*, 2001.
18. W. H. Tsai and K. S. Fu, Error-correcting isomorphism of attributed relational graphs for pattern analysis, *Sys., Man, and Cyber.*, Vol. 9, pp. 757-768, 1979.
19. J. T. L. Wang, K. Zhang, and G. Chirn, The approximate graph matching problem, In *ICPR*, pp. 284-288, 1994.
20. R. C. Wilson and E. R. Hancock, Structural matching by discrete relaxation, *PAMI*, 1997.
21. K. Zhang, A constrained edit distance between unordered labeled trees, *Algorithmica*, Vol. 15, pp. 205-222, 1996.
22. K. Zhang and D. Shasha, Simple fast algorithms for the editing distance between trees and related problems, *SIAM J. of Comp.*, Vol. 18, pp. 1245-1262, 1989.
23. K. Zhang, R. Statman, and D. Shasha, On the editing distance between unorderes labeled trees, *Inf. Proc. Letters*, Vol. 42, pp. 133-139, 1992.