References

1. R. C. Wilson A. M. Finch and E. R. Hancock. An energy function and continuous edit process for graph matching. *Neural Computation*, 10(7):1873–1894, 1998.
2. K. Siddiqi A. Shokoufandeh, S. J. Dickinson and S. W. Zucker. Indexing using a spectral encoding of topological structure. In *Proceedings of the Computer Vision and Pattern Recognition*, 1998.
3. Luo Bin and E. R. Hancock. Procrustes alignment with the em algorithm. In *8th International Conference on Computer Analysis of Images and Image Patterns*, pages 623–631, 1999.
4. H. Buke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18, 1997.
5. Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.
6. M. A. Eshera and K. S. Fu. A graph distance measure for image analysis. *SMC*, 14(3):398–408, May 1984.
7. S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *PAMI*, 18(4):377–388, April 1996.
8. R. Horaud and H. Sossa. Polyhedral object recognition by indexing. *Pattern Recognition*, 1995.
9. L. Lovász. Random walks on graphs: a survey. *Bolyai Society Mathematical Studies*, 2(2):1–46, 1993.
10. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys. Dokl.*, 6:707–710, 1966.
11. Bin Luo and E. R. Hancock. Structural graph matching using the EM algorithm and singular value decomposition. *To appear in IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2001.
12. B. J. Oommen and K. Zhang. The normalized string editing problem revisited. *PAMI*, 18(6):669–672, June 1996.
13. A. Sanfeliu and K. S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 13:353–362, 1983.
14. G. Scott and H. Longuet-Higgins. An algorithm for associating the features of two images. In *Proceedings of the Royal Society of London*, number 244 in B, 1991.
15. L. G. Shapiro and R. M. Haralick. Relational models for scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4:595–602, 82.
16. L. S. Shapiro and J. M. Brady. A modal approach to feature-based correspondence. In *British Machine Vision Conference*, 1991.
17. S. Ullman. Filling in the gaps. *Biological Cybernetics*, 25:1–6, 76.
18. S. Umeyama. An eigen decomposition approach to weighted graph matching problems. *PAMI*, 10(5):695–703, September 1988.
19. R. S. Varga. *Matrix Iterative Analysis*. Springer, second edition, 2000.
20. R. A. Wagner. The string-to-string correction problem. *Journal of the ACM*, 21(1), 1974.
21. J. T. L. Wang, B. A. Shapiro, D. Shasha, K. Zhang, and K. M. Currey. An algorithm for finding the largest approximatelycommon substructures of two trees. *PAMI*, 20(8):889–895, August 1998.
22. R. C. Wilson and E. R. Hancock. Structural matching by discrete relaxation. *PAMI*, 19(6):634–648, June 1997.

Learning Structural Variations in Shock Trees

Andrea Torsello and Edwin R. Hancock

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

Abstract. In this paper we investigate how to construct a shape space for sets of shock trees. To do this we construct a super-tree to span the union of the set of shock trees. We learn this super-tree and the correspondences of the node in the sample trees using a maximizing likelihood approach. We show that the likelihood is maximized by the set of correspondences that minimizes the sum of the tree edit distance between pair of trees, subject to edge consistency constraints. Each node of the super-tree corresponds to a dimension of the pattern space. Individual such trees are mapped to vectors in this pattern space.

1 Introduction

Recently, there has been considerable interest in the structural abstraction of 2D shapes using shock-graphs [9]. The shock-graph is a characterization of the differential structure of the boundaries of 2D shapes. Although graph-matching allows the pairwise comparison of shock-graphs, it does not allow the shape-space of shock-graphs to be explored in detail. In this paper we take the view that although the comparison of shock-graphs, and other structural descriptions of shape, via graph matching or graph edit distance has proved effective, it is in some ways a brute-force approach which is at odds with the non-structural approaches to recognition which have concentrated on constructing shape-spaces which capture the main modes of variation in object shape. Hence, we aim to address the problem of how to organize shock-graphs into a shape-space in which similar shapes are close to one-another, and dissimilar shapes are far apart. In particularly, we aim to do this in a way such that the space is traversed in a relatively uniform manner as the structures under study are gradually modified. In other words, the aim is to embed the graphs in a vector-space where the dimensions correspond to principal modes in structural variation.

There are a number of ways in which this can be achieved. The first is to compute the edit-distance between shock-graphs and use multidimensional scaling to embed the individual graphs in a low-dimensional space [6]. However, as pointed out above, this approach does not necessarily result in a shape-space where the dimensions reflect the modes of structural variation of the shock-graphs. Furthermore, pairwise distance algorithms consistently underestimate the distance between shapes belonging to different clusters. When two shapes are similar, the node-correspondences can be estimated reliably, but as shapes move farther

T. Caelli et al. (Eds.): SSPR&SPR 2002, LNCS 2396, pp. 113–122, 2002.

apart in shape space the estimation becomes less reliable. This is due to the fact that correspondences are chosen to minimize the distance between trees: as the shock-trees move further apart the advantage the "correct" correspondence has over alternative ones diminishes. Until, eventually, a match which yields a lower distance is selected. The result of this is a consistent underestimation of the distance as the shapes move further apart in shape space.

The second approach is to extract feature vectors from the graphs and use these as a shape-space representation. A shape-space can be constructed from such vectors by performing modal analysis on their covariance matrix. However, when graphs are of different size, then the problem of how to map the structure of a shock-graph to a vector of fixed length arises. It may be possible to circumvent the problem using graph spectral features.

In this paper we take a different approach to the problem. We aim to embed shock trees in a pattern space by mapping them to vectors of fixed length. We do this as follows. We commence from a set of shock-trees representing different shapes. From this set we learn a super-tree model of which each tree can be considered a noisy sample. In particular, we assume that each node feature is detected with a probability that depends on its weight, but that the hierarchical relation between two detected nodes is always correct. That is, our model has every possible node and the sampling error is in the existence of nodes in our samples, not in their relational structure. Hence, the structure of each sample tree can be obtained from the structure of the super-tree with node removal operations only. We learn this super-tree and the correspondences between the nodes in the sample trees using a maximizing likelihood approach. We show that the likelihood is maximized by the set of correspondences that minimizes the sum of the tree edit distance between pair of trees, subject to edge consistency constraints. To embed the individual shock-trees in a vector-space we allow each node of the super-tree to represent a dimension of the space. Each shock-tree is represented in this space by a vector which has non-zero components only in the directions corresponding to its constituent nodes. The non-zero components of the vectors are the weights of the nodes. In this space, the edit distance between trees is the L1 norm between their embedded vectors.

2 Tree Edit-Distance

The idea behind edit distance is that it is possible to identify a set of basic edit operations on nodes and edges of a structure, and to associate with these operations a cost. The edit-distance is found by searching for the sequence of edit operations that will make the two graphs isomorphic with one-another and which has minimum cost. 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. Transforming node insertions in one tree into node removals in the other allows us to use only structure reducing operations. This, in turn, means that the edit distance between two

trees is completely determined by the subset of nodes left after the optimal removal sequence. In this section we show how to find the set correspondences that minimizes the edit distance between two trees. To find he edit distance we make use of results presented in [10]. We call $\mathcal{C}(t)$ the closure of tree t, $E_v(t)$ the edit operation that removes node v from t and $\mathcal{E}_v(\mathcal{C}(t))$ the equivalent edit operation that removes v from the closure. The first result is that edit and closure operations commute: $\mathcal{E}_v(\mathcal{C}(t)) = \mathcal{C}(E_v(t))$. For the second result we need some more definitions: We call a subtree s of $\mathcal{C}t$ *obtainable* if for each node v of s if there cannot be two children a and b so that (a, b) is in $\mathcal{C}t$. In other words, for s to be obtainable, there cannot be a path in t connecting two nodes that are siblings in s. We can, now, introduce the following:

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

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

We commence by transforming the problem from the search of the minimum edit cost linked to the removal of some nodes, to the maximum of a utility function linked to the nodes that are retained. To do this we assume that we have a weight w_i assigned to each node i, that the cost of matching a node i to a node j is $|w_i - w_j|$, and that the cost of removing a node is equivalent to matching it to a node with weight 0. We define the set $M \subset \mathcal{N}^t \times \mathcal{N}^{t'}$ the set of pair of nodes in t and t' that match, the set $L^t = \{i \in \mathcal{N}^t | \forall x, < i, x > \notin M\}$ composed of nodes in the first tree that are not matched to any node in the second, and the set $R^{t'} = \{j \in \mathcal{N}^{t'} | \forall x, < x, j > \notin M\}$, which contains the unmatched nodes of the second tree. With these definitions the edit distance becomes:

$$d(t,t') = \sum_{i \in L^t} w_i + \sum_{j \in R^{t'}} w_j + \sum_{<i,j> \in M} |w_i - w_j| = \\ = \sum_{i \in \mathcal{N}^t} w_i + \sum_{j \in \mathcal{N}^{t'}} w_j - 2 \sum_{<i,j> \in M} \min(w_i, w_j). \quad (1)$$

We call he *utility* of the match M. the quantity

$$\mathcal{U}(M) = \sum_{<i,j> \in M} \min(w_i, w_j).$$

Clearly the match that maximizes the utility minimizes the edit distance. That is, Let $O \subset \mathcal{P}(\mathcal{N}t \times \mathcal{N}t')$ be the set of matches that satisfy the obtainability constraint, the node correspondence $M^* = (N_t^*, N_{t'}^*)$ is

$$M^* = \underset{M \in O}{\operatorname{argmax}} \mathcal{U}(M),$$

and the closure of the MCOS is the restriction to N_t^* of $\mathcal{C}t$.

Let us assume that we know the utility of the best match rooted at every descendent of v and w. We aim to find the set of siblings with greatest total utility. To do this we make use of a derived structure similar to the association graph introduced by Barrow in [1]. The nodes of this structure are pairs drawn from the Cartesian product of the descendents of v and w and each pair correspond to a particular association between a node in one tree to a node in the other. We connect two such associations if and only if there is no inconsistency between the two associations, that is the corresponding subtree is obtainable. Furthermore, we assign to each association node (a, b) a weight equal to the utility of the best match rooted at a and b. The maximum weight clique of this graph is the set of consistent siblings with maximum total utility, hence the set of children of v and w that guarantee the optimal isomorphism. Given a method to obtain a maximum weight clique, we can use it to obtain the solution to our isomorphism problem. We refer again to [10] for heuristics for the weighted clique problem.

3 Edit-Intersection and Edit-Union

The edit distance between two trees is completely determined by the set of nodes that do not get removed by edit operations, that is, in a sense, the *intersection* of the sets of nodes. Furthermore, the distance, and hence, the intersection, determines the probability of a match. We would like to extend the concept to more than two trees so that we can compare a shape tree to a whole set T of trees. Moreover, this allows us to determine how a new sample relates to a previous distribution of trees.

Formally, we assume that the set T of tree samples is drawn from an unknown distribution of trees τ that we want to learn. We assume that we have no sampling error in the detection of the hierarchical relation between two nodes in a sample, that is if we detect two nodes, we detect them with the correct ancestor descendent relation. On the other hand, we assume an exponential distribution for the node weight for a node i of tree t:

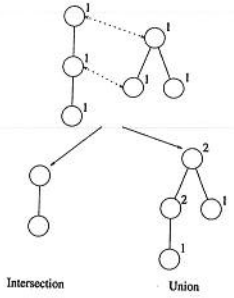

Fig. 1. Union and intersection of trees

$$p_i^t(x) = k \exp\left[-|x - \theta_i^t|\right],$$

where θ_i is a parameter of node i's weight distribution we want to estimate, and k is a normalizing constant.

The log-likelihood function based on the samples T and the set of nodes parameters $\Omega = \{\omega_i\}$ is

$$\mathcal{L} = \sum_{t \in T} \sum_{i \in \mathcal{N}(t)} \log p_i^t = \sum_{t \in T} \sum_{i \in \mathcal{N}(t)} \log k - \sum_{t \in T} \sum_{i \in \mathcal{N}(t)} |w_i^t - \theta_i^t|.$$

We can estimate θ assuming we know the correspondences $C(t, s) \subseteq \mathcal{N}(t) \times \mathcal{N}(s)$ between two trees $t, s \in T$. Fixing this correspondences and estimating θ, we can

write the variable part of the log-likelihood function as:

$$\widehat{\mathcal{L}} = -\sum_{t \in T} \sum_{s \in T} \sum_{<i,j> \in C(t,s)} |w_i^t - w_j^s|.$$

The structure we want to learn must maximize this function, subject to a consistency constraint on the correspondences. That is if node a in tree t_1 is matched to node b in tree t_2 and to node c in tree t_3, then b must be matched to c, i.e.

$$< a, b > \in C(t_1, t_2) \wedge < a, c > \in C(t_1, t_3) \Rightarrow < b, c > \in C(t_2, t_3).$$

To find the match we calculate a *union* of the nodes: a structure from which we can obtain any tree in our set removing appropriate nodes, as opposed to the intersection of nodes, which is a structure that can obtained removing nodes from the original trees (see Figure 1). Any such structure has the added advantage of implicitly creating an embedding space for our trees: assigning to each node a coordinate in a vector space V, we can associate each tree t to a vector $v \in V$ so that $v_i = w_i^t$, where w_i^t is the weight of the node of t associated with node i of the union, $w_i^t = 0$ if no node in t is associated with i.

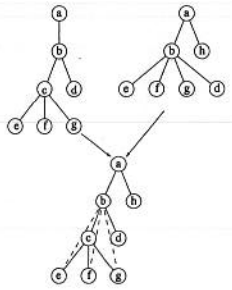

Fig. 2. Edit-union of two trees

The structure of the union of two trees is completely determined by the set of matched nodes: it can be obtained by iteratively merging the nodes of the two trees that are matched. The result will be a directed acyclical graph with multiple paths connecting various nodes (see Figure 2). This structure, thus, has more links than necessary and cannot be obtain from the first tree by node removal operations alone. Removing the superfluous edges. we obtain a tree starting from which we can obtain either one of the original trees by node removal operations alone. Furthermore, this reduced structure maintains the same transitive closure, hence the same hierarchical relation between nodes.

Since the node weights are positive, we can rewrite the variable component of the log-likelihood function as:

$$\widehat{\mathcal{L}} = -\sum_{t \in T} \sum_{s \in T} \sum_{i \in \mathcal{N}(t)} w_i^t - \sum_{t \in T} \sum_{s \in T} \sum_{j \in \mathcal{N}(s)} w_j^s + 2 \sum_{t \in T} \sum_{s \in T} \sum_{<i,j> \in M(t,s)} \min(w_i^t, w_j^s),$$

where $M(t, s)$ is the set of matches between the nodes of the trees t and s. From this we can see that the set of matches M that maximizes the log likelihood maximizes the sum of the utility functions $\sum_{t \in T} \sum_{s \in T} \mathcal{U}(M(t, s))$ and, hence, minimizes the sum of the edit distances between each pair of samples.

3.1 Joining Multiple Trees

Learning the super-structure, or equivalently finding the structure that minimizes the total distance between trees in the set is computationally infeasible,

but we propose a suboptimal iterative approach that iteratively extends the union adding a new tree to it. We want to find the match between the union and the nodes to be added consistent with the obtainability constraint that minimizes the sum of the edit distance between the new tree and each tree in the set. Unfortunately the union operation is not closed in the set of trees, that is the union is not necessarily a tree since it is not always possible to find a tree such that we can edit it to obtain the original trees. For an example where the union of two trees in not a tree see Figure 3. In this figure α and β are subtrees. Because of the constraints posed by matching trees α and trees β, nodes b and b' cannot be matched and neither b can be a child of b' nor b' a child of b. The only option is to keep the two paths as separate alternatives: this way we can obtain the first tree removing the node b' and the second removing b.

For this reason we cannot use our tree edit distance algorithm unchanged to find the matches between the union and the new tree because it would fail on structures with multiple paths from one node a to node b, counting any match in the subtree rooted at b twice. Fortunately, different paths are present in separate trees and so we can assume that they are mutually exclusive. If we constrain our search to match nodes in only one path and we match the union to a tree, we are guaranteed not to count the same subtree multiple times. Interestingly, this constraint can be merged with the obtainability constraint: we say that a match is *obtainable* if for each node v there cannot be two children a and b and a node c so that there is a path, possibly of length 0, from a to c and one from b to c. This constrain reduces to obtainability for trees when $c = b$, but it also prevents a and b from belonging two to separate paths joining at c. Hence from a node where multiple paths fork, we can extract children matches from one path only.

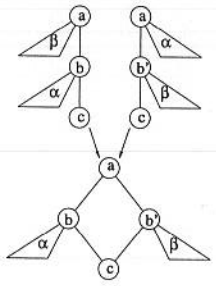

Fig. 3. Edit-union is not a tree

It is worth noting that this approach can be extended to match two union structures, as long as at most one has multiple paths to a node. To do this we iterate through each pair of weights drawn from the two sets, that is, we define the utility as:

$$\mathcal{U}(M) = \sum_{t \in T_1, t' \in T_2} \sum_{<i,j> \in M} \min(w_i^t, w_j^{t'}),$$

where $M \subset \mathcal{N}^{(T_1^\cup)} \times \mathcal{N}^{(T_2^\cup)}$ is the set of matches between the nodes of the union structures $T_1^\cup$ and $T_2^\cup$. The requirement that no more than one union has multiple paths to a node is required to avoid double counting. Solving the modified weighted clique problems we obtain the correspondence between the nodes of the trees in in the two sets.

To be able to calculate the utility we need to keep, for each node in the union structure, the weights of the matched nodes. A way to do this is to assign to each node in the union a vector of dimension equal to the number of trees in the set. The ith coordinate of this vector will be the weight of the corresponding node in the ith tree, 0 if the ith tree doesn't have a node matching to the current node. This representation also allows us to easily obtain the coordinate of each tree in the set in the embedding space induced by the union: the ith weight of the jth node is the jth coordinate of the ith tree.

In order to increase the accuracy of the approximation, we want to merge trees with smaller distance first. This is because we can be reasonably confident that, if the distance is small, the extracted correspondences are correct. We could start with the set of trees, merge the closest two and replace them with the union and reiterate until we end up with only one structure. Unfortunately, since we have no guarantees that the edit-union is a tree, we might end up trying to merge two graphs with multiple paths to a node. For this reason, if merging two trees give a union that is not a tree, we discard the union and try with the next-best match. When no trees can be merged without duplicating paths, we merge the remaining structures always merging the new nodes to the same structure. This way we are guaranteed to merge at each step at most one multi-path graph.

4 Experimental Results

We evaluate the new approach on the problem of shock tree matching. In order to asses the quality of the approach we compare the obtained embeddings with those described in [10,6]. In particular, we compare the the first two principal components of the embedding generated joining purely structural skeletal representations, with 2D multi-dimensional scaling of the pairwise distances of the shock-trees weighted with some geometrical information. The addition of matching consistency across shapes allows the embedding to better capture the structural information present in the shapes, yielding embedding comparable to those provided by localized geometrical information.

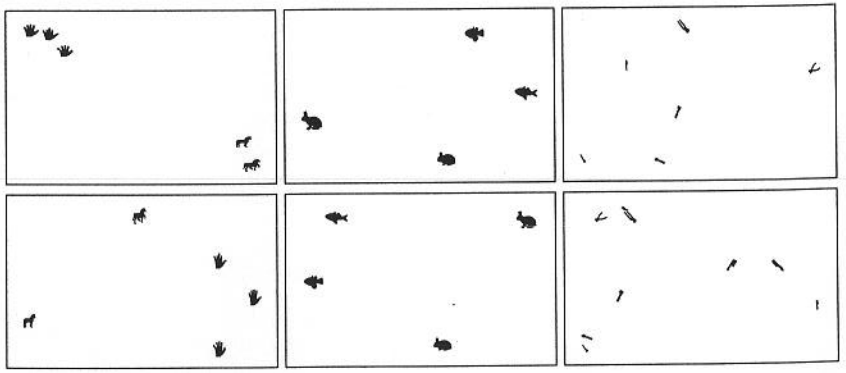

Fig. 4. Top: Embedding through union. Bottom: 2D MDS of pairwise distance

We run three experiments with 4, 5, and 9 shapes each. In each experiment the shapes belong to two or more distinct visual clusters. In order to avoid scaling effect due to difference in the number of nodes, we normalize the embedding vectors so that they have L1 norm equal to 1, and then we extract the first 2 principal components.

Figure 4 shows a comparison between embedding obtained through edit-union of shock trees and through multi-dimensional scaling of the pairwise distances. The first column shows a clear example where the pairwise edit-distances approach underestimate the distance while edit-union keep the clusters well separated. The second and third column show examples where the distance in shape space is not big enough to observe the described behavior, yet the embedding obtained through union fares well against the pairwise edit-distance, especially taking into account the fact that it uses only structural information while the edit-distance matches weight the structure with geometrical information. In particular, the third column shows a better ordering of shapes, with brushes being so tightly packed that they overlap. It is interesting to note how the union embedding puts the monkey wrench (top-center) somewhere in-between pliers and wrenches: the algorithm is able to consistently match the head to the heads of the wrenches, and the handles to the handles of the pliers.

Figure 5 plots the distances obtained through edit union of weighted shock trees (x axis) versus the corresponding pairwise edit distances (y axis). The plot clearly highlights that the pairwise distance approach tends to underestimate the distances between shapes.

4.1 Synthetic Data

To augment these real world experiments, we have performed the embedding on synthetic data. The aim of the experiments is to characterize the ability of the approach to generate a shape space. To meet this goal we have randomly generated some prototype trees and, from each tree, we generated five or ten structurally perturbed copies. 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 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. To perturb the trees we simply add nodes using the same approach.

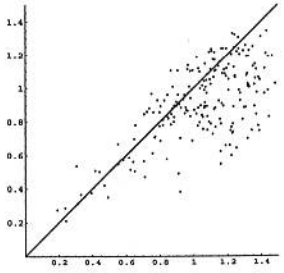

Fig. 5. Edit-union vs. pairwise edit distances

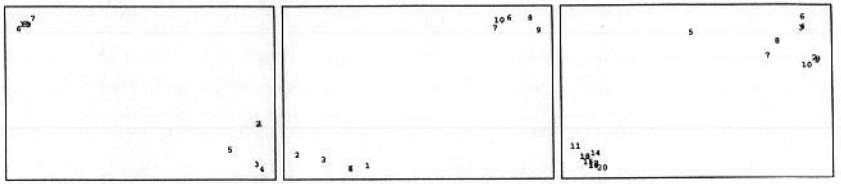

Fig. 6. Synthetic clusters

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

5 Conclusions

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

In a set of experiments we apply this algorithm to match shock graphs, a graph representation of the morphological skeleton. The results of these experiments are very encouraging, showing that the algorithm is able to group similar shapes together in the generated embedding space.

Our future plans are to extend the framework reported in this paper by using the apparatus of variational inference to fit a mixture of trees, rather than a union tree, to the training data. Here we will perform learning by minimizing the Kullback divergence between the training data and the mixture model.

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. H. Bunke and A. Kandel, Mean and maximum common subgraph of two graphs, *Pattern Recognition Letters*, Vol. 21, pp. 163-168, 2000.
3. T. F. Cootes, C. J. Taylor, and D. H. Cooper, Active shape models - their training and application, *CVIU*, Vol. 61, pp. 38-59, 1995.

4. T. Heap and D. Hogg, Wormholes in shape space: tracking through discontinuous changes in shape, *ICCV*, pp. 344-349, 1998.
5. T. Sebastian, P. Klein, and B. Kimia, Recognition of shapes by editing shock graphs, in *ICCV*, Vol. I, pp. 755-762, 2001.
6. B. Luo, et al., Clustering shock trees, in *CVPR*, Vol 1, pp. 912-919, 2001.
7. M. Pelillo, K. Siddiqi, and S. W. Zucker, Matching hierarchical structures using association graphs, *PAMI*, Vol. 21, pp. 1105-1120, 1999.
8. S. Sclaroff and A. P. Pentland, Modal matching for correspondence and recognition, *PAMI*, Vol. 17, pp. 545-661, 1995.
9. K. Siddiqi et al., Shock graphs and shape matching, *Int. J. of Comp. Vision*, Vol. 35, pp. 13-32, 1999.
10. A. Torsello and E. R. Hancock, Efficiently computing weighted tree edit distance using relaxation labeling, in *EMMCVPR*, LNCS 2134, pp. 438-453, 2001
11. K. Zhang, R. Statman, and D. Shasha, On the editing distance between unorderes labeled trees, *Inf. Proc. Letters*, Vol. 42, pp. 133-139, 1992.

A Comparison of Algorithms for Maximum Common Subgraph on Randomly Connected Graphs

Horst Bunke[1], Pasquale Foggia[2], Corrado Guidobaldi[1,2], Carlo Sansone[2], and Mario Vento[3]

[1] Institut für Informatik und angwandte Mathematik,Universität Bern
Neubrückstrasse 10, CH-3012 Bern, Switzerland
bunke@iam.unibe.ch
[2] Dipartimento di Informatica e Sistemistica, Università di Napoli "Federico II"
via Claudio, 21 I-80125 Napoli, Italy
{foggiapa,cguidoba,carlosan}@unina.it
[3] Dipartimento di Ingegneria dell'Informazione ed Ingegneria Elettrica
Università di Salerno, via P.te Don Melillo, 1 I-84084 Fisciano (SA), Italy
mvento@unisa.it

Abstract. A graph g is called a *maximum common subgraph* of two graphs, g_1 and g_2, if there exists no other common subgraph of g_1 and g_2 that has more nodes than g. For the maximum common subgraph problem, exact and inexact algorithms are known from the literature. Nevertheless, until now no effort has been done for characterizing their performance. In this paper, two exact algorithms for maximum common subgraph detection are described. Moreover a database containing randomly connected pairs of graphs, having a maximum common graph of at least two nodes, is presented, and the performance of the two algorithms is evaluated on this database.

1 Introduction

Graphs are a powerful and versatile tool useful in various subfields of science and engineering. There are applications, for example, in pattern recognition, machine learning and information retrieval, where one needs to measure the similarity of objects. If graphs are used for the representation of structured objects, then measuring the similarity of objects becomes equivalent to determining the similarity of graphs.

There are some well-known concepts that are suitable graph similarity measures. Graph isomorphism is useful to find out if two graphs have identical structure[1]. More generally, subgraph isomorphism can be used to check if one graph is part of another [1,2]. In two recent papers [3,4], graph similarity measures based on maximum common subgraph and minimum common supergraph have been proposed.

Detection of the maximum common subgraph (MCS) of two given graphs is a well-known problem. In [5], such an algorithm is described and in [6] the use of this algorithm in comparing molecules has been discussed. In [7] a MCS algorithm that uses a backtrack search is introduced. A different strategy for deriving the MCS first

T. Caelli et al. (Eds.): SSPR&SPR 2002, LNCS 2396, pp. 123-132, 2002.