An Error-Tolerant Approximate Matching Algorithm for Attributed Planar Graphs and Its Application to Fingerprint Classification

Michel Neuhaus and Horst Bunke

Department of Computer Science, University of Bern
Neubrückstrasse 10, CH-3012 Bern, Switzerland
{mneuhaus,bunke}@iam.unibe.ch

Abstract. Graph edit distance is a powerful error-tolerant similarity measure for graphs. For pattern recognition problems involving large graphs, however, the high computational complexity makes it sometimes impossible to apply edit distance algorithms. In the present paper we propose an efficient algorithm for edit distance computation of planar graphs. Given graphs embedded in the plane, we iteratively match small subgraphs by locally optimizing structural correspondences. Eventually we obtain a valid edit path and hence an upper bound of the edit distance. To demonstrate the efficiency of our approach, we apply the proposed algorithm to the problem of fingerprint classification.

1 Introduction

In recent years graphs have been recognized as a powerful concept to represent structural patterns. Similarity measures for graphs that are based on an exact structural correspondence such as graph isomorphism and maximum common subgraph are often elegant and quite efficient [1–3]. For real applications, however, it is often difficult to find a graph representation that deals sufficiently well with structural variations between graphs from the same class. Graph matching procedures that allow for such structural variations, so-called error-tolerant algorithms, have been introduced with the development of the graph edit distance [4, 5]. The edit distance of graphs is computed by determining the least costly way to edit one graph into another, given an underlying set of edit operations on graphs and their costs. Due to the enormous computational complexity of the matching problem for general graphs, a number of authors have studied special classes of graphs, such as trees, bounded-valence graphs, and graphs with unique node labels [6–8].

In the present paper we focus on the problem of efficiently matching large attributed planar graphs in the context of the edit distance framework. Planar graphs are interesting in many applications involving images, because common graph representations extracted from an image are planar. A well-known example is region adjacency graphs [9].

In Section 2 of this paper the graph edit distance terminology is introduced and in Section 3 the proposed approximate distance algorithm for planar graphs

A. Fred et al. (Eds.): SSPR&SPR 2004, LNCS 3138, pp. 180–189, 2004.

is described. Next, in Section 4, we demonstrate how planar graph matching can be applied to the fingerprint classification problem and present experimental results. Finally, conclusions are provided in Section 5.

2 Graph Edit Distance

Graph edit distance is an error-tolerant similarity measure for graphs [4, 5]. Structural variations between graphs are modeled with a set of edit operations such as node insertion, node deletion, node substitution, edge insertion, edge deletion, and edge substitution. The key concept is to describe structural differences with the sequence of edit operations that best explain the variations. For this purpose it is common to assign costs to edit operations such that they reflect the strength of the corresponding distortion. The edit distance $d(G, G')$ of two graphs G and G' is then defined as the cost of the least expensive edit path that transforms G into G'. Theoretically, every node of G could be matched to every node of G', as edit operations are defined such that they are able to correct any structural error, and a straight-forward pruning criterion (such as the one for graph isomorphism) does not exist. Hence, it is easy to observe that the computational complexity of the graph edit distance algorithm is exponential in the number of nodes involved. Nonetheless, for small graphs it has proven a powerful graph similarity measure [9, 10]. But for large graphs it becomes computationally infeasible due to its high running time and memory complexity.

3 Approximate Planar Graph Edit Distance

In order to overcome the difficulties arising from the high computational complexity, we propose an approximate, but efficient algorithm for the computation of the edit distance for attributed planar graphs. In the following we assume that our data graphs are provided with a planar embedding, that is, a drawing of the graph in the plane such that none of its edges intersect. An example is shown in Fig. 1. In contrast to exact graph edit distance computation, which defines the distance in terms of the least expensive of all edit paths, we restrict the number of possible edit operations and determine the least expensive member of a smaller set of candidate edit paths. This set of candidate paths is obtained in the course of a process that embeds the graphs under consideration in the plane. If the candidate generation process produces an edit path that is close the the optimal path, the planar edit distance will approximate the graph edit distance well.

For the description of the generation process of the candidate paths we need the following definition. The *neighborhood* of a node u in a graph is defined as the subgraph consisting of node u, all nodes connected to u, and all edges between these nodes. More formally, if we denote a graph by $G = (V, E, \alpha, \beta)$, where V is the set of nodes, E the set of directed edges, $\alpha : V \rightarrow L_V$ the node labeling function, and $\beta : E \rightarrow L_E$ the edge labeling function, the neighborhood $N(u)$ of u in G is defined as the induced subgraph $N(u) = (V_u, E_u, \alpha_u, \beta_u)$ of G, where

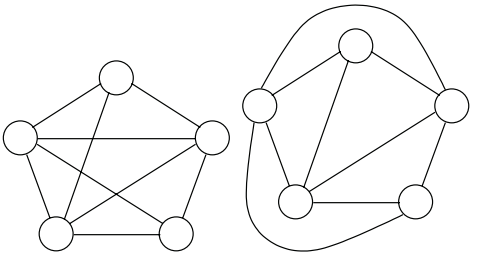


Fig. 1. Illustration of a) a planar graph and b) the same graph embedded in the plane

$$
\begin{aligned}
V_u &= \{u\} \cup \{v \in V | (v,u) \in E \text{ or } (u,v) \in E\} \\
E_u &= E \cap (V_u \times V_u) \\
\alpha_u &= \alpha|_{V_U} \\
\beta_u &= \beta|_{E_U} \quad .
\end{aligned}
$$

An illustration of a neighborhood is shown in Fig. 2. Note that the embedding of the planar graph is preserved in the neighborhood, that is, there is an order defined on the nodes connected to u.

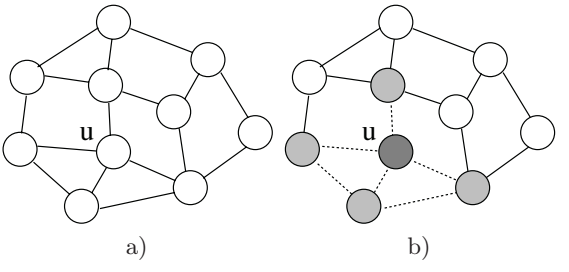


Fig. 2. a) Planar graph and b) graph with marked neighborhood of u

In order to initialize the generation of a candidate path in the process of matching graphs G and G', a *seed substitution* $u \rightarrow u'$ has to be chosen, where u is a node from G and u' a node from G'. Next an optimal matching from subgraph $N(u)$ to subgraph $N'(u')$ (where symbol N refers to graph G and symbol N' to graph G') based on the underlying set of edit operations is to be determined. All new substitutions that occur in this matching are marked for further processing. In consecutive steps the neighborhoods belonging to unprocessed substitutions are processed in the same manner, where substitutions that were previously obtained are preserved in subsequent neighborhood matchings. The matching begins with the seed neighborhood and is iteratively expanded across the two graphs. The result of this procedure is a valid edit path from the first to the second graph. The algorithm is outlined in Table 1.

Table 1. Planar edit distance algorithm

Input: Two planar graphs $G = (V, E, \alpha, \beta)$ and $G' = (V', E', \alpha', \beta')$ to be matched.
Output: A matching between G and G' and the corresponding edit distance, $d(G, G')$

0. Determine seed substitution $u_0 \rightarrow u'_0$
1. Add seed substitution $u_0 \rightarrow u'_0$ to the FIFO queue Q
2. Fetch next substitution $u \rightarrow u'$ from Q
3. Match neighborhood $N(u)$ to neighborhood $N'(u')$
4. Add new substitutions occurring in step 3 to Q
5. If Q is not empty, go to step 2
6. Delete all unprocessed nodes and edges in both G and G'

Let us consider step 3 of the algorithm, the neighborhood matching, more closely. A neighborhood consists of a center node, a set of adjacent nodes, and edges between these nodes. The set of adjacent nodes can be considered an ordered sequence of nodes due to the planar embedding of the neighborhood. In order to obtain such a node sequence, we randomly start at an adjacent node and traverse all nodes in a clockwise manner. Instead of regarding a neighborhood as a graph to be matched, we can represent a neighborhood as an ordered node sequence and match two neighborhoods simply by finding an optimal node alignment. With this restriction we assume that the optimal neighborhood matching preserves the ordering of the nodes adjacent to the center node. The node alignment can be performed with a cyclic string matching algorithm [11–15], where the sequence of nodes is regarded as a string and the string edit operation costs are derived from the corresponding graph edit operation costs. If we consider graphs with a bounded valence of v, this procedure takes $O(v^2)$. The algorithm terminates after $O(n)$ loops, where n denotes the number of nodes in the graphs. The computational complexity of string matching can further be reduced by preserving previously matched nodes. If we consider a string substitution $u \rightarrow u'$, we require that its operation costs amount to zero if $u \rightarrow u'$ has occurred previously, to infinity if a substitution $u \rightarrow v'$ or $v \rightarrow u'$ with $u \neq v$ and $u' \neq v'$ has occurred previously, and to graph edit operation costs $c(u \rightarrow u')$ otherwise. This means that the present edit path must never be violated by newly added edit operations.

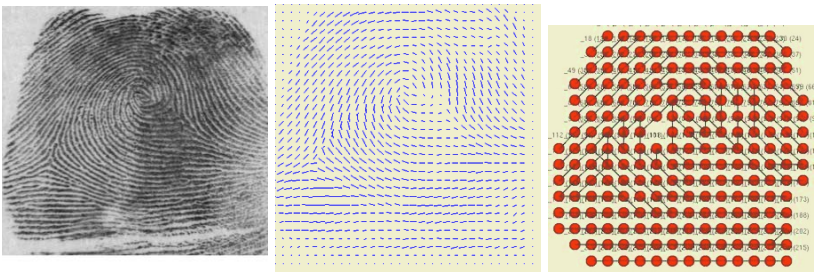
The optimality of the neighborhood matching is determined with respect to the original graph edit operations. New edit operations matching previously obtained operations are added to the edit path in every neighborhood matching. When the algorithm terminates, the generation process yields a valid edit path. The approximate distance value is therefore an upper bound of the true graph edit distance. Since the resulting edit path strongly depends on the seed substitution, we suggest to use several planar distance computations with different seed substitutions and choose the one that returns the minimum matching costs. Promising seed substitution candidates can for instance be found close to the barycenter of the planar embedding in both graphs or may be determined with a local graph matching. If knowledge of the underlying application is available, it may also be utilized to find seed substitution candidates.

4 Application to Fingerprint Classification

Fingerprint recognition tasks can coarsly be divided into verification (one-to-one matching), identification (one-to-many matching), and classification. Fingerprint classification refers to the process of assigning fingerprints to classes with similar characteristics. A large number of fingerprint classification approaches have been reported in the literature, including rule-based [16, 17], syntactic [18], statistical [19], and neural-network-based [20] algorithms. Structural pattern recognition seems to be particularly well suited to the classification problem, as fingerprint analysis naturally involves the comparison of ridge and valley structures. For instance, Maio and Maltoni [9] segment the orientation field of ridge lines into homogeneous regions and convert these into a region adjacency graph. The classification is then performed with an edit distance algorithm. Due to the nature of the segmentation process, the resulting graphs are guaranteed to contain at most ten nodes. Marcialis et al. [21] describe how to improve classification results by fusing this structural algorithm with a statistical classification algorithm. In the present paper, we propose to use larger graphs for the description of the orientation field. Instead of segmenting the orientation field, we combine orientation vectors in a window of constant size and represent them as a single node. In the following, the graph extraction and classification procedure is described in detail. Experimental results are reported in Section 5.

In our fingerprint experiments we use a subset of 450 fingerprints from the NIST-4 database [22]. This database consists of 2000 pairs of grayscale fingerprint images that are classified into one of the classes *arch*, *tented arch*, *left loop*, *right loop*, and *whorl*. An example of a *whorl* image is depicted in Fig. 3a. The image background is segmented from the foreground by computing the grayscale variance in a window around each pixel. The pixels that exhibit a variance lower than a threshold are considered background. For each pixel we then estimate the discrete gradient of the grayscale surface by applying a Sobel operator in the vertical and horizontal direction. After a smoothing process we obtain a ridge orientation field as illustrated in Fig. 3b. Then we represent each pixel in a window as a graph node without attributes. From every node an edge is generated in those two, out of eight, possible directions that best match the vector orthogonal to the average window gradient. A single discrete attribute $\gamma \in \{1, 2, \ldots, 8\}$ is attached to every edge representing the orientation of the edge. The size of the resulting graph depends on the size of the pixel window. In Fig. 3c such a graph is illustrated. The 450 fingerprint graphs from the NIST-4 subset contain an average of 174 nodes and 193 edges per graph at a resolution of 32×32 pixels per window.

We use a simple edit cost function that assigns constant costs p_n to node insertions and deletions, and constant costs p_e to edge insertions and deletions. As nodes are unlabeled, there is no cost for node substitutions, and edge substitution costs are set proportional to the distance of the two involved angles, $d(\gamma, \gamma') = \min\{(\gamma - \gamma') \bmod 8, (\gamma' - \gamma) \bmod 8\}$, for $\gamma, \gamma' \in \{1, 2, \ldots, 8\}$. The ratio of the edge insertion and deletion penalty p_e and the edge substitution cost p_s, i.e. $2p_e/p_s$, determines when an edge deletion followed by an edge insertion is less expensive than an edge substitution.



a) b) c)

Fig. 3. a) NIST-4 whorl image `f0011`, b) averaged ridge orientation field, and c) orientation graph

Table 2. Running time of exact graph edit distance algorithm (GED, 1 run) and planar edit distance approximation (PED, 50 runs) — empty entries indicate failure due to lack of memory

Nodes	GED	PED
5	<1s	<1s
7	<1s	<1s
9	9s	1s
12	-	1s
20	-	1s
30	-	2s
42	-	5s
169	-	15s

The fingerprint classification is performed by evaluating distances of unknown input graphs to labeled prototype graphs. We adopt a nearest-neighbor paradigm and classify graphs according to a maximum similarity, or minimum edit distance, criterion with respect to the prototype graphs. Note that, with this classification procedure, we rather intend to demonstrate the applicability of the approximate planar edit distance algorithm than provide a thoroughly optimized fingerprint classification system.

5 Experimental Results

To evaluate the running time of the approximate algorithm for planar edit distance computation, we perform the standard graph edit distance computation and the planar edit distance computation for the same pair of graphs. The standard graph edit distance is a deterministic algorithm that yields the exact distance value, whereas the planar edit distance approximation requires several runs to be carried out. The results of several distance computations for pairs of fingerprint graphs are shown in Table 2. For small graphs with less than 10 nodes and edges, the exact graph edit distance computation is computationally

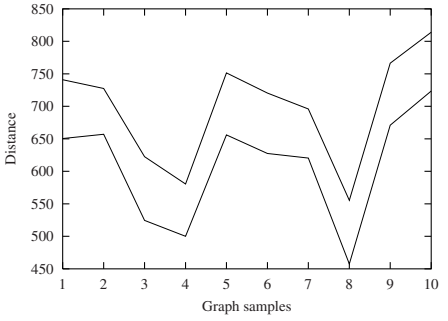


Fig. 4. Exact graph edit distance (lower curve) and approximated planar edit distance (upper curve) for 10 graphs and subgraphs with 10 nodes

feasible. For larger graphs, however, the edit distance search tree exceeds the memory capacity of our testing machine (1024MB). The planar edit distance, on the other hand, provides a result for every tested graph pair, taking only a few seconds for all 50 runs.

Due to memory contraints, the exact edit distance cannot be computed for large graphs. It is therefore difficult to directly evaluate the accuracy of the approximation algorithm. If we delete some nodes from a given graph, however, we obtain a pair of graphs for which a minimum cost edit path is known, so that we can easily compute the exact edit distance between these graphs. The planar edit distance approximation for these graphs is computed in the usual manner without utilizing any knowledge of the special form of the sample graphs. In our first experiment, we delete all but the 10 nodes located closest to the barycenter of the planar embedding from a fingerprint graph and match the resulting graph with the original one. In the second experiment, we use the same procedure to construct subgraphs with 100 nodes. The resulting (known) exact edit distance and the (computed) approximate distance of the first 10 pairs of graphs from NIST-4 are illustrated in Fig. 4. As expected the approximation yields an upper bound of the exact distance. Interestingly enough, the approximation seems to closely follow the exact distance up to an additive constant. If we compute the empiric correlation coefficient of the approximated and the exact distance of the first 100 graphs from NIST-4, we obtain a coefficient of $r = 0.99$ for the subgraphs with 10 nodes and $r = 0.85$ for the subgraphs with 100 nodes. This result indicates that the approximated and the exact distance are strongly correlated in a linear way. In Fig. 5, the correlation can clearly be observed. A regression analysis of the exact distance x and the approximation y according to the linear model $y = \alpha x + \beta$ yields a slope of $\alpha = 0.99$ and an offset of $\beta = 93$ for subgraphs with 10 nodes, and a slope of $\alpha = 1.10$ and an offset of $\beta = 803$ for subgraphs with 100 nodes. A slope of approximately $\alpha = 1$ is equivalent to the reduced linear regression model $y = x + \beta$. We conclude that the difference of the approximation and the exact distance (as illustrated in Fig. 4) is almost

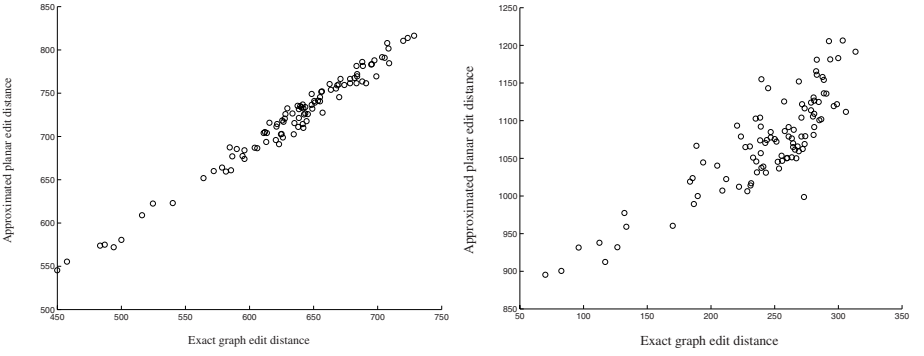


Fig. 5. Exact graph edit distance and approximated planar edit distance for subgraphs with 10 nodes (left) and subgraphs with 100 nodes (right)

Table 3. Fingerprint classification recognition rates per class

Class	Recognition rate
Arch	62.5%
Tented arch	72.5%
Left loop	77.5%
Right loop	85%
Whorl	90%

constant and that the approximation therefore reflects the structural similarity of the underlying graphs well.

In our third experiment we test the applicability of the proposed planar edit distance to the problem of fingerprint classification. The experiment proceeds as follows. For each of the five classes *arch*, *tented arch*, *left loop*, *right loop*, and *whorl* we randomly select 40 input graphs to be classified and another 50 graphs representing the respective fingerprint category. This results in a test set of size 200 and a training, or prototype, set of size 250 graphs. By computing the approximate planar edit distance, we obtain a similarity value between each input graph and each prototype and classify the input graph with a nearest-neighbor classifier. The recognition rates we achieve with this procedure are shown in Table 3. Evaluating some misclassified samples, we note that the recognition errors mainly occur on pairs of fingerprints from different classes that have a high subjective similarity.

6 Conclusions

In the present paper we propose an efficient approximate edit distance algorithm for planar graphs. The graph matching is performed by iteratively extending pairs of matching subgraphs of two given graphs. Our algorithm generates a single edit path between two graphs by locally optimizing the structure cor-

respondence. The optimization is accomplished with an efficient cyclic string matching algorithm.

We evaluate the planar edit distance on fingerprint graphs extracted from grayscale fingerprint images from the NIST-4 database. The edit distance approximation is very fast compared to a standard edit distance computation. The approximated distance values seem to be sufficiently accurate for the measurement of the structural similarity of graphs. Particularly for larger graphs with more than 100 nodes and edges, the planar edit distance offers a good tradeoff between running time and accuracy. In the future we intend to study the influence of the set of prototypical structures on the classification performance and evaluate the fingerprint classification system on larger data sets.

Acknowledgment

This research was supported by the Swiss National Science Foundation NCCR program "Interactive Multimodal Information Management (IM)2" in the Individual Project "Multimedia Information Access and Content Protection".

References

1. Bunke, H., Shearer, K.: A graph distance metric based on the maximal common subgraph. Pattern Recognition Letters **19** (1998) 255–259
2. Fernandez, M.L., Valiente, G.: A graph distance metric combining maximum common subgraph and minimum common supergraph. Pattern Recognition Letters **22** (2001) 753–758
3. Wallis, W., Shoubridge, P., Kraetzl, M., Ray, D.: Graph distances using graph union. Pattern Recognition Letters **22** (2001) 701–704
4. Sanfeliu, A., Fu, K.: A distance measure between attributed relational graphs for pattern recognition. IEEE Transactions on Systems, Man, and Cybernetics **13** (1983) 353–363
5. Messmer, B., Bunke, H.: A new algorithm for error-tolerant subgraph isomorphism detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 493–504
6. Hopcroft, J., Wong, J.: Linear time algorithm for isomorphism of planar graphs. In: Proc. 6th Annual ACM Symposium on Theory of Computing. (1974) 172–184
7. Luks, E.: Isomorphism of graphs of bounded valence can be tested in ploynomial time. Journal of Computer and Systems Sciences **25** (1982) 42–65
8. Dickinson, P., Bunke, H., Dadej, A., Kraetzl, M.: On graphs with unique node labels. In Hancock, E., Vento, M., eds.: Proc. 4th Int. Workshop on Graph Based Representations in Pattern Recognition. LNCS 2726 (2003) 13–23
9. Lumini, A., Maio, D., Maltoni, D.: Inexact graph matching for fingerprint classification. Machine Graphics and Vision, Special Issue on Graph Transformations in Pattern Generation and CAD **8** (1999) 231–248
10. Ambauen, R., Fischer, S., Bunke, H.: Graph edit distance with node splitting and merging and its application to diatom identification. In Hancock, E., Vento, M., eds.: Proc. 4th Int. Workshop on Graph Based Representations in Pattern Recognition. LNCS 2726 (2003) 95–106

11. Bunke, H., Bühler, U.: Applications of approximate string matching to 2D shape recognition. Pattern Recognition **26** (1993) 1797–1812
12. Lladós, J., Martí, E., Villanueva, J.: Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence **23** (2001) 1137–1143
13. Peris, G., Marzal, A.: Fast cyclic edit distance computation with weighted edit costs in classification. In Kasturi, R., Laurendeau, D., Suen, C., eds.: Proc. 16th Int. Conf. on Pattern Recognition. Volume 4. (2002) 184–187
14. Mollineda, R., Vidal, E., Casacuberta, F.: A windowed weighted approach for approximate cyclic string matching. In Kasturi, R., Laurendeau, D., Suen, C., eds.: Proc. 16th Int. Conf. on Pattern Recognition. (2002) 188–191
15. Robles-Kelly, A., Hancock, E.: String edit distance, random walks and graph matching. Int. Journal of Pattern Recognition and Artificial Intelligence (2004) to appear.
16. Kawagoe, M., Tojo, A.: Fingerprint pattern classification. Pattern Recognition **17** (1984) 295–303
17. Karu, K., Jain, A.: Fingerprint classification. Pattern Recognition **29** (1996) 389–404
18. Rao, K., Balck, K.: Type classification of fingerprints: A syntactic approach. IEEE Transactions on Pattern Analysis and Machine Intelligence **2** (1980) 223–231
19. Jain, A., Prabhakar, S., Hong, L.: A multichannel approach to fingerprint classification. IEEE Transactions on Pattern Analysis and Machine Intelligence **21** (1999) 348–359
20. Wilson, C., Candela, G., Watson, C.: Neural network fingerprint classification. Journal of Artificial Neural Networks **1** (1994) 203–228
21. Marcialis, G., Roli, F., Serrau, A.: Fusion of statistical and structural fingerprint classifiers. In Kittler, J., Nixon, M., eds.: 4th Int. Conf. Audio- and Video-Based Biometric Person Authentication. LNCS 2688 (2003) 310–317
22. Watson, C., Wilson, C.: NIST Special Database 4. Fingerprint Database. (1992)