Efficient Techniques for a Very Accurate Measurement of Dissimilarities between Cyclic Patterns

R.A. Mollineda, E. Vidal, and F. Casacuberta*

Instituto Tecnológico de Informática, Universidad Politécnica de Valencia
Camino de Vera s/n, 46071 Valencia, Spain
{rmollin,evidal,fcn}@iti.upv.es

Abstract. Two efficient approximate techniques for measuring dissimilarities between cyclic patterns are presented. They are inspired on the quadratic time algorithm proposed by Bunke and Bühler. The first technique completes *pseudoalignments* built by the Bunke and Bühler algorithm (BBA), obtaining full alignments between cyclic patterns. The edit cost of the minimum-cost alignment is given as an upper-bound estimation of the exact cyclic edit distance, which results in a more accurate bound than the lower one obtained by BBA. The second technique uses both bounds to compute a weighted average, achieving even more accurate solutions. Weights come from minimizing the sum of squared relative errors with respect to exact distance values on a training set of string pairs. Experiments were conducted on both artificial and real data, to demonstrate the capabilities of new techniques in both accurateness and quadratic computing time.

Keywords: Cyclic patterns, cyclic strings, approximate string matching, structural pattern analysis, 2D shape recognition.

1 Introduction

The problem of evaluating a measure of similarity between symbol strings arises in numerous applications [1,2], and has become a fundamental issue of structural pattern analysis. A number of similarities or distance measures have been proposed, many of them being special cases or generalizations of the Levenshtein metric [3]. A commonly used generalization of this metric is the minimum cost of transforming a string x into a string y, when the allowable *edit operations* are symbol insertion, deletion and substitution, with costs that are functions of the involved symbol(s). From the analogue with the DNA sequences, this transformation procedure is also known as *alignment*. The *optimal alignment* between x and y, defined as the minimum-cost edit sequence to transform x into y, is

* This work has been supported by a grant founded by the Agencia Española de Cooperación Internacional and the European ESPRIT project 30268 EUTRANS.

F.J. Ferri et al. (Eds.): SSPR&SPR 2000, LNCS 1876, pp. 337–346, 2000.

efficiently computed in quadratic time by a dynamic programming technique proposed, among others, by Wagner and Fisher [4].

There are specific applications in which, some string x is conveniently considered as the representative of a class of strings composed by all circular shifts of x. These strings are named *cyclic strings*. An optimal alignment between two cyclic strings x and y is defined as one having the minimum cost of transforming some fixed shift of x into any circular rotation of y [5]. This brute-force approach takes $\mathcal{O}(|x| \cdot |y|^2)$ time, where $|\cdot|$ represents the string length. Maes [5] proposed an $\mathcal{O}(|x| \cdot |y| \cdot \log |y|)$ algorithm that efficiently uses dynamic programming properties. Another approach of cyclic string matching is presented in [6]. It has a theoretical cubic time complexity but, its practical execution time may significantly be lower. On the other hand, by sacrificing the strict optimality of the solution, another way to efficiently deal with this problem arises. Bunke and Bühler [1] introduced an $\mathcal{O}(|x| \cdot |y|)$ suboptimal method which was successfully used for 2D shape recognition.

In this paper two new approximate techniques are developed based on the Bunke and Bühler algorithm (BBA). They are compared with BBA and Maes algorithm, attending to both solution optimality and algorithm execution time. Results show that the accuracy of the estimations of the new methods are much better than those provided by BBA, closely approaching the exact distance values given by the computationally more intensive Maes algorithm.

2 Foundations

Let Σ be an alphabet and let Σ^* be the set of all finite-length strings over Σ. Let ϵ denote the empty symbol. An edit operation is an ordered pair $(x, y) \neq (\epsilon, \epsilon)$ of strings of lengths less than 2, denoted by $x \to y$. For all x_i, and y_j in Σ, the function γ assigns nonnegative real-valued costs to the following edit operations: *substitute* operation, $\gamma(x_i \to y_j) \geq 0$ (it is also known as *match* if $x_i = y_j$); *delete* operation, $\gamma(x_i \to \epsilon) > 0$; and *insert* operation, $\gamma(\epsilon \to y_j) > 0$.

The function γ can be extended to a sequence of edit operations $E = e_1 e_2 \dots e_k$, by defining cost of a sequence as $\gamma(E) = \sum_{i=1}^{k} \gamma(e_i)$. The *edit distance* δ between strings x and y is then expressed as

$$\delta(x, y) = \min\{\gamma(S) \mid S \text{ is an edit sequence transforming } x \text{ into } y\} \ . \quad (1)$$

This computation is performed by working on a dynamic programming matrix called *edit graph* [4] defined by $(|x| + 1)$ rows and $(|y| + 1)$ columns, with time and space complexity in $\mathcal{O}(|x| \cdot |y|)$.

3 Cyclic Alignments

A *cyclic shift* $\sigma : \Sigma^* \to \Sigma^*$ is defined by $\sigma(x_1 x_2 \dots x_{|x|}) = x_2 x_3 \dots x_{|x|} x_1$. Let σ^k be the composition of k cyclic shifts. A *cyclic string* x is an equivalent class, denoted by $[x]$, defined by an equivalent relation on Σ^*: $x' \equiv x \Leftrightarrow x' = \sigma^k(x)$, for

some $k \in \mathbb{N}$. Given two cyclic strings $[x]$ and $[y]$, the cyclic distance δ_{C} between them is defined using (1) as $\delta_{\mathrm{C}}([x],[y]) = \min\{\delta[\sigma^k(x), \sigma^l(y)]|\ k,l \in \mathbb{N}\}$. The following lemma [5], shows a more efficient way to compute $\delta_{\mathrm{C}}([x],[y])$.

Lemma 1. *For each representative x of a cyclic string $[x]$ and for each cyclic string $[y]$, $\delta_C([x],[y]) = \delta_C(x,[y])$, where $\delta_C(x,[y]) = \min\{\delta[x,\sigma^l(y)]|\ l \in \mathbb{N}\}$.*

The above lemma states that the brute-force approach to compute the cyclic edit distance between $[x]$ and $[y]$ has $\mathcal{O}(|x| \cdot |y|^2)$ time complexity. The problem we deal with in this paper is, given two cyclic strings $[x]$ and $[y]$, to approximate $\delta_{\mathrm{C}}(x,[y])$ as well as possible in a quadratic computing cost.

4 The Approximate Algorithm of Bunke and Bühler: A Non Length-Preserving Approach

Let $y^2 = y_1 y_2 \ldots y_{|y|} y_1 y_2 \ldots y_{|y|}$ be the concatenation of the string y with itself. The Bunke and Bühler Algorithm (BBA) [1] is based on the following lemma, which comes from the fact that the set of substrings of length $|y|$ in y^2 is equal to the equivalent class $[y]$.

Lemma 2. *Let x and y be two strings. The distance $\delta_C(x,[y])$ can be computed as $\delta_{|y|}(x,y^2)$, where $\delta_{|y|}(x,y^2)$ is the edit distance between x and its most similar substring in y^2 of length $|y|$.*

BBA produces an estimation of $\delta_{|y|}(x,y^2)$ by searching in the edit graph E defined by x and y^2 the minimum weighted path from any of the *starting* nodes $(0,0),(0,1),\ldots,(0,|y|)$ to any of the *final* nodes $(|x|,|y|+1),(|x|,|y|+2),\ldots,(|x|,2\cdot|y|)$ without any control over the length of the paths. This leads to the following computation:

$$
\begin{aligned}
E(0,j) &= \begin{cases} 0 & \forall j,\ 0 \le j \le |y| \\ E(0,j-1) + \gamma(\epsilon \to y_{j-|y|}) & \forall j,\ |y|+1 \le j \le 2\cdot|y| \end{cases} \\
E(i,0) &= E(i-1,0) + \gamma(x_i \to \epsilon) \qquad \forall i,\ 1 \le i \le |x| \\
E(i,j) &= \min \begin{cases} E(i-1,j) + \gamma(x_i \to \epsilon) \\ E(i-1,j-1) + \gamma(x_i \to y_j) \\ E(i,j-1) + \gamma(\epsilon \to y_j) \end{cases} \begin{array}{l} \forall i,\ 1 \le i \le |x| \\ \forall j,\ 1 \le j \le 2\cdot|y| \end{array} .
\end{aligned}
\tag{2}
$$

The BBA estimation $\delta_{\mathrm{B}}(x,y)$ of $\delta_{\mathrm{C}}(x,[y])$ is the smallest cost value among the set of values computed at final nodes. BBA has $\mathcal{O}(|x| \cdot |y|)$ time complexity. In the rest of the paper, $\delta_{\mathrm{C}}(x,[y])$ will be denoted as $\delta_{\mathrm{C}}(x,y)$.

Figure 1 illustrates the two possible edit graphs to approximately compute, by BBA, the cyclic edit distance between the strings *ba* and *abab*. Both graphs show the suboptimality of this algorithm, giving the approximate values 0 and 1 respectively, which are lower than the exact distance value 2.

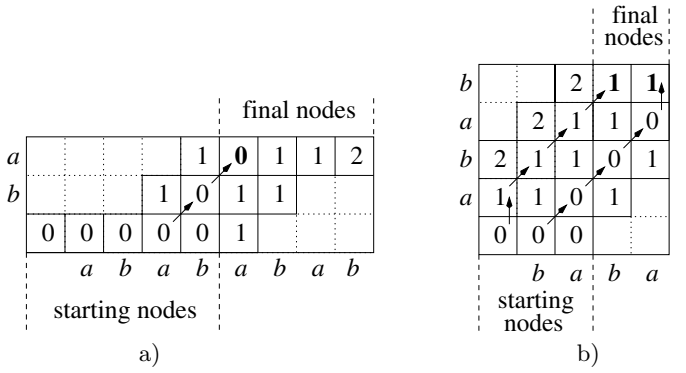

Fig. 1. Two possible edit graphs to approximately compute, by BBA, the cyclic distance between $x = ba$ and $y = abab$: a) graph associated to x and y^2 and b) graph associated to y and x^2. Minimum-cost alignments are marked by sequences of arrows. The estimation given by $\delta_{\mathrm{B}}(y, x)$ (1) in the graph b), where the longest string was vertically placed, is more accurate than the one obtained by $\delta_{\mathrm{B}}(x, y)$ (0). The exact cyclic distance value is 2 due to $x = ba$ matches with a prefix of the rotated version $baba$ from the original $y = abab$, and two further symbols need to be inserted.

It was shown in references [7] and [1] that, given two strings x and y, BBA computes $\delta_{\mathrm{B}}(x, y) = \delta(x, y')$, where y' is a substring of y^2, which is most similar to x, i.e. $\delta(x, y') = \min\{\delta(x, z) |\ z \text{ is a substring of } y^2\}$. The corresponding editing path will be called *pseudoalignment*. The following lemma is straightforward from the above discussion.

Lemma 3. *The BBA estimation is a lower bound of the exact cyclic distance:* $\delta_B(x, y) \leq \delta_C(x, y)$

Significant differences between estimations given by $\delta_{\mathrm{B}}(x, y)$ and $\delta_{\mathrm{B}}(y, x)$ emerge. Let us assume that $|x| < |y|$. When $\delta_{\mathrm{B}}(x, y)$ is computed, y' is the substring of y^2 most similar to x in contents and length. Therefore, the length of y' may tend to be closer to $|x|$ and farther from $|y|$. In this way, the quality of the estimation may inversely depend on the difference $|y| - |x|$. On the other hand, in $\delta_{\mathrm{B}}(y, x)$ computation, there is not a clear trend on values taken by $|x'|$ (x' is the substring of x^2 most similar to y), except a closer approximation to both $|y|$ and $|x|$. Consequently, the estimation given by $\delta_{\mathrm{B}}(y, x)$ is expected to be more accurate than the one obtained by $\delta_{\mathrm{B}}(x, y)$(see Fig. 1). Exhaustive experiments on both synthetic and real data, clearly confirmed this expectation.

5 Extending BBA Pseudoalignments to Full Alignments

In this section, an extension of pseudoalignments built by BBA is proposed to create complete alignments. First, a simple mechanism is proposed to know, along with the $\delta_{\mathrm{B}}(x, y)$ computation, the length of the longest substring $\overline{y_i}$ of

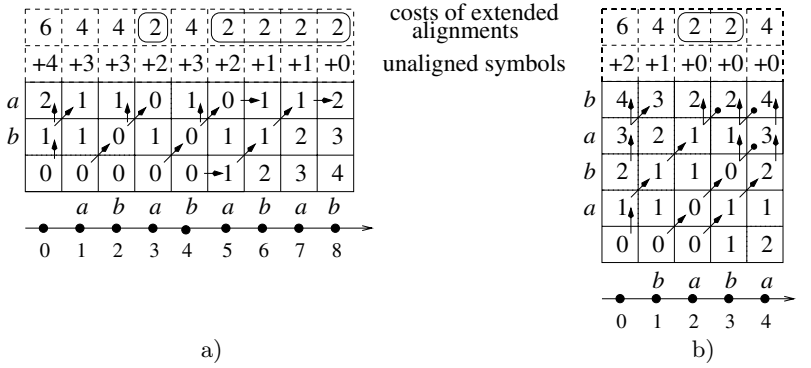

Fig. 2. Two possible edit graphs to approximately compute, by EBBA, the cyclic distance between $x = ba$ and $y = abab$: a) graph associated to x and y^2 and b) graph associated to y and x^2. Minimum-cost alignments to each final node are marked by sequences of arrows. The lower dashed rows contain the number of symbols not aligned with the vertical string by each minimum-cost alignment, and the upper dashed rows list the costs of the EBBA alignments. In the graph b), round-cap arrows represent situations in which alignments are pruned to avoid paths longer than $|x|$. Both edition graphs lead to the exact cyclic distance value 2.

y^2 that reach the final node $(|x|, i)$, $0 \leq i \leq 2 \cdot |y|$, among all possible substrings associated with minimum edition paths to this particular node. If $|\overline{y_i}|$ is known, the subset of $|y| - |\overline{y_i}|$ symbols not aligned with x, which are final components of certain $\overline{y} \in [y]$ for which $\overline{y_i}$ is a prefix, can be easily identified. Then, by inserting such unaligned symbols to the end of the partial edit sequence, a complete alignment between x and $\overline{y}$ is constructed.

In the example of Fig. 1a), the string $x = ba$ was aligned with the substring $\overline{y_5} = ba$ via the partial edit sequence $E' = \{b \to b, a \to a\}$. Since $|\overline{y_5}| = 2$ and $|y| = 4$, it can be concluded that two symbols from y (symbols b and a) have not been aligned with x. Then, by inserting these unaligned symbol to E', a complete alignment is built, leading to $E' = \{b \to b, a \to a, \epsilon \to b, \epsilon \to a\}$.

The estimation $\delta_{\mathrm{EB}}(x, y)$ given by this *extended* version of BBA (EBBA) takes the same $\mathcal{O}(|x| \cdot |y|)$ time of BBA. Figure 2 shows actions performed by EBBA on the two edit graphs presented in Fig. 1. In both situations, the approximation given by EBBA is the exact cyclic distance between the strings involved.

The length of each substring of y^2 aligned with x can be computed when the edit graph corresponding to $\delta_{\mathrm{EB}}(x, y)$ is being built using (2). In fact, only the starting node is actually needed, and the required computation can be easily performed along with the standard minimization. This mechanism is also used to keep alignments between x and substrings of y^2 of sizes lower than $|y|$, by checking the length of each partial winner path prior to extend it (see Fig. 2b).

Lemma 4. *The EBBA estimation is an upper bound of the exact cyclic distance:* $\delta_{EB}(x, y) \geq \delta_C(x, y)$

Proof. EBBA builds full alignments between x and some members of $[y]$, and $\delta_{\mathrm{C}}(x,y)$ is the cost of the minimum-cost alignment between x and any $\overline{y} \in [y]$.

6 Using BBA and EBBA as Bounds to Construct More Approximate Solutions

Lemmas 3 and 4 give lower and upper bounds of the exact solution to the problem of computing the distance between cyclic strings, i.e., $\delta_{\mathrm{B}}(x,y) \leq \delta_{\mathrm{C}}(x,y) \leq \delta_{\mathrm{EB}}(x,y)$. In this section a new approximate solution $\delta_{\mathrm{W}}(x,y)$ (Weighted BBA) is proposed, as a weighted average between the lower bound $\delta_{\mathrm{B}}(x,y)$ and the upper bound $\delta_{\mathrm{EB}}(x,y)$. Coefficients (weights) can be estimated by minimizing the sum of squared relative errors of the weighted solutions with respect to the exact distances. They are computed by using a training set T of string pairs of the problem at hand. Since both bounds can be computed simultaneously in $\mathcal{O}(|x| \cdot |y|)$ time, the combined solution is also computed in $\mathcal{O}(|x| \cdot |y|)$ time. A formalization is given below:

$$\delta_{\mathrm{W}}(x,y) = \alpha \cdot \delta_{\mathrm{B}}(x,y) + (1-\alpha) \cdot \delta_{\mathrm{EB}}(x,y) \ . \tag{3}$$

The approximation error is:

$$E_{\mathrm{W}} = \sum_{\forall (x,y) \in T} \left[\frac{\delta_{\mathrm{C}}(x,y) - \delta_{\mathrm{W}}(x,y)}{\delta_{\mathrm{C}}(x,y)} \right]^2 \ . \tag{4}$$

The error is minimized for a value of α such that $\frac{dE_W}{d\alpha} = 0$:

$$\alpha = \frac{\sum_{\forall (x,y) \in T} [\delta_{EB}(x,y) - \delta_B(x,y)] \cdot \left[\frac{\delta_{EB}(x,y)}{\delta_C(x,y)} - 1 \right]}{\sum_{(x,y) \in T} \frac{[\delta_{EB}(x,y) - \delta_B(x,y)]^2}{\delta_C(x,y)}} \ . \tag{5}$$

7 Experimental Results

A number of experiments were conducted on simulated as well as on real data. The first series (Sect. 7.1) compares the quality of the solutions yield by the Bunke and Bühler algorithm (BBA), the *extended* BBA (EBBA) and the *weighted* version of BBA (WeBBA), measured as the Average Relative Error (ARE) with respect to exact distance values. It is defined as $ARE = \sqrt{E_*/P}$, where $E_* \in \{E_{\mathrm{B}}, E_{\mathrm{EB}}, E_{\mathrm{W}}\}$ is computed as in (4) for the three suboptimal techniques, and P is the number of pairs of the test set. A comparison among their computing times and the time needed by the Maes algorithm (MA) is also included. In Sect. 7.2 a real classification problem is considered, where distance estimation techniques were used as dissimilarity measures between cyclic patterns. In all experiments, only the way of sorting the input string pairs in which the longest string is vertically placed in the edit graph is considered (see Sect. 4). Exhaustive

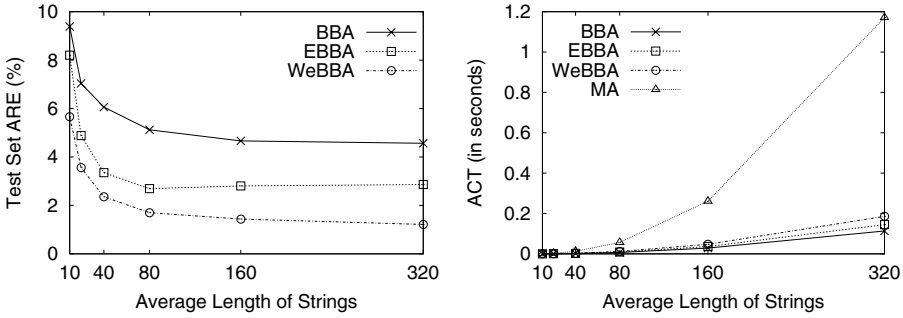

Fig. 3. Comparison among BBA, EBBA and WeBBA for each set of synthetic strings regarding: a) the Average Relative Error (ARE) on test sets, given as percentages; b) the Average Computing Time (ACT) in seconds needed by suboptimal techniques and by Maes algorithm, to approximate and to exactly compute a cyclic distance, respectively.

experiments not presented in this paper show that, using this way of considering input strings, BBA achieves a significantly better estimation to the exact solution and consequently, it can give us a more precise criterion of how better the proposal techniques (EBBA and WeBBA) are with respect to BBA. The value assigned to each positive-cost edit operation is 1 for all experiments. All the algorithms were implemented in C programming language, and experiments were performed on a 166 MHz Intel Pentium MMX with 128 Mb of RAM.

7.1 A Comparison among BBA, EBBA and WeBBA

Experiments on Synthetic Strings. Six pairs of training and test sets with 70 randomly drawn strings in each one, were generated from a uniform distribution law. The symbols of the strings were chosen from an alphabet composed by six symbols. To guarantee different average lengths of strings among pairs of sets, individual lengths were also randomly drawn following a uniform distribution law in the ranges $[5, 15]$, $[15, 25]$, $[30, 50]$, $[65, 95]$, $[135, 185]$ and $[280, 360]$ respectively. From the 70 strings of each set, a set of 2415 different non-ordered string pairs was built. For each of the six training sets, a corresponding optimal value of α was computed using (5).

Results are presented in Fig. 3 in terms of the approximation error (ARE) on test sets and the Average Computing Time (ACT) of estimating a cyclic distance as a function of the average string length.

A very important error reduction is achieved by WeBBA with respect to previous approaches, mainly for longer strings. On the other hand, computing times of all suboptimal methods are similar and are dramatically smaller than that of the exact Maes algorithm.

The optimal values of α computed from each training set were 0.431, 0.316, 0.316, 0.276, 0.339 and 0.368 respectively. All these values were lower than 0.5,

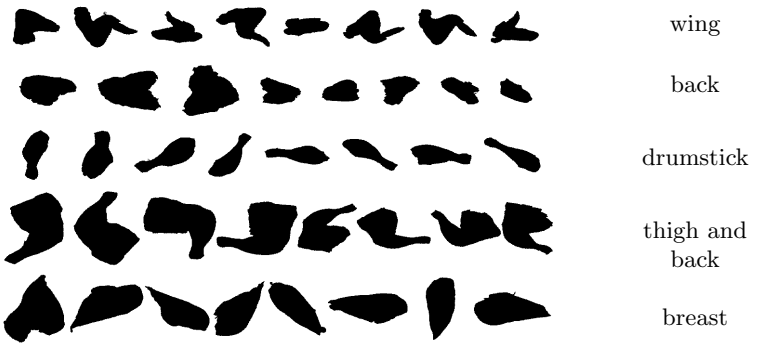

Fig. 4. Chicken piece images from different classes.

assessing the fact that EBBA makes notably better estimations than BBA. In this way, the value of α represents for each training set, a reliable index of the relationship between the accuracies of the BBA and EBBA. To better approximations of EBBA with respect to BBA corresponds lower values of α. On the other hand, the estimations given by these suboptimal techniques are data-dependent with respect to, for example, the lengths of the involved strings, or with respect to the size of the alphabet of symbols. In this particular example with synthetic data, different values of α were obtained from each training set. This is due to the fact that the average lengths of the strings members of the sets are notably different, which leads to different relations between the accuracies of the estimations given by BBA and EBBA, respectively.

Experiments with Chain-Code Representations from Silhouettes of Chicken Pieces. The previous experiment was repeated with a set of chain-code contours describing silhouettes of chicken parts. A set composed by 446 images from chicken pieces was used [8]. Each piece belongs to one of five categories, which represent specific parts of the chicken: wing (117 samples), back (76), drumstick (96), thigh and back (61), and breast (96). Each image is in binary format containing a silhouette from a particular piece. Pieces were placed in a natural way without considering orientation. All images were adequately clipped and scaled into 64x64 pixels images (see examples in Fig. 4). A standard 4-direction chain-encoding procedure [9] was applied, and the resulting chain-code contours were re-encoded into rotation-invariant representations where new codes specify relative change of angles as a function of line length [10].

From these 446 chain-code strings, two sets of 100 (one for training and one for test) were randomly picked, preserving the frequencies among different classes. From each string set, an associated set composed by 4950 different non-ordered string pairs was built. The training set of pairs was used to compute the optimal value for α using (5). Table 1 shows the computed value of α and the approximation error (ARE) of BBA, EBBA and WeBBA on both the test set and the training set. In this case, WeBBA obtained an ARE reduction of 40.43% with respect to EBBA on test set.

Table 1. Approximation error (ARE) of BBA, EBBA and WeBBA for rotation-invariant chain-code representations of silhouettes of chicken pieces.

α	ARE of WeBBA on Training Set	ARE of WeBBA on Test Set	ARE of EBBA on Test Set	ARE of BBA on Test Set
0.328	1.73%	2.24%	3.76%	7.88%

7.2 Classification Experiments with the Chicken Pieces Data Set

In this section a classification experiment on the data set composed by chain-code representations from chicken pieces is presented. The Levenshtein non-cyclic edit distance (ED), the BBA, the WeBBA and the Maes procedure were used to compute dissimilarities between cyclic patterns. These dissimilarities were normalized by the sum of the lengths of the two cyclic patterns involved. Classification error rate was estimated for all the techniques, through a "leaving one out" scheme with the 1-Nearest Neighbor classification rule using the 446 samples. Since WeBBA needs a training set for estimating the value of α, the data set was partitioned into four subsets keeping the frequencies among classes for this particular case. From each subset i, $0 \leq i \leq 3$, a related set of all different non-ordered pairs of strings was built. It was used to compute the value of the corresponding α_i using (5). To classify a contour of the subset i the coefficient $\alpha_{(i+1) \bmod 4}$ was used. In this way, each cyclic pattern did not take part in the computation of the coefficient α used in its classification.

Table 2 shows the classification error rate and the average time needed to make a decision for each technique. The WeBBA is as accurate as the Maes optimal algorithm with respect to the error rate, and at the same time, it has been almost as efficient as BBA and ED, when computation time was considered.

8 Conclusions and Further Work

In this paper two efficient techniques based on the Bunke and Bühler algorithm (BBA)[1] have been proposed to approach the cyclic edit distance between two strings. They have been compared attending to the quality of their estimations and the computing time with respect to BBA and the Maes algorithm (MA).

Table 2. Classification results on chicken pieces data set.

String-to-string Technique	Error Rate	Average Time (in seconds) to Make a Decision
ED	32.51%	2.90
BBA	24.66%	6.01
WeBBA	21.97%	9.69
Maes	22.65%	56.49

The first technique (EBBA) extends pseudoalignments built by BBA to complete alignments by inserting all the symbols not considered by BBA. The second one (WeBBA) consists of a weighted average between the estimations given by BBA (lower bound of the exact value) and EBBA (upper bound), to achieve a significant better approximation. Weights are those which minimize the sum of the squared relative errors of the weighted solutions for a training set.

Experiments on both synthetic and real data, shown that better approximations are achieved by WeBBA and EBBA with respect to BBA, keeping its same quadratic computation time, and far from the time cost of MA. Synthetic experiments showed that estimation errors given by suboptimal techniques tend to decrease as lengths of strings grow, specially for WeBBA. A classification experiment was also carried out, in which WeBBA had a very good behavior on both accurateness and computing time, with respect to other string-to-string techniques. This approximate technique (WeBBA) seems to be a very attractive approach to deal with classification tasks based on cyclic string matching, because it constitutes a good trade-off between accuracy and computation time.

Another worth mentioning point concerns the possibility of computing values for the coefficient α, which optimize criteria different from accurateness. For example, if the problem has well-defined classes, different α values could be learned for each class according to some criterion related with classification error. To classify a new sample, distances to each class are computed by considering the values of the parameters for this specific class.

References

[1] H. Bunke and U. Bühler. Applications of approximate string matching to 2D shape recognition. *Pattern Recognition*, 26(12):1797–1812, 1993.

[2] H. Bunke and A. Sanfeliu, editors. *Syntactic and Structural Pattern Recognition Theory and Applications*, Singapore, 1990. World Scientific.

[3] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966.

[4] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *J. Assoc. Comput. Machinery*, 21:168–173, 1974.

[5] M. Maes. On a cyclic string-to-string correction problem. *Information Processing Letters*, 35(2):73–78, June 1990.

[6] J. Gregor and M. G. Thomason. Dynamic programming alignment of sequences representing cyclic patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(2):129–135, February 1993.

[7] P. H. Sellers. The theory and computation of evolutionary distances: Pattern recognition. *Journal of Algorithms*, 1:359–373, 1980.

[8] G. Andreu, A. Crespo, and J. M. Valiente. Selecting the toroidal self-organizing feature maps (TSOFM) best organized to object recognition. In *Proceedings of ICNN 97*, volume 2, pages 1341–1346, Houston, Texas (USA), June 1997. IEEE.

[9] H. Freeman. Computer processing of line drawing images. *Computer Surveys*, 6:57–98, 1974.

[10] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, 1992.