International Journal of Pattern Recognition and Artificial Intelligence
Vol. 16, No. 3 (2002) 291–299

CYCLIC SEQUENCE ALIGNMENTS: APPROXIMATE VERSUS OPTIMAL TECHNIQUES

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@iti.upv.es*
†*evidal@iti.upv.es*
‡*fcn@iti.upv.es*

The problem of cyclic sequence alignment is considered. Most existing optimal methods for comparing cyclic sequences are very time consuming. For applications where these alignments are intensively used, optimal methods are seldom a feasible choice. The alternative to an exact and costly solution is to use a close-to-optimal but cheaper approach. In previous works, we have presented three suboptimal techniques inspired on the quadratic-time suboptimal algorithm proposed by Bunke and Bühler. Do these approximate approaches come sufficiently close to the optimal solution, with a considerable reduction in computing time? Is it thus worthwhile investigating these approximate methods? This paper shows that approximate techniques are good alternatives to optimal methods.

Keywords: Cyclic sequences; cyclic string matching; structural pattern analysis.

1. Introduction

It is often necessary to compare two or more sequences, and measure the extent to which they differ. A basic approach is to analyze the total difference between two sequences, regarding a collection of individual elementary differences, usually, substitutions, deletions and insertions. They are known as *edit operations*, because they *operate* over a source sequence by actively changing (*editing*) it into a target sequence. Levenshtein[5] introduced a *distance* from one sequence x to another sequence y, $\delta(x, y)$, defined as the smallest number of edit operations to change x into y. One widely adopted generalization to the Levenshtein metric is to weigh the edit operations. It is defined as a minimum-weighted sequence of transforming x into y. The optimization procedure for computing (weighted) edit distances falls within the framework of dynamic programming. The general algorithm, that was reported in Ref. 10, among many others, requires a computing time that is quadratic in the length of the sequences.

*Author for correspondence.

An interesting extension of the previous method considers a string x as a *cyclic string*. A *cyclic string* x is an equivalence class, denoted by $[x]$, defined by the following equivalent relation: $x' \equiv x \Leftrightarrow x' = \sigma^k(x)$, for some $k \in \mathbb{N}$, where $\sigma^k(x) = x_{k+1}x_{k+2}\cdots x_{|x|}x_1\cdots x_k$. Given two cyclic strings $[x]$ and $[y]$, the *cyclic distance* δ_C between them is defined as

$$\delta_C([x],[y]) = \min\{\delta[\sigma^k(x),\sigma^l(y)]|k,l \in \mathbb{N}\}\,. \tag{1}$$

In the same way, the edit distance between a linear string x and a cyclic string $[y]$ can also be defined by

$$\delta_C(x,[y]) = \min\{\delta[x,\sigma^l(y)]|l \in \mathbb{N}\}\,. \tag{2}$$

The following lemma[6] presents a more efficient way to compute the edit distance between cyclic strings by requiring a cubic time computation complexity.

Lemma 1. *For each representative x' of a cyclic string $[x]$ and for each cyclic string $[y]$ we have $\delta_C([x],[y]) = \delta_C(x',[y])$.*

A Divide and Conquer approach was presented in Ref. 6 (MA) to efficiently compute the optimal cyclic alignment between two strings x and y, where $|x|$ represents the length of x, reducing the computational complexity from cubic to $O(|x||y|\log|y|)$ in the worst case. Another algorithm for cyclic string matching was introduced in Ref. 4 (GT). It has a theoretical cubic time complexity but, its practical execution time may be significantly lower. The MA carries out an exhaustive scanning of a Divide and Conquer binary partition of the solution space. A Branch and Bound strategy based on the Maes approach was proposed in Ref. 7, to explore only those promising branches that may lead to the optimal solution. Two lower bound functions (MB1 and MB2) are introduced for the best possible solution which can be reached from a current state. Both versions keep the worst case complexity of the Maes approach, but they always compute the optimal solution faster than MA and GT.

On the other hand, by sacrificing the strict optimality of the solution, another way to efficiently deal with this problem arises. An $O(|x||y|)$ *suboptimal* method was proposed in Ref. 2 (BBA). In Refs. 8 and 9 we proposed three new approximate approaches inspired on the BBA. These previous works showed that our suboptimal methods are much more accurate than BBA, closely approaching the optimal solution. But, how much faster are the approximate solutions obtained with respect to the computation of the exact solution? Is it worthwhile to use an estimation instead of the exact distance value? This article attempts to show that approximate techniques are good alternatives to optimal methods. The experiments were limited to the cyclic use of the Levenshtein metric, i.e. unit-weight insertions, deletions and substitutions.

2. The Approximate Computation of Cyclic Distance

The Bunke and Bühler Algorithm (BBA)[2] produces a lower-bound estimation of $\delta_C(x,[y])$.[8] This computation is performed by working on a dynamic programming

matrix called *edit graph*,[10] with time and space complexities in $O(|x||y|)$. It is defined by a quadratic set of nodes composed of $(|x|+1)$ rows and $(2|y|+1)$ columns and a set of arcs, where horizontal arcs correspond to insertions, diagonal arcs to substitution, and vertical ones to deletions. The BBA builds *partial* edit sequences between x and substrings of y^2 (the concatenation of y with itself) of uncontrolled lengths, and takes the minimum-weighted sequence as its approximate value $\delta_B(x, y)$. The quality of BBA estimation strongly depends on the strings order, being notably worse when the shorter string is the first. This behavior is proportionally strengthened with the increase in the difference between string lengths.

A first *extension* to the original BBA (EBBA) was reported in Ref. 8. The idea is to keep track, along with the $\delta_B(x, y)$ computation, of the length of the longest substring $\overline{y_i}$ of y^2 aligned with x, that reach each final node $(|x|, i)$, $0 \leq i \leq 2 \cdot |y|$. Then, the subset of $|y| - |\overline{y_i}|$ symbol positions not aligned with x can be easily identified and the corresponding symbols inserted at the end of the partial edit sequence. See Fig. 1(a). In this way, EBBA builds complete edit sequences between x and rotations of y, giving as solution the minimum-weighted one, which is denoted by $\delta_{EB}(x, y)$. This mechanism is also used to retain alignments between x and substrings of y^2 of sizes lower than $|y|$, by checking the length of each partial winner path prior to extending it. It is also computed in quadratic time, and it is an upper-bound estimation of the exact cyclic distance.[8]

A second derived technique[8] *combines* the lower and upper estimations given by BBA and EBBA, respectively, in the *weighted* solution (WeBBA), $\delta_W(x, y) = \alpha \cdot \delta_B(x, y) + (1 - \alpha) \cdot \delta_{EB}(x, y)$. The coefficient α is analytically computed such

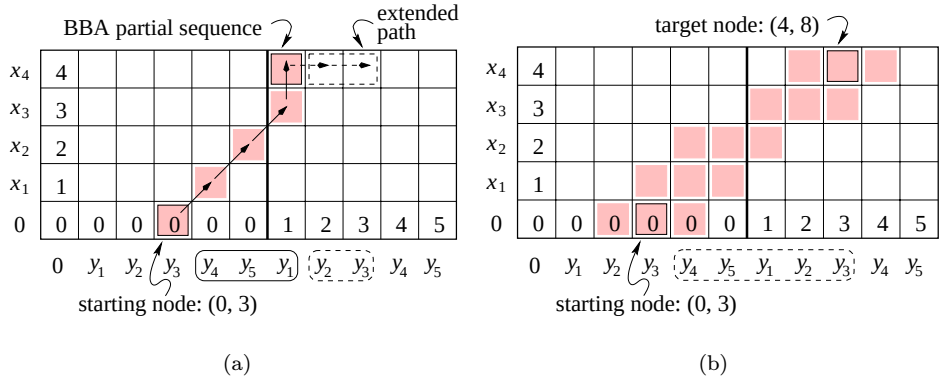

Fig. 1. **(a) EBBA procedure**: BBA transforms x into the substring of y^2, $\overline{y_6} = y_4 y_5 y_1$. Since $|y| = 5$ and $|\overline{y_6}| = 3$, there are two symbols in y not considered (y_2, y_3). By adding their insertions to the partial sequence, x is transformed into $y' = y_4 y_5 y_1 y_2 y_3 = \sigma^3(y)$, which is a real rotated version of y. **(b) WinBBA procedure**: The gray region is the diagonal window attached to $(0, 3)$, and it is also defined by the window width 1, and the slope $s = \frac{4}{5}$. The target node $(4, 8)$ is the one that must be reached by a complete alignment between x and the rotation of y, $y_4 y_5 y_1 y_2 y_3$. Thus, the window forces the alignments which start at $(0, 3)$, to finish closer than $w = 1$ node from the target.

that it minimizes the sum of squared relative errors of δ_W, over a training set of string pairs. The δ_W is also obtained in $O(|x||y|)$, because δ_B and δ_{EB} can be simultaneously computed.

A third inexact technique (WinBBA)[9] introduced a bounding diagonal *window* for each starting node $(0, i)$, $0 \leq i \leq |y|$, on the edit graph associated to BBA. It is also determined by a window width w and the slope $s = \frac{|x|}{|y|}$, where w is the number of nodes to each side of $(0, i)$. Figure 1(b) illustrates this technique. The algorithm constrains an alignment to be inside of the window associated to its starting node. Therefore, its final node will be closer than w nodes from the node $(|x|, i + |y|)$, which is the one reached by a full alignment between x and the corresponding rotation of y, $\bar{y} = y_{i+1}y_{i+2}\cdots y_{|y|}y_1\cdots y_i$ [see Fig. 1(b)]. In this way, x is being transformed into a substring of y^2 very similar to $\bar{y}$. This new estimation, denoted by $\delta_{WB}(x, y)$, takes the same $O(|x||y|)$ time complexity. A derived open problem is how to efficiently compute the optimal value for w for each pair of strings.

Examples in which these approximate algorithms produce different results can be easily found. Let $x = babba$ and $y = aba$. The cyclic distance between them is 2, and the rest of the algorithms compute the following values: $\delta_B(x, y) = 1$, $\delta_{EB}(x, y) = 2$ and $\delta_{WB}(x, y) = 2$. Figure 2 shows the operation modes of these techniques on this example through their corresponding edit graphs.

3. Synthetic Data Experiments

Six pairs of training and test sets, each with 100 randomly drawn strings, were uniformly generated from an alphabet composed by six symbols. The individual

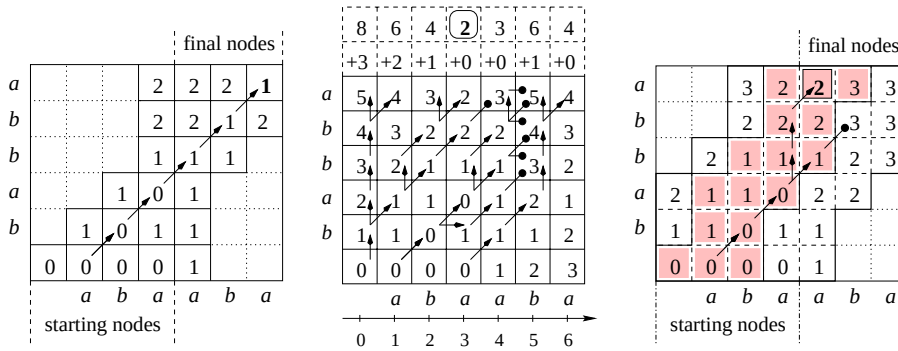

Fig. 2. Edit graphs associated to the approximate computation of the cyclic edit distance between $x = babba$ and $y = aba$, by (a) $\delta_B(x, y) = 1$, (b) $\delta_{EB}(x, y) = 2$ and (c) $\delta_{WB}(x, y) = 2$. The exact cyclic distance is 2. Minimum-cost alignments are marked by sequences of arrows. (b) The lower dashed row contains the number of symbols of y not aligned with x, and the upper one lists the final costs of EBBA extended alignments. Round-cap arrows show situations in which paths are pruned. Alignments are avoided with substrings of y^2 longer than $|y|$. (c) The winner sequence in (a) is pruned to prevent paths out the constraint window.

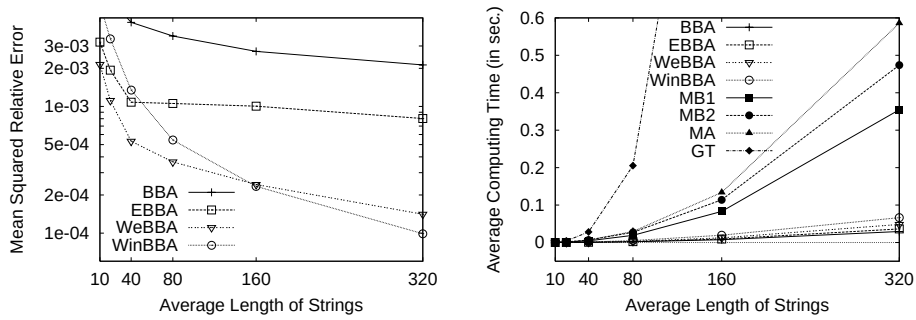

Fig. 3. Comparison among BBA, EBBA, WeBBA and WinBBA regarding: (a) the Mean Squared Relative Error on test sets of synthetic data (each one represented by the average length of its strings); (b) the Average Computing Time needed by inexact and all the exact algorithms, to approximate, and to exactly compute, a cyclic distance, respectively.

lengths of strings among pairs of sets, were also randomly drawn in the ranges $[9, 11]$, $[18, 22]$, $[36, 44]$, $[72, 88]$, $[144, 176]$ and $[288, 352]$, respectively. From the 100 strings of each set, a corresponding set of all the 4950 different nonordered string pairs was built. The parameter values required by WeBBA (α) and WinBBA (w), were estimated from the training sets of string pairs.

Results are shown in Fig. 3 in terms of the Mean Squared Relative Error (MSRE) on the test sets of string pairs, and the average time (on a Pentium III, 450 MHz) of estimating a cyclic distance, as functions of the average lengths of the strings for each set (10, 20, 40, 80, 160 and 320). Our approaches (EBBA, WeBBA, WinBBA) were all significantly more accurate than BBA. For long strings, the best one was WinBBA with a squared relative error more than 20 times lower than that of BBA. The optimal values for α were in all cases lower than 0.5 (0.26, 0.25, 0.29, 0.32, 0.35 and 0.36) assessing the fact that EBBA makes better estimations than BBA.[8] The estimated values for w were 2, 3, 3, 5, 7 and 11, respectively. The computing times of inexact methods were very similar and much smaller than that needed by the best exact algorithm (MB1).

4. Real-World Data Results

This data set is composed 446 binary images, each one containing a silhouette from a chicken piece.[1] Each piece belongs to one of five categories: wing (117 samples), back (76), drumstick (96), thigh and back (61), and breast (96). All images were adequately clipped and scaled into 64×64 pixels images (see examples in Fig. 4). A standard 4-direction chain-encoding procedure 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.[3] The average length of these sequences is about 225 codes.

Fig. 4. Chicken piece silhouettes from different classes.

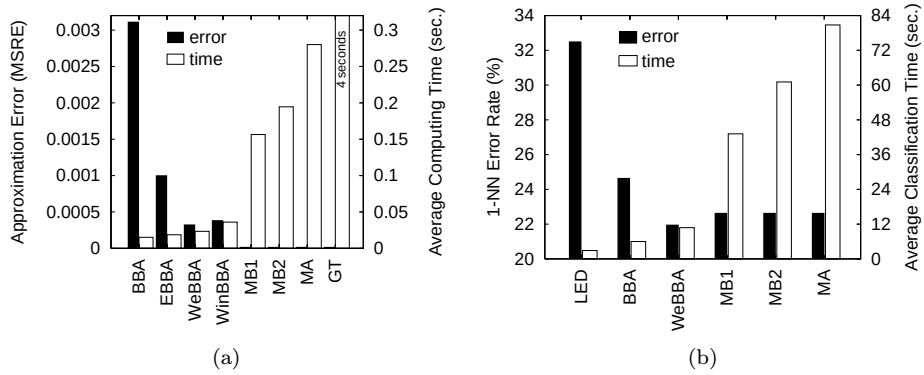

Fig. 5. (a) Approximation error (MSRE) and average computing time required by inexact and exact algorithms, to estimate and to exactly compute, a cyclic distance, on the test set of chicken pieces data set. (b) Classification results: 1-NN error rate and average time to classify a sample. LED is the Levenshtein noncyclic metric.

From these 446 chain-code strings, two (training and test) subsets of 100 samples were randomly picked. From each string subset, an associated set with all the 4950 different nonordered string pairs was built. The training set of string pairs was used to estimate the parameter values $\alpha = 0.265$ and $w = 10$, as the ones for which a minimum MSRE was achieved. The approximation errors (MSRE) computed by the inexact techniques on the test set of string pairs, and the average time needed by all the algorithms to obtain a cyclic distance, are shown in Fig. 5(a). The WeBBA reduced 90% of the BBA approximation error, and it required only 179% of the BBA time and 15% of the time needed by the faster exact algorithm (MB1).

A classification experiment on the complete data set was also carried out, using the following dissimilarities between cyclic patterns: Levenshtein (*noncyclic*) edit distance (LED), BBA, WeBBA, MB1, MB2 and MA. They were normalized by the sum of the lengths of the two cyclic patterns involved. Classification was based on the 1-NN rule and error rate was estimated through a "leaving one out" scheme using the 446 samples. Results are presented in Fig. 5(b). The WeBBA was even

more accurate as exact algorithms and, at the same time, it was almost as efficient as BBA.

5. Conclusions

In this contribution, the problem of cyclic sequence alignment was considered. An extensive comparison among the better known exact algorithms, and three sub-optimal (but faster) methods proposed by the authors in previous works, has been carried out. Experiments on both synthetic and real data demonstrated the usefulness of our approaches regarding both, accuracy and computing time.

Acknowledgments

The authors would like to thank A. Marzal for providing a robust implementation of all exact algorithms for computing the cyclic distance. Thanks are also due to G. Andreu for providing the real-world data set used in the experiments.

References

1. G. Andreu, A. Crespo and J. M. Valiente, "Selecting the toroidal self-organizing feature maps (TSOFM) best organized to object recognition," *Proc. ICNN97*, IEEE, Houston, Texas, USA, Vol. 2, 1997, pp. 1341–1346.
2. H. Bunke and U. Bühler, "Applications of approximate string matching to 2D shape recognition," *Patt. Recogn.* **26**, 12 (1993) 1797–1812.
3. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison-Wesley, 1992.
4. J. Gregor and M. G. Thomason, "Dynamic programming alignment of sequences representing cyclic patterns," *IEEE Trans. Patt. Anal. Mach. Intell.* **15**, 2 (1993) 129–135.
5. V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Cybern. Contr. Th.* **10**, 8 (1966) 707–710.
6. M. Maes, "On a cyclic string-to-string correction problem," *Inform. Process. Lett.* **35**, 2 (1990) 73–78.
7. A. Marzal and S. Barrachina, "Speeding up the computation of the edit distance for cyclic strings," *15th Int. Conf. Pattern Recognition*, Vol. 2, IEEE Computer Society, Barcelona, Spain, 2000, pp. 895–898.
8. R. Mollineda, E. Vidal and F. Casacuberta, "Efficient techniques for a very accurate measurement of dissimilarities between cyclic patterns," *Advances in Pattern Recognition*, Lecture Notes in Computer Science, Vol. 1876, Springer, 2000, pp. 337–346.
9. R. A. Mollineda, E. Vidal and F. Casacuberta, "A windowed version of the Bunke and Bühler algorithm to better approximate dissimilarities between cyclic patterns," *Proc. 5th Iberoam. Symp. Pattern Recognition*, Lisbon, 2000, pp. 311–321.
10. R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *J. Assoc. Comput. Mach.* **21** (1974) 168–173.

Ramón A. Mollineda received the B.S. and M.S. in computer science from Central University of Las Villas, Cuba, in 1995 and 1997, respectively. In 2001, he received the Ph.D. from Politechnical University of Valencia, Spain.

In 1998 he joined the Pattern Recognition and Human Language Technology group in the Politechnical University of Valencia, first with a grant from the Agencia Española de Cooperación Internacional (AECI) and later as a research fellow at the Informatic Technology Institute at the same university where he has been involved in several research projects.

Dr. Mollineda is a member of the Spanish Society for Pattern Recognition and Image Analysis (AERFAI) and the International Association for Pattern Recognition (IAPR).

His current topics of interest include statistical pattern recognition, nonparametric classifiers, feature selection and string matching.

Enrique Vidal received the *Licenciado* degree in physics in 1978 and the *Doctor en Ciencias Físicas* (Ph.D. in physics) degree in 1985, both from the Universitat de València.

From 1978 to 1986, he was with the university serving in computer system programming and teaching positions. In the same period he coordinated a research group in the fields of pattern recognition and automatic speech recognition. In 1986 he joined the Departamento de Sistemas Informáticos y Computación of the Universidad Politécnica de Valencia (UPV), where he has been serving as a Full Professor of the Facultad de Informática. In 1995 he joined the Instituto Tecnológico de Informática, where he has coordinated several projects on pattern recognition and machine translation. He is coleader of the Pattern Recognition and Human Language Technology group of the UPV.

Dr. Vidal is a member of the Spanish Society for Pattern Recognition and Image Analysis (AERFAI) and the International Association for Pattern Recognition (IAPR).

His current fields of interest include statistical and syntactic pattern recognition, and their applications to language, speech and image processing. In these fields, he has published more than one hundred papers in journals, conference proceedings and books.

Francisco Casacuberta received the Master and Ph.D. degrees in physics from the University of València, Spain, in 1976 and 1981, respectively.

From 1976 to 1979, he worked with the Department of Electricity and Electronics at the University of València as an FPI fellow. From 1980 to 1986, he was with the Computing Center of the University of València. Since 1980, he has been with the Department of Information Systems and Computation of the Polytechnic University of València first as an Associate Professor (Profesor Titular) and from 1990 as a Full Professor (Catedrático). Since 1981, he has been an active member of a research group in the fields of automatic speech recognition and machine translation.

Dr. Casacuberta is a member of the Spanish Society for Pattern Recognition and Image Analysis (AERFAI), which is an affiliate society of IAPR, the IEEE Computer Society and the Spanish Association for Artificial Intelligence (AEPIA).

His current research interests lie in the areas of speech recognition, machine translation, syntactic pattern recognition, statistical pattern recognition and machine learning.