ELSEVIER

Theoretical Computer Science 201 (1998) 275–279

Theoretical Computer Science

Note

An improved algorithm for solving the banded cyclic string-to-string correction problem

Kuo-Liang Chung *

Department of Information Management, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei, Taiwan 10672, China

Received July 1997; revised October 1997
Communicated by M. Crochemore

Abstract

The banded cyclic string-to-string correction (BCSSC) problem is a generalized version of the cyclic string-to-string correction (CSSC) problem, and has some applications in stereo matching and speech recognition. This note presents an improved algorithm for solving the BCSSC problem and the time complexity required ranges from $O(nm)$ to $O(nm \log b)$, where n and m are the lengths of the two strings and b is the allowable bandwidth. The result of this paper generalizes the result of Gregor and Thomason (1996) for solving the CSSC problem since the special version of the BCSSC problem can be transformed into the CSSC problem when setting $b = m$.

Keywords: Banded string-to-string correction, Cyclic patterns, Algorithms

1. Introduction

The cyclic string-to-string correction (CSSC) problem appears in many applications such as approximate polygonal shape recognition and speech recognition. Let n and m ($\leqslant n$) be the lengths of the two given strings, where each string may represent the boundary of a two-dimensional (2-D) object, a sequence of some periodic data, or a cyclic attribute string. Using the dynamic programming method of Wagner and Fischer [14, 5], Fu and Lu [6] presented an $O(nm^2)$-time algorithm for solving this CSSC problem. In [8, 2], the CSSC problem can be solved in $O(nm)$ time, but may not obtain the optimal solution. Then, Maes [11] presented an $O(nm \log m)$-time algorithm for solving the same problem. Gregor and Thomason [9] presented the first data-dependent algorithm whose time complexity ranges from $O(nm)$ to $O(nm^2)$. Recently, Gregor and

* Tel.: +886 2 7376771; fax: +886 2 7376777; e-mail: klchung@cs.ntust.edu.tw.

0304-3975/98/$19.00
PII S0304-3975(97)00280-6

Thomason [10] improved their previous algorithm [9] and the time complexity ranges from $O(nm)$ to $O(nm \log m)$.

Let the two given strings be $T = T_1 T_2 \ldots T_n$ and $P = P_1 P_2 \ldots P_m$. In some applications such as stereo matching for obtaining 3-D depth information [12, 1, 13] and dynamic time wrapping for speech recognition [7], for each i, T_i cannot be corrected to any P_j when j is too far away from i. That is, for each allowable correction, $|i - j|$ must be smaller than a bandwidth, say b; otherwise, the correction is not allowable. In other words, T_i can be corrected to P_j only under the constraint that $|i - j| < b$. This correction problem is called the banded CSSC (BCSSC) problem and the CSSC problem is a special version of the BCSSC problem when setting $b = m$. Recently, Chung [4] presented an $O(nm \log b)$-time algorithm for solving the BCSSC problem.

This note presents an improved algorithm for solving the BCSSC problem where the time complexity ranges from $O(nm)$ to $O(nm \log b)$. The result of this note not only improves the best-case time complexity of [4], but it also generalizes the result of Gregor and Thomason [10].

The remainder of this note is organized as follows. Section 2 introduces a naive algorithm for solving the BCSSC problem. Section 3 presents the proposed algorithm. Some concluding remarks are given in Section 4.

2. The naive algorithm

In this section, a naive algorithm for solving the BCSSC problem is presented. Let the length of the string $T(P)$ be $n = 10$ ($m = 10$) and the allowable bandwidth be $b = 3$. Let us consider P, as shown in Fig. 1. We have that T_1 can not be corrected to P_4; T_2 cannot be corrected to P_5; T_4 cannot be corrected to P_1; T_5 can neither be corrected to

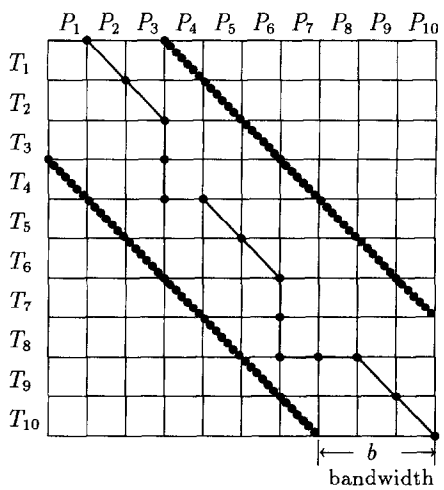

Fig. 1. The search space for BSSC problem.

P_1, nor to P_2. The BCSSC problem without considering all the cyclic strings of P, also called the banded string-to-string correction (BSSC) problem, becomes a minimal-cost path finding problem on the search space bounded by the two thick dotted lines shown in the figure that can be solved using dynamic programming.

Following some notations used in [4], three types of edit operations are defined to change one symbol in T into another symbol in P. Let $Re(a)=b$ be the replacement operation to change one nonempty symbol $a\in\sum$, where $\sum$ is the set of symbols, into another nonempty symbol $b\in\sum$; $De(a)=\bigwedge$ be the deletion operation to change one nonempty symbol a into the empty symbol $\bigwedge$ in $\sum$; $In(\bigwedge)$ be the insertion operation to change the empty symbol $\bigwedge$ into one nonempty symbol a. The cost of $Re(a)=b$ is defined to be $edit(a,b)=1$ when $a\neq b$ or $edit(a,b)=0$ when $a=b$. The cost of $De(a)=\bigwedge$ is defined to be $edit(a,\bigwedge)=1$. The cost of $In(\bigwedge)=a$ is defined to be $edit(\bigwedge,b)=1$. The task of finding the minimal-cost path in the banded search space is equal to finding an edit sequence such that the total cost of the search path is minimal.

The edit sequence in Fig. 1 consists of the following 14 edit operations: (1) $In(\bigwedge)=P_1$ (2) $Re(T_1)=P_2$ (3) $Re(T_2)=P_3$ (4) $De(T_3)=\bigwedge$ (5) $De(T_4)=\bigwedge$ (6) $In(\bigwedge)=P_4$ (7) $Re(T_5)=P_5$ (8) $Re(T_6)=P_6$ (9) $De(T_7)=\bigwedge$ (10) $De(T_8)=\bigwedge$ (11) $In(\bigwedge)=P_7$ (12) $In(\bigwedge)=P_8$ (13) $Re(T_9)=P_9$ (14) $Re(T_{10})=P_{10}$. Let the left-upper (right-bottom) corner of the above search space be at coordinate $(0,0)$ $((10,10))$. The set of vertex nodes along the minimal-cost path $P(0)$ is denoted by the left-upper node and the set of black circles within the banded zone. In Fig. 1, the vertex nodes along the path are represented by $\{v(0,0),v(0,1),v(1,2),v(2,3),\ldots,v(10,10)\}$.

Let $T[1..i]=T_1T_2\ldots T_i$ and $P[1..j]=P_1P_2\ldots P_j$ for $1\leqslant i\leqslant n$ and $1\leqslant j\leqslant m$. Let the cost table of the edit sequence for changing $T[1..i]$ into $P[1..j]$ be denoted by $EDIT[i,j]=edit(T[1..i],P[1..j])$ with the initial conditions $EDIT[0,k]=k$ and $EDIT[k,0]=k$ for $1\leqslant k<b$. The cost table $EDIT[i,j]$ is defined by $EDIT[i,j]=\min(EDIT[i-1,j]+edit(T_i,\bigwedge),\ EDIT[i,j-1]+edit(\bigwedge,P_j),EDIT[i-1,j-1]+edit(T_i,P_j))$. Since the concerned search space is $O(nd)$, the BSSC problem can be solved in $O(nd)$ time.

Suppose the string $P^{(1)}=P=P_1P_2\ldots P_m$ represents the boundary of one object. Two strings, $P^{(1)}$ and the cyclic string $P^{(i)}=P_iP_{i+1}\ldots P_mP_1\ldots P_{i-1}$, $2\leqslant i\leqslant m$, are said to be equivalent since the two strings represent the same boundary. This cyclic consideration extends the BSSC problem to the BCSSC problem. Since the length of $P^{(1)}$ is m, there are m cyclic right strings to be considered in the BCSSC problem. In the BCSSC problem, since we have m BSSC problems to be solved, the time complexity for solving the BCSSC problem is $O(mnd)$.

3. The proposed algorithm

Let us go back to Fig. 1 where T is changed into $P^{(1)}$, and let us denote $P(0)$ the minimal-cost path. Suppose the minimal-cost path for changing T into $P^{(2)}$ is $P(1)$. From Maes's results [11], we have the following properties.

Property 1. If the vertex node $v(i,j) \in P(0)$, there exists a $k \geqslant j$ such that the vertex node $v(i,k) \in P(1)$.

Property 2. If the vertex node $v(i,j) \in P(1)$, there exists a $k \leqslant j$ such that the vertex node $v(i,k) \in P(0)$.

From Properties 1 and 2, we have the following property.

Property 3. Let i,j,k be three nonnegative integers with $0 \leqslant i < j < k \leqslant m-1$ and let the minimal-cost paths $P(i)$ and $P(k)$ be noncrossing. Then the minimal-cost path $P(j)$ lies between $P(i)$ and $P(k)$. That is the path $P(i)$ and $P(j)$ are noncrossing; the path $P(j)$ and $P(k)$ are also noncrossing.

Property 3 can narrow the search space further and leads to an $O(nm \log m)$-time algorithm [11] for solving the CSSC problem.

From Proposition 2.3 in [10], let the cost of one deletion or one insertion be 1, then we have the following property.

Property 4. Let the cost of the path $P(i)$ be $COST(P(i))$ for $0 \leqslant i \leqslant m-1$. If $COST(P(i)) < COST(P(j))$, then $COST(P(i)) < COST(P(j \pm \ell))$ for $0 \leqslant \ell \leqslant \lceil (COST(P(j)) - COST(P(i)))/2 \rceil - 1$.

Property 4 implies that maybe a sub-search space with bandwidth $2 * \lceil (COST(P(j)) - COST(P(i)))/2 \rceil - 1$ can be omitted at a time. This leads to an algorithm for solving the CSSC problem using between $O(nm)$ and $O(nm \log m)$ time.

Without loss of generality, we assume that m is a multiple of b, i.e., $m = rb$, and b is a power of 2, i.e., $b = 2^s$. The proposed algorithm for solving the BCSSC problem consists of the following three steps.

Step 1: We compute the minimal-cost path $P(ib)$ for changing T into $P^{(ib+1)}$ for $0 \leqslant i \leqslant r-1$. It takes $O(nb)$ time to obtain the cost $EDIT[n, m+ib] = edit(T, P^{(ib+1)})$. Each $P(i)$ can be obtained in $O(nd)$ time. By Property 3, we know that the search space for finding $P(i)$ and the search space for finding $P(j)$, $i \neq j$, are disjoint from each other, so there are r paths, $P(0), P(b), P(2b), \ldots$, and $P(m-b)$, to be determined. The total time required in this stage is $O(nm) = O(nbr)$.

Step 2: Now, the paths $P(ib+1), P(ib+2), \ldots$, and $P(ib+b-1)$ for $0 \leqslant i \leqslant b-1$ are to be determined. By Property 4 and the result in [10], for each i, $P(ib+1)$, $P(ib+2), \ldots$, and $P(ib+b-1)$ can be determined using between $O(nb)$ and $O(nbs)$ time. All the related paths $P(ib+1)$, $P(ib+2), \ldots$, and $P(ib+b-1)$ for $0 \leqslant i \leqslant r-1$ can therefore be found using between $O(mn)$ and $O(nm \log b) = O(nbrs)$ time.

Step 3: The minimum among these m paths $P(0), P(1), P(2), \ldots$, and $P(m-1)$ can be found from $EDIT[n, m+j]$ for $0 \leqslant j \leqslant m-1$ using $O(m)$ time.

In summary, Step 1 takes $O(nm)$ time. The time bound required in Step 2 ranges from $O(mn)$ to $O(nm \log b)$ time. Step 3 takes $O(m)$ time. Combining the three time bounds, we have the main result.

Theorem 1. *The BCSSC problem can be solved using between* $O(mn)$ *and* $O(mn \log b)$ *time.*

4. Conclusions

We have presented an algorithm for solving the BCSSC problem. Our result improves the best-case time complexity of [4] and generalizes the result of Gregor and Thomason [10] when setting $b = m$. Applying the proposed algorithm to generalize the result of Bunke and Csirik [3] in the care of run length coded strings is our next research issue.

Acknowledgements

The author would like to thank the valuable comments of the anonymous referees. This research was supported in part by the National Science Council of R.O.C. under grants NSC87-2213-E011-001 and NSC87-2213-E011-003.

References

[1] D.H. Ballard, C.M. Brown, Computer Vision, Prentice-Hall, Englewood Cliffs, NJ, 1982, Section 3.4.
[2] H. Bunke, U. Bühler, Application of approximate string matching to 2D shape recognition, Pattern Recognition 26 (1993) 1797–1812.
[3] H. Bunke, J. Csirik, An algorithm for matching run-length coded strings, Computing 50 (1993) 297–314.
[4] K.L. Chung, A fast algorithm for stereo matching, Inform. Process. Lett. 61 (1997) 97–99.
[5] M. Crochemore, W. Rytter, Text Algorithms, Ch. 11, Oxford University Press, Oxford, 1994.
[6] K.S. Fu, S.Y. Lu, Size normalization and pattern orientation problems in syntactic clustering, IEEE Trans. Systems Man Cybernet. 9 (1979) 55–58.
[7] S. Furui, Digital Speech Processing, Synthesis, and Recognition, Section 8.5 Marcel–Dekker, New York, 1989.
[8] J.W. Gorman, O.R. Mitchell, F.P. Kuhl, Partial shape recognition using dynamic programming, IEEE Trans. on Pattern Anal. Mach. Intell. PAMI-10 (1988) 257–266.
[9] J. Gregor, M.G. Thomason, Dynamic programming alignment of sequences representing cyclic patterns, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-15 (1993) 129–135.
[10] J. Gregor, M.G. Thomason, Efficient dynamic programming alignment of cyclic strings by shift elimination, Pattern Recognition 29 (7) (1996) 1179–1185.
[11] M. Maes, On a cyclic string-to-string correction problem, Inform. Process. Lett. 35 (1990) 73–78.
[12] D. Marr, T. Poggio, Cooperative computation of stereo disparity, Science 194 (1976) 283–287.
[13] Y. Ohta, T. Kanade, Stereo by intra-and inter-scanline search using dynamic programming, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-7 (2) (1985) 139–154.
[14] R.A. Wagner, M.J. Fischer, The string-to-string correction problem, J. Assoc. Comput. Mach. 21 (1) (1974) 168–173.