Bit-parallel Computation for String Alignment

Yu, Yunqing
Faculty of Information Science and Electrical Engineering, Kyushu University

Baba, Kensuke
Faculty of Information Science and Electrical Engineering, Kyushu University

E, Hanmei
Faculty of Information Science and Electrical Engineering, Kyushu University

Murakami, Kazuaki
Faculty of Information Science and Electrical Engineering, Kyushu University

https://hdl.handle.net/2324/15562

出版情報：Lecture Series on Computer and Computational Sciences. 7 (A), pp.589-593, 2006-10. Brill
バージョン：
権利関係：

Bit-parallel Computation for String Alignment*

Yunqing Yu   Kensuke Baba[†]   Hanmei E   Kazuaki Murakami

Abstract

One of the most important ideas in data mining is alignment of two strings. This idea is based on a distance on strings and the most popular and simple one is the edit distance. For two strings of lengths m and n, the alignment and the edit distance is computed in $O(mn)$ time by dynamic programming approach. Bit-parallelism can speed-up the computation of the edit distance w times, where w is the word size of a computer, however this parallelism can not be applied straightforwardly to computing the alignment. This paper proposes a bit-parallel algorithm to compute all the possible alignments.

Keywords: String alignment, dynamic programming, bit-parallelism.

1 Introduction

In an analysis of biology in terms of strings or a mining on a large-scale data base, the basic and the most important idea is similarity on strings. For example, the base of some homology-search systems [5, 1] practically used in biology is the idea of *edit distance* [7] and its generalization [6], and more specific analysis are operated on the idea of *alignment* which is a correspondence of the characters.

The edit distance between two strings is the minimal number of edit operations which transforms one string to the other string, where the permitted operations are "insertion", "deletion", and "replacement" on a character, therefore it is the minimal value of the number of the edit operations with respect to all the possible correspondences between each characters of two strings. The correspondence of the characters is an alignment and formally represented by strings over the alphabet $\{I, D, R, M\}$, where the characters specify the operations "insertion", "deletion", "replacement", and "match", respectively [2]. For example, an alignment from the string `entry` to the string `empty` is represented by the string $MRIMDM$ and the edit distance is 3 as follows.

$$\begin{array}{cccccc} \texttt{e} & \texttt{n} & & \texttt{t} & \texttt{r} & \texttt{y} \\ \texttt{e} & \texttt{m} & \texttt{p} & \texttt{t} & & \texttt{y} \\ \hline M & R & I & M & D & M \end{array}$$

The edit distance for two strings of lengths m and n is computed in $O(mn)$ time by dynamic programming approach [7]. In this approach, for two strings, the cost matrix is evaluated, whose (i, j)-element is the edit distance between the prefix of length i of a string and the prefix of length j of the other string. The alignment is also obtained by remembering how each element was decided. Bit-parallelism is an approach of speed-up for string processing. The main idea is to represent each state of the processes as a bit-sequence, and compute plural states simultaneously by logical operations. Myers [4] introduced an algorithm based on this idea which computes a general case of the edit distance for two strings of lengths m and n in $O(\lceil m/w \rceil n)$ time, where w is the word size of a computer (with a suitable computational model). The essential idea of the algorithm by Myers is to compute a column of a cost matrix as a bit-vector in a constant time.

The idea of bit-parallelism could not be applied straightforwardly to computing the alignment since only the last line of a cost matrix is computed explicitly. Hyyrö [3] proposed a bit-parallel computation method according to the edit distance and recovering an optimal alignment. However, two or more alignments corresponding to an edit distance can exist and some applications require not to restrict the candidate of their data mining in the syntactic phase. Our aim in this paper is to construct a bit-parallel algorithm to compute all the alignments which correspond to the edit distance for given two strings.

2 Pointer Matrix

Let Σ be a finite set of characters. For an integer $n > 0$, Σ^n denotes the set of strings each of length n over Σ, and Σ^* denotes the set of the strings over Σ. Let ε be the empty string and $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. Then, for $s \in \Sigma^+$, $|s|$ denotes the length of s and $s[i]$ denotes the ith character of s for $1 \leq i \leq |s|$. The string $s_i = s[1]s[2] \cdots s[i]$ for $1 \leq i \leq |s|$ is a *prefix* of

*An edited version of this report was published in: *Selected Papers from the International Conference of Computational Methods in Sciences and Engineering 2006 (ICCMSE 2006), Lecture Series on Computer and Computational Sciences, vol.7(A), pp.589–593, VSP/Brill, Oct, 2006.*

[†]Faculty of Information Science and Electrical Engineering, Kyushu University, *baba@i.kyushu-u.ac.jp*

s. For $a \in \Sigma$, a^n denotes the string constructed by n a's.

Let $Sa = \{sa \mid s \in S\}$ for $S \subseteq \Sigma^*$ and $a \in \Sigma$. An *edit transcript* from $p \in \Sigma^*$ to $q \in \Sigma^*$ is a string on $\{I, D, R, M\}$, such that, the set $T(p, q)$ of the edit transcripts from p to q is

- $T(p, q) = \{\varepsilon\}$ if $pq = \varepsilon$;
- $T(p, q) = T(p, q')I$ if $p = \varepsilon$ and $q = q'a$ for $a \in \Sigma$;
- $T(p, q) = T(p', q)D$ if $p = p'a$ and $q = \varepsilon$ for $a \in \Sigma$;
- $T(p, q) = T(p, q')I + T(p', q)D + T(p', q')R$ if $p = p'a$, $q = q'b$, and $a \neq b$ for $a, b \in \Sigma$;
- $T(p, q) = T(p, q')I + T(p', q)D + t(p', q')M$ if $p = p'a$, $q = q'b$, and $a = b$ for $a, b \in \Sigma$.

An *optimal edit-transcript* from p to q is an edit transcript in which the number of occurrences of I, D, and R is minimal in $T(p, q)$, and the number is called the *edit distance* between p and q, denoted by $d(p, q)$.

Let $T_o(p, q)$ be the set of the optimal edit-transcripts from p to q. Then, the (i, j)-element of the *pointer matrix* P from $p \in \Sigma^m$ to $q \in \Sigma^n$ is the set of characters such that $P[i, j] = \{\varepsilon\}$ if $i = 0$ and $j = 0$, $\{I\}$ if $i = 0$ and $j \neq 0$, and $\{D\}$ if $i \neq 0$ and $j = 0$, and, for $1 \leq i \leq m$ and $1 \leq j \leq n$,

- $I \in P[i, j]$ if $T_o(p_i, q_{j-1})I \subseteq T_o(p_i, q_j)$,
- $D \in P[i, j]$ if $T_o(p_{i-1}, q_j)D \subseteq T_o(p_i, q_j)$,
- $R \in P[i, j]$ if $T_o(p_{i-1}, q_{j-1})R \subseteq T_o(p_i, q_j)$,
- $M \in P[i, j]$ if $T_o(p_{i-1}, q_{j-1})M \subseteq T_o(p_i, q_j)$.

3 Dynamic Programming Approach and Bit-parallelism

The edit distance and the optimal edit-transcripts for two strings are evaluated by computing the *cost matrix* whose (i, j)-element is the edit distance between the prefix of length i of a string and the prefix of length j of the other string. By the definition of the edit distance, the (i, j)-element of the cost matrix C for $p, q \in \Sigma^*$ is

$$\begin{aligned} C[i, j] &= \min\{C[i-1, j-1] + \delta(p[i], q[j]), \\ &\qquad C[i-1, j] + 1, \\ &\qquad C[i, j-1] + 1\} \end{aligned}$$

for $1 \leq i \leq |p|$ and $1 \leq j \leq |q|$, where $\delta(a, b)$ is 0 if $a = b$, and 1 if $a \neq b$ for $a, b \in \Sigma$. The base conditions are $C[0, j] = j$ and $C[i, 0] = i$. Since the (m, n)-element of the cost matrix is the edit distance between $p \in \Sigma^m$ and $q \in \Sigma^n$, the edit distance is obtained by computing mn elements of the matrix. The optimal edit-transcripts are evaluated by searching how each element was decided by the min operation, and this searching corresponds to backtracking on the pointer matrix from the last element.

Bit-parallelism is an idea of speed-up for computing the cost matrix. Consider the cost matrix for $p \in \Sigma^m$ and $q \in \Sigma^n$. Let

- $\Delta v_j[i] = d(p_i, q_j) - d(p_{i-1}, q_j)$ for $1 \leq i \leq m$ and $0 \leq j \leq n$,
- $\Delta h_j[i] = d(p_i, q_j) - d(p_i, q_{j-1})$ for $0 \leq i \leq m$ and $1 \leq j \leq n$.

Since $C[i, 0] = i$ for $1 \leq i \leq m$ and $C[0, j] = j$ for $1 \leq j \leq n$, $C[m, n]$ is obtained by computing the vectors Δv_j and Δh_j for $1 \leq j \leq n$. If we assume that the bit-wise operations are done simultaneously in an unit time, computing a Δv_i takes $O(\lceil m/w \rceil)$ time for a word-size w. Therefore, this idea yields w times speed-up, however can not be applied straightforwardly to edit-transcripts since only the last line of a cost matrix is computed explicitly.

4 Bit-parallel Algorithm for Pointer Matrix

We introduce a bit-parallel algorithm to compute the optimal edit-transcripts for given two strings. First, we prepare the following bit-vectors:

- $E_j[i]$ is 1 if $p[i] = q[j]$, and 0 otherwise, for $1 \leq i \leq m$ and $1 \leq j \leq n$;
- $V_j^p[i]$ ($V_j^m[i]$) is 1 if $\Delta v_j[i] = 1$ (resp. -1), and 0 otherwise, for $1 \leq i \leq m$ and $0 \leq j \leq n$;
- $H_j^p[i]$ ($H_j^m[i]$) is 1 if $\Delta h_j[i] = 1$ (resp. -1), and 0 otherwise, for $0 \leq i \leq m$ and $1 \leq j \leq n$.

Then, by the definition, we have the following relations between the pointer matrix and the bit-vectors: for $1 \leq i \leq m$ and $1 \leq j \leq n$,

- $I \in P[i, j]$ if and only if $H_j^p[i] = 1$;
- $D \in P[i, j]$ if and only if $V_j^p[i] = 1$;
- $R \in P[i, j]$ if and only if $E_j[i] = 0$, $H_j^m[i-1] = 0$, and $V_{j-1}^m[i] = 0$;
- $M \in P[i, j]$ if and only if $E_j[i] = 1$.

To apply the previous relations to a bit-parallel algorithm, we represent the pointer matrix P by the bit-sequences P^I, P^D, P^R, and P^M such that $P_j^a[i]$ is 1 if $a \in P[i, j]$, and 0 otherwise for $a \in \{I, D, R, M\}$. Then, an algorithm to compute the optimal edit-transcripts is:

```
1. input $p \in \Sigma^m$ and $q \in \Sigma^n$;
2. for $j = 1, 2, \ldots, n$, compute $E_j$ and $P^M_j = E_j$;
3. $V^p = 1^m$ and $V^m = 0$;
4. for $j = 1, 2, \ldots, n$
    4.1. $X_v = E_j \vee V^m$ and $X_h = E_j \vee (((E_j \wedge V^p) + V^p) \oplus V^p)$;
    4.2. $H^p = V^m \vee \neg(X_h \vee V^p)$, $P^I_j = H^p$, and $H^m = V^p \wedge X_h$;
    4.3. $(H^p \ll) + 1$ and $H^m \ll$;
    4.4. $V^p = H^m \vee \neg(X_v \vee H^p)$, $P^D_j = V^p$, and $V^m = H^p \wedge X_v$;
    4.5. $P^R_j = \neg(E_j \vee H^m \vee V^m)$;
5. output $P^I$, $P^D$, $P^R$, and $P^M$ as $P$,
```

where the notation $\vee$, $\wedge$, $\neg$, $\oplus$, and $\ll$ represent the bit-operations "OR", "AND", "NOT", "XOR", and "shift to left", respectively.

5 Conclusion

We proposed the bit-parallel algorithm to compute all the optimal edit-transcripts for two strings as the pointer matrix. The pointer matrix can be obtained as the cost matrix is computed. The backtracking in [3] is one of the methods to find an optimal edit-transcript from the pointer matrix and the edit transcript is the first string under a lexicographical order. Clearly, our algorithm can find such an edit transcript by connecting the corresponding elements of the pointer matrix as the elements are computed, and the lexicographical order depends on which matrix is considered in P^I, P^D, P^R, and P^M.

References

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215(3):403–410, 1990.

[2] D. Gusfield. *Algorithms on Strings, Trees, and Sequences.* Cambridge University Press, New York, 1997.

[3] H. Hyyrö. A note on bit-parallel alignment computation. In *Proc. Prague Stringology Conference '04 (PSC2004)*, 2004.

[4] G. Myers. A fast bit-vector algorithm for approximate string matching based on dynamic programming. *J. ACM*, 46(3):395–415, May 1999.

[5] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. In *Proc. Natl. Acad. Sci. USA*, volume 85, pages 2444–2448, April 1988.

[6] T. F. Smith and M. S. Waterman. Identification of common molecular subseqences. *J. Mol. Biol.*, 147:195–197, 1981.

[7] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, January 1974.