Algorithms for the Merged-LCS Problem and Its Variant with Block Constraint *

Kuo-Si Huang, Chang-Biau Yang, Kuo-Tsung Tseng,
Hsing-Yen Ann and Yung-Hsing Peng
Department of Computer Science and Engineering
National Sun Yat-sen University, Kaohsiung, Taiwan
cbyang@cse.nsysu.edu.tw

Abstract

The longest common subsequence (LCS) problem was widely discussed and regarded as the measurement for the relationship among sequences. Let A' and B' are the subsequences of A and B, respectively. The merged-sequence $E(A, B)$ is composed of A' and B'. In this paper, we consider the merged-LCS problem, denoted as $LCS(T, E(A, B))$, for measuring the relationship among three sequences T, A and B. We first propose an algorithm for solving the merged-LCS problem, whose time complexity is $O(n^3)$, where n is the sequence length. We further discuss the variant version of the merged-LCS problem with block constraint, that is, the block information of A and B is given in advance. For the blocked merged-LCS problem, we propose an algorithm with time complexity $O(n^2m)$, where m is the number of blocks. An improved $O(n^2 + nm^2)$ algorithm is proposed for the same blocked merged-LCS problem by using the concept of preprocessing.

Key words: longest common subsequence, dynamic programming, sequence merging, synteny, DNA interleaving

1 Introduction

For a sequence S, over a finite symbol set Σ, a *subsequence* of S is the sequence P obtained from S by deleting zero or more (not necessarily contiguous) symbols. The *longest common subsequence* (LCS) problem is defined as follows: Given k ($k \geq 2$) sequences, find a longest sequence P such that P is a common subsequence of these k sequences [13]. If the number of sequence k is not fixed, the problem is NP-complete even over binary alphabets [7]. In general, many applications and researches are focusing on the LCS problem for only two sequences. For two sequences $A =$ `cgatacc` and $B =$ `aattcgc`, sequences `cc`, `cgc`, `atcc`, and `aacc` are common subsequences of A and B; `atcc` and `aacc` are two LCS's of A and B, denoted as $LCS(A, B)$, with length 4. The LCS problem is a famous and classical problem in computer science and molecular biology. The parts of common subsequences can viewed as the identical parts of sequences and can help us to reconstruct an alignment of these sequences.

Let A' and B' denote the subsequences of A and B, respectively. The merged-sequence $E(A, B)$ is composed of A' and B'. That is, we can separate $E(A, B)$ into two disjoint sequences A' and B'. Note that a sequence is also one of its subsequences by deleting nothing, such as A is a subsequence of A. For understanding the relationship among sequences T, A and B, we can first merge A and B to a specific interleaving sequence $E(A, B)$; then find the relationship, such as LCS or alignment, of T and $E(A, B)$. Figure 1 shows an example for illustrating the alignment relationship of T, A and B. We define the merged-LCS problem, denoted as $LCS(T, E(A, B))$, to measure the interleaving relationship among sequences T, A and B. It is not so trivial because there are many various sequences of $E(A, B)$ based on different interleaving recombination of A and B. Note that in this paper, $LCS(A, B)$ may be used to represent the length value of $LCS(A, B)$ or the string of $LCS(A, B)$ alternatively if there is no ambiguity of its meaning.

The riffle shuffle, which is a shuffle trick when we play poker cards, merges two halves of cards into one interlacing deck. If only one shuffle is performed, it is easy to find the mapping relation between the interlacing deck and two halves.

*This research work was partially supported by the National Science Council of the Republic of China under contract NSC-94-2213-E-110-051.

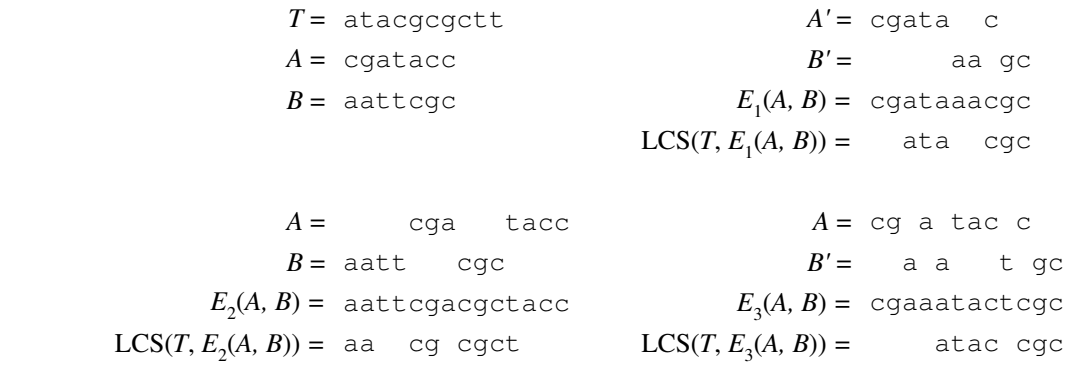

Figure 1: An example for illustrating the merged-LCS problem, denoted as $LCS(T, E(A, B))$, of sequences T, A and B.

The people use this shuffle trick for playing poker everyday. It is interesting that the riffle shuffle phenomenon exists in the genomes of organisms, but it is more complex than that in poker cards. The reasons of complexity in genomes are the existence of identical cards (identical symbols), the missing of cards for the deck and its original halves (gene loss), and the clones of some cards (DNA duplication). We can also enjoy these complex specificities in playing poker cards with several decks and allowing some cards missing, if we want. But *synteny* [3, 4, 12], which means the order of specific cards (the order of genes in chromosomes) is conserved over halves (different organisms), is the important and especial characteristic in the riffle shuffle.

For example, the one-time riffle shuffle is shown in the genomic comparison of two yeast-species *Kluyveromyces waltii* and *Saccharomyces cerevisiae* by Kellis et al. [4], which provided the support for genome duplication [9, 8]. In this genomic comparison, they tried to find the doubly conserved synteny (DCS) blocks of the two genomes, which are the synteny blocks that each region of *K. waltii* is related to two regions of *S. cerevisiae*, as the support for the whole-genome duplication (WGD). In addition, Jaillon et al. proposed the genomic comparison of human and *Tetraodon*, a teleost fish, chromosomes [3]. The synteny blocks and the WGD are also revealable in the genomic comparison of human and *Tetraodon*. Sequence alignment is helpful to find the DCS blocks. The researchers must analyze the alignment results of whole genome and identify the DCS blocks. The alignment can identify the relationship of similar regions by the local alignment [11]. But it is not an automatic approach to report the similarity of DCS blocks. For example, we may desire to find which are the best halves of synteny blocks to form a known target sequence or to find which is the best target sequence consists of given halves.

In this paper, we propose a measurement based on LCS to quantify the synteny score for a target sequence and two merging sequences. The rest of this paper is organized as follows. In Section 2, we first introduce the merged-LCS problem, and give the algorithms for solving the problem. Our algorithm for solving the merged-LCS problem requires $O(n^3)$ time, where n is the sequence length. In Section 3, for blocked merged-LCS problem, a variant of the merged-LCS problem with block constraint, we propose two algorithms whose time complexities are $O(n^2m)$ and $O(n^2 + nm^2)$, respectively, where m is the number of blocks. And finally, a conclusion will be given in Section 4.

2 Merged-LCS

Given a target-sequence $T = t_1t_2 \cdots t_{|T|}$, and a pair of merging sequences $A = a_1a_2 \cdots a_{|A|}$ and $B = b_1b_2 \cdots b_{|B|}$, where $|T|$, $|A|$ and $|B|$ denote the lengths of T, A and B, respectively. A merged-sequence $E(A, B) = e_1e_2e_3 \cdots e_{|E(A,B)|}$ is obtained from merging subsequences of A and B, where $e_i \in \{a_j | 1 \leq j \leq |A|\} \cup \{b_k | 1 \leq k \leq |B|\}$ and $1 \leq i \leq |E(A, B)|$; that is, there exists one subsequence S of $E(A, B)$ such that S is a subsequence of A and $E(A, B) - S$ is a subsequence of B. The merged-LCS problem is to find the LCS of T and $E(A, B)$, denoted by $LCS(T, E(A, B))$ $= LCS(t_1t_2 \cdots t_{|T|}, E(a_1a_2 \cdots a_{|A|}, b_1b_2 \cdots b_{|B|}))$. Note that based on the definition, the merged-sequence of two picked sequences may not be

unique.

For example, consider sequences T = atacgcgctt, A = cgatacc, and B = aattcgc. $E_1(A,B)$ = cgataaacgc, $E_2(A,B)$ = aattcgacgctacc, and $E_3(A,B)$ = cgaaatactcgc are some merged-sequences of A and B. We can find that $LCS(T, E_1(A,B))$ = atacgc, $LCS(T, E_2(A,B))$ = aacgcgct, $LCS(T, E_3(A,B))$ = ataccgc. In this case, $LCS(T, E(A,B))$ = $LCS(T, E_2(A,B))$ with length 8. Figure 1 is the aligned illustration for this example.

One may design a simple heuristic for solving the merged-LCS problem. One can first find $LCS(T,A)$ and $LCS(T - LCS(T,A), B)$, and then merge them into an LCS candidate. Another candidate is the merged-sequence of $LCS(T,B)$ and $LCS(T - LCS(T,B), A)$. Unfortunately, the above example gives us a counter example for this heuristic: $LCS(T,A)$ + $LCS(T - LCS(T,A), B)$ = ataccgt, $LCS(T,B)$ + $LCS(T - LCS(T,B), A)$ = atcgcct, but $LCS(T, E(A,B))$ = aacgcgct. In essential, this heuristic did not consider all possible subsequences of A and B for finding the best candidate.

Here, we propose a dynamic programming algorithm for solving the merged-LCS problem. For abbreviating the expressions, let $L(i,j,k)$ = $LCS(t_1t_2\cdots t_i,\ E(a_1a_2\cdots a_j,\ b_1b_2\cdots b_k))$, where $E(a_1a_2\cdots a_j, b_1b_2\cdots b_k)$ denotes the merged-sequence of $a_1a_2\cdots a_j$ and $b_1b_2\cdots b_k$. The algorithm for solving the merged-LCS problem is given as follows.

Algorithm Merged-LCS

Input: A target sequence T, and two merging sequences A, and B.

Output: $LCS(T, E(A,B)) = L(|T|, |A|, |B|)$.

Step 1. Initialize $L(0,j,k) = L(i,0,k) = L(i,j,0)$ = 0, for $0 \le i \le |T|$, $0 \le j \le |A|$ and $0 \le k \le |B|$.

Step 2. For $1 \le i \le |T|$, $1 \le j \le |A|$ and $1 \le k \le |B|$, calculate $L(i,j,k)$ by the dynamic programming equation in Figure 2.

Step 3. Return $LCS(T, E(A,B)) = L(|T|, |A|, |B|)$.

The time complexity of Algorithm Merged-LCS is $O(n^3)$, where n is the length of the longest sequence. For reducing the space complexity, let $L^R(i,j,k) = LCS(t_it_{i-1}\cdots t_2t_1,\ E(a_ja_{j-1}\cdots a_2a_1,\ b_kb_{k-1}\cdots b_2b_1))$. The merged-LCS can also be computed by the recurrence formula $L(i,j,k) = max_{j,k}\{L(\lfloor i/2\rfloor, j, k) + L^R(\lfloor i/2\rfloor + 1, j+1, k+1)\}$. We can reduce the space complexity to $O(n^2)$ by the same concept of divide and conquer in the LCS algorithm [2].

3 Merged-LCS with Block Constraint

In real applications, the merged-LCS problem would be accomplished with some constraints. Let us consider a specific version of the merged-LCS problem with the block constraint. The block constraint is based on the known sequence-segments of genes or the specific DNA segments that we are interested in. We can find the block constraint in many areas such as DNA recombination, evolutionary analysis and genome duplication [3, 4, 8, 9, 12].

The merged-LCS problem with block constraint is described as follows, abbreviated as the blocked merged-LCS problem. Given a target sequence $T = t_1t_2\cdots t_{|T|}$, and a pair of merging block-sequences $A = A_1A_2\cdots A_\alpha = a_1a_2\cdots a_{|A|}$ and $B = B_1B_2\cdots B_\beta = b_1b_2\ldots b_{|B|}$, where α, β are the block numbers of A and B, respectively; $|T|$, $|A|$ and $|B|$ are the lengths of T, A and B, respectively. A blocked merged-sequence $E(A,B) = E_1E_2E_3\cdots E_\varepsilon = e_1e_2e_3\cdots e_{|E(A,B)|}$ is composed of the blocks in A and B, where $E_i \in \{A_j | 1 \le j \le \alpha\} \cup \{B_k | 1 \le k \le \beta\}$ and $1 \le i \le \varepsilon$, ε is the number of blocks in $E(A,B)$. That is, there exists one block-subsequence S of $E(A,B)$ such that S is a block-subsequence of A and $E(A,B) - S$ is a block-subsequence of B. Here, one block-subsequence of a block-sequence A can be obtained by deleting zero or more blocks from A. The blocked merged-LCS problem is to find the block-constrained LCS of T and $E(A,B)$, denoted by $bLCS(T, E(A,B)) = bLCS(t_1t_2\cdots t_{|T|},\ E(A_1A_2\cdots A_\alpha,\ B_1B_2\cdots B_\beta))$.

Following the example in Section 2 and adding some block constraints, consider sequences T = atacgcgctt, $A = A_1A_2$ = cgat acc, and $B = B_1B_2B_3$ = aat tc gc. In detail, blocks A_1 = cgat, A_2 = acc, B_1 = aat, B_2 = tc, and B_3 = gc. By the block constraint, $E_4(A,B) = A_1B_1B_2A_2$ = cgat aat tc acc, $E_5(A,B) = B_1A_1A_2B_2$ = aat cgat acc tc, and $E_6(A,B) = B_1B_2A_1B_3A_2$ = aat tc cgat acc gc are some blocked merged-sequences of A and B. We can compute that $bLCS(T, E_4(A,B))$

$$L(i,j,k) = \max \begin{cases} L(i-1,j-1,k)+1 & \text{if } t_i = a_j, \\ L(i-1,j,k-1)+1 & \text{if } t_i = b_k, \\ \max \begin{cases} L(i-1,j,k) \\ L(i,j-1,k) \\ L(i,j,k-1) \end{cases} & \text{if } t_i \neq a_j \text{ and } t_i \neq b_k. \end{cases} \quad (1)$$

Figure 2: The dynamic programming formula for Algorithm Merged-LCS.

= ataccc, $bLCS(T, E_5(A,B))$ = atcgcct, $bLCS(T, E_6(A,B))$ = ataccc. In this block-constrained case, $bLCS(T, E(A,B))$ = $bLCS(T, E_5(A,B))$ with length 7. Note that we can not find the same solution of $LCS(T, E_2(A,B))$ because of the additional block-constraints of A and B. Figure 3 is the aligned illustration for this blocked merged-LCS example.

Similar to the merged-sequence, the blocked merged-sequence of two merging block-sequences may not be unique based on the definition. Here, we do not specify any block constraint of the target sequence T; so we do not expect that any block of $E(A,B)$ should align to a specific block of T. If there is also block constraint in T, we can easily extend Algorithm Merged-LCS to solve the problem with that each block is regarded as a specific symbol. In addition, if the target sequence is just sequenced, its annotation and block information may not be available yet. In the blocked merged-LCS problem, the absence of block information of any one merging sequence, A or B, is allowed because we can regard each symbol as one block. Our algorithm for solving the blocked merged-LCS problem is given as follows.

Algorithm Blocked Merged-LCS

Input: A target sequence T, and two merging block-sequences A and B.

Output: $bLCS(T, E(A,B)) = L(|T|, |A|, |B|)$.

Step 1. Initialize $L(0,j,k) = L(i,0,k) = L(i,j,0) = 0$, for $0 \le i \le |T|$, $0 \le j \le |A|$ and $0 \le k \le |B|$.

Step 2. For $1 \le i \le |T|$, $1 \le j \le |A|$ and $1 \le k \le |B|$, calculate $L(i,j,k)$ by the dynamic programming equation in Figure 4.

Step 3. Return $bLCS(T, E(A,B)) = L(|T|, |A|, |B|)$.

All of the three cases in this algorithm are related to an important characteristic, the end symbol of a block (EOB). Consider the symbol index of A or B in $L(i,j,k)$, the symbol index can be changed validly only when the symbol index is EOB. If the index is not at the end of a block, we must move this index until it reaches an EOB. There are $O(n)$ and $O(m)$ positions that should be considered for T and the EOB's of the two merging sequences A and B, respectively, where $m = max\{\alpha, \beta\}$. When the positions of T and one EOB are fixed, $O(n)$ positions are needed to be compared in the worst case. We conclude that the time complexity of Algorithm Blocked Merged-LCS is $O(n^2m)$, which is less than that of Algorithm Merged-LCS.

Let us consider a different viewpoint of the blocked merged-LCS problem. If we perform some preprocessing rather than the straightforward computation, we can design an algorithm with lower time complexity. For sequences T, A_1 and B_3, we can spend $O(n^2)$ time to compute $LCS(T, A_1)$ and $LCS(T, B_3)$, and store the necessary information S-table [5, 6] in advance. If we want to compute $LCS(T, A_1 + B_3)$, where $A_1 + B_3$ is the concatenating sequence of A_1 and B_1, we can reuse the S-table. Here, we will invoke an algorithm that can merge two preprocessed LCS with $O(n^2)$ time for preprocessing [5, 6] and $O(n)$ time for merging [1, 6]. We may note that Algorithm SMAWK [1] is a linear time merging algorithm but it is a recursive one. Landau and Ziv-Ukelson proposed a non-recursive merging algorithm for the edit distance problem with the same complexity [6].

Let $L^b(i,j,k) = LCS(t_1t_2\cdots t_i, E(A_1A_2\cdots A_j, B_1B_2\cdots B_k))$, where $1 \le i \le |T|$, $1 \le j \le \alpha$, $1 \le k \le \beta$, and $E(A_1A_2\cdots A_j, B_1B_2\cdots B_k)$ denote the block merged-sequence of $A_1A_2\cdots A_j$ and $B_1B_2\cdots B_k$. Let $T[i_1, i_2]$ be one substring $t_{i_1}t_{i_1+1}\cdots t_{i_2-1}t_{i_2}$ of T, and $T[i_1, i_2]$ be empty if $i_1 > i_2$. The vector $V^b(j,k) = L^b(1,j,k)L^b(2,j,k) \cdots L^b(|T|,j,k)$ stores the LCS of $L^b(i,j,k)$ for each prefix $T[1,i]$ of T and $E(A_1A_2\cdots A_j, B_1B_2\cdots B_k)$. For a specific position x, $L^b(x,j,k) = \max_{0\le i\le x}\{L^b(i,j-1,k) + LCS(T[i+1,x], A_j), L^b(i,j,k-1) + LCS(T[i+1,x], B_k)\}$. The S-table of sequences T and X, denoted as $S(T,X)$, stores

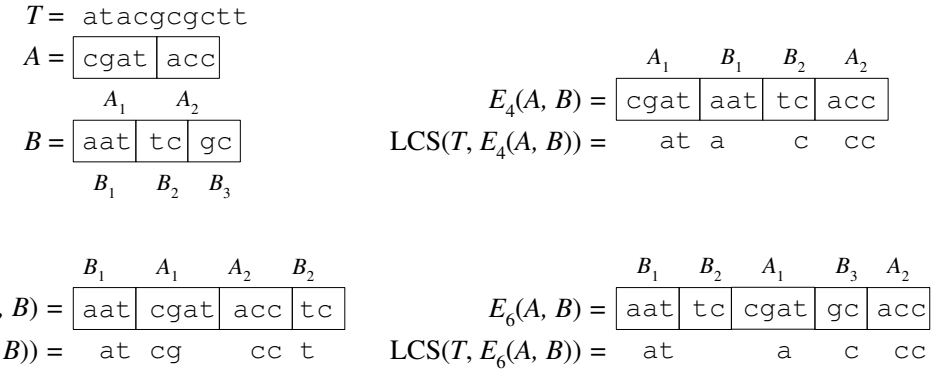

Figure 3: An illustration for the blocked merged-LCS problem, denoted as $bLCS(T, E(A, B))$, of sequences T, A and B, where $A = A_1A_2$ and $B = B_1B_2B_3$ are the block information that is known in advance.

$$L(i,j,k) = \max \begin{cases} \max \begin{cases} L(i-1,j-1,k)+1 & \text{if } t_i = a_j, \\ L(i,j-1,k) & \text{if } t_i \neq a_j, \\ L(i-1,j,k-1)+1 & \text{if } t_i = b_k, \\ L(i,j,k-1) & \text{if } t_i \neq b_k, \\ L(i-1,j,k) & \text{if } t_i \neq a_j \text{ and } t_i \neq b_k, \end{cases} & \text{if } a_j = \text{EOB and } b_k = \text{EOB}, \\ \max \begin{cases} L(i-1,j-1,k)+1 & \text{if } t_i = a_j, \\ L(i,j-1,k) & \text{if } t_i \neq a_j, \\ L(i-1,j,k) & \text{if } t_i \neq a_j, \end{cases} & \text{if } a_j \neq \text{EOB and } b_k = \text{EOB}, \\ \max \begin{cases} L(i-1,j,k-1)+1 & \text{if } t_i = b_k, \\ L(i,j,k-1) & \text{if } t_i \neq b_k, \\ L(i-1,j,k) & \text{if } t_i \neq b_k, \end{cases} & \text{if } a_j = \text{EOB and } b_k \neq \text{EOB}. \end{cases} \tag{2}$$

Figure 4: The dynamic programming formula for Algorithm Blocked Merged-LCS.

the LCS length of X and each suffix $T[i, |T|]$ of T. The symbol $\oplus$ denotes the merging operation of a vector and an S-table. Our improved algorithm for solving the blocked merged-LCS problem by using the S-table is in the following.

Algorithm Blocked Merged-LCS 2

Input: A target sequence T, and two merging block sequences A and B.

Output: $bLCS(T, E(A, B)) = L(|T|, \alpha, \beta)$.

Step 1. Compute and store $S(T, A_j)$ and $S(T, B_k)$ from $LCS(T, A_j)$ and $LCS(T, B_k)$, respectively, where $1 \leq j \leq \alpha$, $1 \leq k \leq \beta$.

Step 2. Initialize $L^b(i, 0, 0) = 0$, for $1 \leq i \leq |T|$.

Step 3. Compute $V^b(j,k) = \max\{V^b(j-1,k) \oplus S(T, A_j), V^b(j,k-1) \oplus S(T, B_k)\}$, for $1 \leq j \leq \alpha$, $1 \leq k \leq \beta$.

Step 4. Return $bLCS(T, E(A, B)) = L(|T|, \alpha, \beta)$.

In Step 1, it requires $O(n^2)$ time and space to compute and store the S-table, where n is the maximum length of T, A, and B. In Step 2, the initialization for $L^b(i, 0, 0)$ requires $O(n)$ time. In Step 3, the length of vector $V^b(j,k)$ is $n = |T|$. The merging operation $\oplus$ requires $O(n)$ time and space based on the *totally monotone* property of LCS matrix [1, 6, 5, 10]. We can obtain the merged vectors $V^b(j-1,k) \oplus S(T, A_j)$ and $V^b(j,k-1) \oplus S(T, B_k)$ in $O(n)$ time. The vector $V^b(j,k)$ can be computed in $O(n)$ time by simply comparing $V^b(j-1,k) \oplus S(T, A_j)$ and $V^b(j,k-1) \oplus S(T, B_k)$ for each position in T. The merging operation $\oplus$ is invoked for each j and k, where $1 \leq j \leq \alpha \leq m$ and $1 \leq k \leq \beta \leq m$. So, vector $V^b(j,k)$ can be computed in $O(nm^2)$ time. Here, we can store all vectors of $V^b(j,k)$, for trac-

ing the LCS path; this will also require $O(nm^2)$ storage. Step 4 returns the value $L^b(|T|, \alpha, \beta)$ merely. The time or space complexity for this algorithm is $O(n^2+nm^2)$. Figure 5 shows the details of $V^b(0,1)$, $V^b(1,0)$, $S(T,A_1)$ and $S(T,B_1)$, which we can use to compute $V^b(1,1)$.

4 Conclusion

In this paper, we introduce the merged-LCS problem and its variant with the block constraint. The merged-LCS problem can be used as one measurement of doubly conserved synteny block and thus it is a real problem in the molecular biology research. We propose first an algorithm for the merged-LCS problem, whose time complexity and space complexity are $O(n^3)$ and $O(n^2)$, respectively, where n is the sequence length. For blocked merged-LCS problem, we propose two algorithms with time complexities $O(n^2m)$ and $O(n^2+nm^2)$, where m is the number of blocks.

There are still some problems related to this paper for further study. In the whole genome comparison, the large-scale input sequence may cause that the traditional algorithms cannot work because of the limitation of computer storage. We can develop a mechanism by focusing on the small pieces of the genome to find the high score regions. In addition, our algorithms here focus on the LCS problem. For protein sequences or considering the effect of gaps, we may extend the merged-LCS problem to the merged-alignment problem. Furthermore, we can extend the problem to the multiple merged-LCS or multiple merged-alignment problems for merging several sequences. It is reasonable that the events of gene loss take place several times in the evolution. So it is worthy to develop efficient and feasible algorithms, which can run in current computers and return the result in a feasible time, for solving the multiple merged-LCS or merged-alignment problems.

References

[1] A. Aggarwal, M. Klawe, S. Moran, P. Shor, and R. Wilber, "Geometric applications of a matrix searching algorithm," *Algorithmica*, Vol. 2, pp. 195–208, 1987.

[2] D. S. Hirschberg, "A linear space algorithm for computing maximal common subsequence," *Communications of the ACM*, Vol. 24, pp. 664–675, 1975.

[3] O. Jaillon, J.-M. Aury, F. Brunet, J.-L. Petit, N. Stange-Thomann, E. Mauceli, L. Bouneau, C. Fischer, C. Ozouf-Costaz, A. Bernot, S. Nicaud, D. Jaffe, S. Fisher, G. Lutfalla, C. Dossat, B. Segurens, C. Dasilva, M. Salanoubat, M. Levy, N. Boudet, S. Castellano, V. Anthouard, C. Jubin, V. Castelli, M. Katinka, B. Vacherie, C. Biemont, Z. Skalli, L. Cattolico, J. Poulain, V. de Berardinis, C. Cruaud, S. Duprat, P. Brottier, J.-P. Coutanceau, J. Gouzy, G. Parra, G. Lardier, C. Chapple, K. J. McKernan, P. McEwan, S. Bosak, M. Kellis, J.-N. Volff, R. Guigo, M. C. Zody, J. Mesirov, K. Lindblad-Toh, B. Birren, C. Nusbaum, D. Kahn, M. Robinson-Rechavi, V. Laudet, V. Schachter, F. Quetier, W. Saurin, C. Scarpelli, P. Wincker, E. S. Lander, J. Weissenbach, and H. R. Crollius, "Genome duplication in the teleost fish *tetraodon nigroviridis* reveals the early vertebrate proto-karyotype," *Nature*, Vol. 431, No. 7011, pp. 946–957, 2004.

[4] M. Kellis, B. W. Birren, and E. S. Lander, "Proof and evolutionary analysis of ancient genome duplication in the yeast *saccharomyces cerevisiae*," *Nature*, Vol. 428, No. 6983, pp. 617–624, 2004.

[5] G. M. Landau, B. Schieber, and M. Ziv-Ukelson, "Sparse lcs common substring alignment," *Information Processing Letters*, Vol. 88, pp. 259–270, 2003.

[6] G. M. Landau and M. Ziv-Ukelson, "On the common substring alignment problem," *Journal of Algorithms*, Vol. 41, pp. 338–354, 2001.

[7] D. Maier, "The complexity of some problems on subsequences and supersequences," *Journal of the ACM*, Vol. 25, pp. 322–336, 1978.

[8] S. Ohno, *Evolution by Gene Duplication*. Springer-Verlag, Heidelberg, 1970.

[9] G. Panopoulou and A. J. Poustka, "Timing and mechanism of ancient vertebrate genome duplications - the adventure of a hypothesis," *TRENDS in Genetics*, Vol. 21, No. 10, pp. 559–567, 2005.

[10] J. P. Schmidt, "All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings," *SIAM Journal on Computing*, Vol. 27, No. 4, pp. 972–992, 1998.

[11] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, Vol. 147, pp. 195–197, 1981.

[12] D. Vallenet, L. Labarre, Z. Rouy, V. Barbe, S. Bocs, S. Cruveiller, A. Lajus, G. Pascal, C. Scarpelli, and C. Medigue, "MaGe: a microbial genome annotation system supported

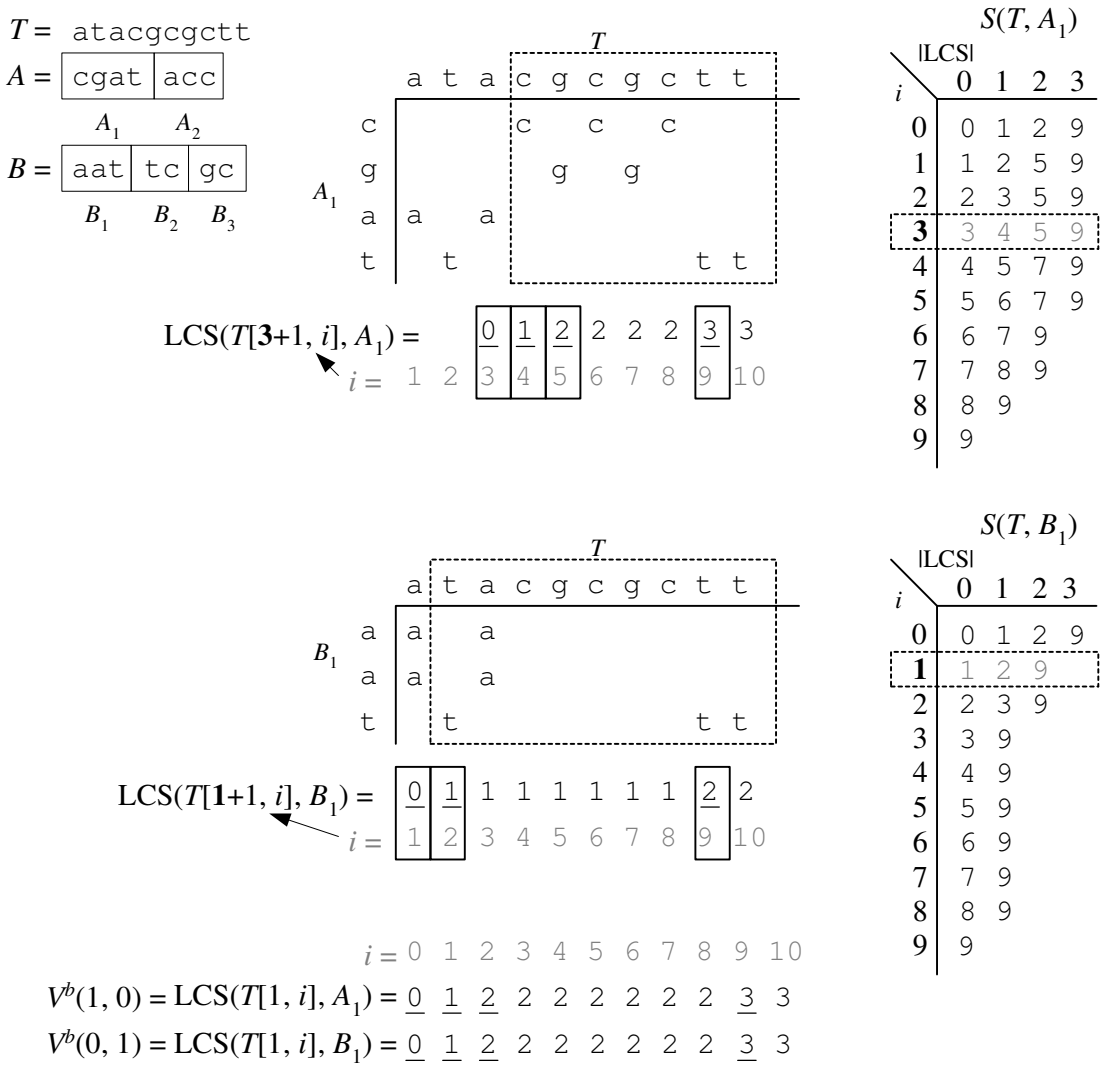

Figure 5: The details of of $V^b(0,1)$, $V^b(1,0)$, $S(T,A_1)$, $S(T,B_1)$ and $V^b(1,1)$.

by synteny results," *Nucleic Acids Research*, Vol. 34, No. 1, pp. 53–65, 2006.

[13] C. B. Yang and R. C. T. Lee, "Systolic algorithms for the longest common subsequence problem," *Journal of the Chinese Institute of Engineers*, Vol. 10, No. 6, pp. 691–699, 1987.