RNA Multiple Structural Alignment with Longest Common Subsequences*

Sergey Bereg[1] and Binhai Zhu[2]

[1] Department of Computer Science, University of Texas at Dallas
Richardson, TX 75083-0688, USA
besp@utdallas.edu

[2] Department of Computer Science, Montana State University
Bozeman, MT 59717-3880, USA
bhz@cs.montana.edu

Abstract. In this paper, we present a new model for RNA multiple sequence structural alignment based on the *longest common subsequence*. We consider both the off-line and on-line cases. For the off-line case, i.e., when the longest common subsequence is given as a linear graph with n vertices, we first present a polynomial $O(n^2)$ time algorithm to compute its maximum nested loop. We then consider a slightly different problem – the Maximum Loop Chain problem and present a factor-2 approximation which runs in $O(n^{2.5})$ time. For the on-line case, i.e., given m RNA sequences of lengths n, compute the longest common subsequence of them such that this subsequence either induces a maximum nested loop or the maximum number of matches, we present efficient algorithms using dynamic programming when m is small.

1 Introduction

In the study of noncoding RNA (*ncRNA*), it is well known that the corresponding genes are very active among genomic DNA. There are four such genes (polymers of nucleotides): A, C, G and U. Different from regular genes, ncRNAs fold directly into secondary and tertiary structures and the stability of the foldings are mainly determined by A-U, C-G and G-U bonds.

However, it is still not completely known how such a ncRNA folds into secondary and tertiary structures. One of the method is to take a multiple sequence of ncRNAs and investigate their common folding patterns or secondary structures [4, 19, 20]. In [4], it is proposed that the largest common nested linear subgraph of m given linear graphs (induced by m ncRNA sequences of length n) presents a solution for this problem. This problem is NP-complete and the authors presented an $O(\log^2 n)$ approximation for this problem [4].

In this paper, we follow the general methodology of [4]. However, we think that computing largest common nested linear subgraph cannot perfectly solve

* This research is partially supported by EPSCOR Visiting Scholar's Program and MSU Short-term Professional Development Program.

the problem in many situations. For example, if we have two ncRNA sequences: AGUU and CAGG, even though they induce the same largest common nested linear subgraph, the corresponding bonds and letters are completely different. (A letter cannot form a *bond* or *match* with a neighboring letter.)

The above idea forms the basis of our research. In this paper, we propose to use the Longest Common Subsequence (LCS) of m given ncRNA sequences as the basis to tackle this problem. We consider two general cases: off-line and on-line cases. In the off-line case, the LCS is already given and we want to find meaningful properties of such a LCS, namely, whether this LCS admits a special kind of fold. In the on-line case, we want to compute the LCS which admits certain kind of folding.

In general, the longest common subsequence of two sequences is not unique. Rick [17] developed an algorithm for finding *all* longest common subsequences. The number of longest common subsequences can be quite large. Greenberg [9] proved an exponential lower bound for the maximum number of distinct longest common subsequences of two sequences of length n. Therefore, in a lot of biological applications we believe that merely finding a longest common subsequence is not quite meaningful. In fact, finding a longest common subsequence satisfying a useful property is the goal of this paper. To the best of our knowledge, this has never done before, though there have been a lot of related work on identifying a (sub)string which is close to a set of given strings [13, 14] and close to a set of 'bad' strings and far from a set of 'good' strings [5].

In this paper, we mainly focus on three kinds of folding: maximum nested loop, maximum loop chains and maximum number of total matches. For the off-line case the problem is more of a graph theoretical one and we present both exact and approximation solutions. For the on-line case, the problem is NP-complete in general as computing LCS of multiple sequences, even without any other constraints, is NP-complete. We try to present efficient algorithms for cases when m is relatively small.

2 Preliminaries

Two characters (letters) $a, b \in \{A, C, G, U\}$ *match* or form a *bond* if $\{a, b\} = \{A, U\}$, or $\{a, b\} = \{C, G\}$, or $\{a, b\} = \{G, U\}$. Given a sequence $t = a_1 a_2 ... a_n$, $a_i \in \{A, C, G, U\}$, the corresponding linear graph $G(t)$ is defined as follows. The vertices of $G(t)$ are integers $1, 2, ..., n$ and there is an edge between i and $j (j > i + 1)$ if a_i and a_j match each other. For the obvious reason a_i cannot match a_{i+1} for $i = 1, 2, ..., n-1$. In other words, there is no edge between i and $i + 1$ for $i = 1, 2, ..., n-1$. This linear graph certainly characterizes the general folding possibilities of all the letters in t. In [7], a similar graph called *contact map graph* is also used for identifying protein structure similarity.

Given two edges e_1, e_2 in $G(t)$ and the intervals $I_1 = [a, b], I_2 = [c, d]$ spanned by them, we say e_1 intersects e_2 if exactly one of a, b lies on $[c, d]$ and vice versa. Therefore, if e_1 does not intersect e_2, then either I_1 and I_2 are disjoint or I_1 is contained in I_2 (assuming I_2 is longer). For the latter case we simply denote $I_1 \subset I_2$ with the understanding that e_1 does not intersect e_2. A set of edges $e_1, e_2, ..., e_p$ in $G(t)$ form a *nested loop with depth* p if the intervals $I_1, I_2, ..., I_p$ spanned by $e_1, e_2, ...e_p$ is properly contained in one another, i.e., $I_1 \subset I_2 \subset \cdots \subset I_p$ (Figure 1 (1)).

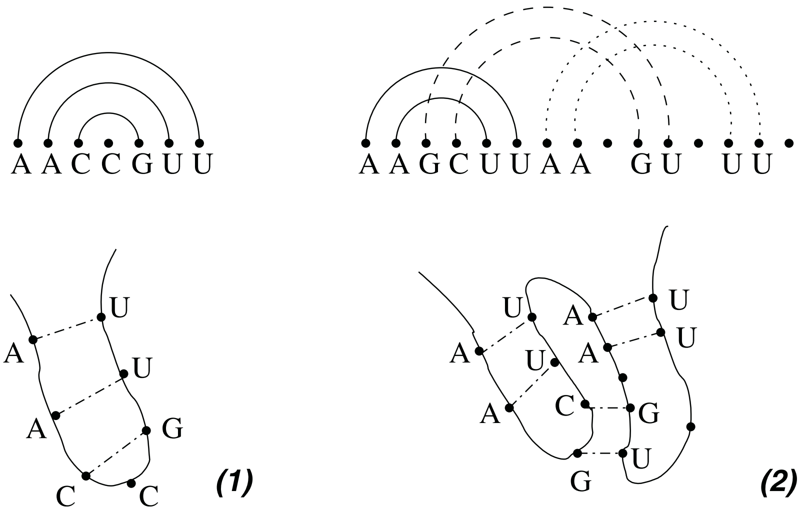

Fig. 1. An illustration of ncRNA folding with maximum loop and maximum loop chains.

Given a linear graph $G(t)$, two loops *overlap* if the edges in one loop L_1 intersect all the edges in the other loop L_2. Such an overlap is *legal* if no two edges from L_1, L_2 share the same vertex in $G(t)$. We define a *chain of loops* (or *loop chains*) as a set of loops $L_1, L_2, ..., L_w$ such that L_i overlaps with L_{i+1} but does not overlap with L_{i+x} for $i \leq w-1, x > 1$; moreover, each overlap is legal. The motivation behind this is that a chain of (relatively deep) loops provide a special kind of stable folding for a given ncRNA sequence (Figure 1 (2)).

In this paper, we propose to study several problems based on the Longest Common Subsequence (LCS). The LCS problem has been throughly studied in Hirschberg's PhD thesis [10]. Its application in computational biology dated back to 1960s [2, 3]. Some other applications of LCS in computational biology can be found in [11, 18]. Basically, for a set of m (m being a constant) sequences of length n, the corresponding LCS can be computed in $O(n^m)$ time. If m is not a constant, then the problem is NP-complete; moreover, if the alphabet is unbounded then it is difficult to find an approximation solution (in fact, as hard as approximating the Maximum Clique problem) [12].

3 The Off-Line Case: When the LCS Is Already Given

For the ncRNA multiple structural alignment problem, in general we want to compute a LCS with some additional constraints. In this section, we consider the off-line case when a LCS of some ncRNA sequences is already computed. The first problem is based on the idea that the (maximum) deepest nested loop is likely to occur in ncRNA folding (Figure 1 (1)). The second problem is based on the idea that a chain of loops is likely to fold compactly with some specified regions (Figure 1 (2)).

3.1 The Maximum Nested Loop Problem

Given a sequence (which is the LCS of some ncRNA sequences) $t = a_1 a_2 ... a_n, a_i \in \{A, C, G, U\}$, and the corresponding linear graph $G(t)$, compute the maximum or the deepest nested loop (MNL) in $G(t)$. We have the following theorem.

Theorem 1. *Given a sequence $t = a_1 a_2 ... a_n, a_i \in \{A, C, G, U\}$, and the corresponding linear graph $G(t)$, the maximum nested loop can be computed in $O(n^2)$ time.*

Proof. For $1 \leq i \leq j \leq n$, let $t_{i,j}$ denote the sequence $a_i, a_{i+1}, \ldots, a_j$. Let S be a two-dimensional array where $S[i, j]$ is the maximum depth of a nested loop in the sequence $t_{i,j}$. The values of the array S can be computed as follows. For any $1 \leq i \leq n$, $S[i, i] = 0$.

Suppose that, for $1 \leq i < j \leq n$, a_i and a_j match. Then there is a maximum nested loop of $t_{i,j}$ that contains the edge (a_i, a_j). Thus $S[i, j] = S[i+1, j-1]+1$.

Suppose that a_i and a_j do not match. Then either a_i or a_j is not an endpoint of the outermost edge of a maximum nested loop of $t_{i,j}$. Thus, $S[i, j] = \max(S[i+1, j], S[i, j-1])$. We summarize all the cases in pseudo-code (Algorithm MNL).

The depth of maximum nested loop in the input sequence is $S[1, n]$. In order to compute the nested loop we store auxiliary arrays A and B such that $(A[i, j], B[i, j])$ is the outermost edge of a maximum nested loop of $t_{i,j}$. The values $A[i, j]$ and $B[i, j]$ can be updated at the time when $S[i, j]$ is updated.

The algorithm clearly takes $O(n^2)$ in the worst case. □

A slightly different $O(n^3)$ time result on loop matching in programming language research was known long time ago [16]. That result has been used in RNA folding [4]. Although the Maximum Nested Loop problem is slightly more easier to solve, in biology it could be a very important subroutine. In ncRNAs, A-U, C-G and G-U bonds almost always occur in a nested fashion [4]; so finding such maximum nested loop is very meaningful, at least it will allow biologists to try different alternatives in folding.

3.2 The Maximum Loop Chain Problem

In this subsection we investigate a slightly different problem. When a ncRNA sequence (with its corresponding linear graph) and a set of nested loops are given, we have the following problem of computing the *maximum loop chain.*

```
Algorithm MNL
for $l = 1$ to $n$
  for $i = 1$ to $n - l + 1$
    $j = i + l - 1$
    if $l = 1$ then
      $S[l, l] = 0$
    elseif $a_i$ and $a_j$ match then
      $S[i, j] = S[i + 1, j - 1] + 1$
      $A[i, j] = i$
      $B[i, j] = j$
    elseif $S[i + 1, j] > S[i, j - 1]$ then
      $S[i, j] = S[i + 1, j]$
      $A[i, j] = A[i + 1, j]$
      $B[i, j] = B[i + 1, j]$
    else
      $S[i, j] = S[i, j - 1]$
      $A[i, j] = A[i, j - 1]$
      $B[i, j] = B[i, j - 1]$
```

The **Maximum Loop Chain** problem: Given a ncRNA sequence $t = a_1 a_2 \ldots a_n$, $a_i \in \{A, C, G, U\}$, the corresponding linear graph $G(t)$ and a set of nested loops in $G(t)$, compute a loop chain out of these loops such that its size is maximized.

The *size* of a loop chain is the sum of depths of those loops in it. The motivation of this problem is that between two overlapping loops some edges might share the same vertices (i.e., *illegal*), which violates rules in ncRNA bonding (Figure 2). In the case of Figure 2, there is one node (corresponding to U) in the

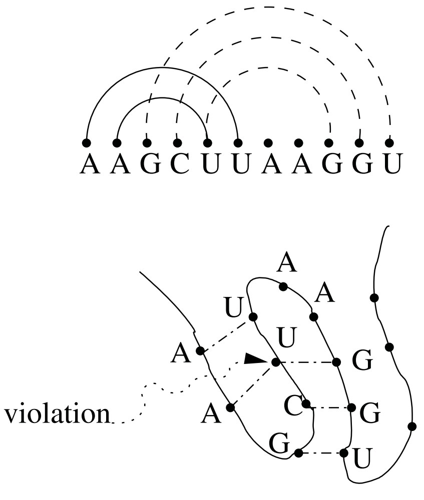

Fig. 2. A violation in ncRNA folding with two loop chains.

corresponding linear graph which is shared by two overlapping loops. Among the edges $(A, U), (U, G)$ we can only pick up one of them. So the problem is a matter of identifying the right loops and eliminating illegal edges in them.

It is not known yet whether this problem is NP-complete. We have the following approximation results.

Theorem 2. *The Maximum Loop Chain problem can be approximated in the following sense: in $O(n^{2.5})$ time a loop chain can be computed whose size is at least $\frac{1}{2}$ of the corresponding optimum.*

Given a set of nested loops we can construct a graph G' as follows: each node v_i corresponds to a nested loop L_i, there is an edge e_{ij} between two nodes v_i, v_j if their corresponding loops L_i, L_j overlap. Given two nested loops L_i, L_j with depths $|L_i|, |L_j|$ respectively, let $|L_i \cap L_j|$ be the number of nodes (letters) shared by edges in L_i, L_j. The weight of e_{ij} can be defined as $|L_i| + |L_j| - |L_i \cap L_j|$.

Then, the Maximum Loop Chain problem is to compute a path in G' such that between two neighboring nodes v_i and v_j, we need to throw away at most $|L_i \cap L_j|$ edges in either L_i or L_j (to obtain two new, smaller loops) such that eventually no two neighboring loops overlap illegally and the total number of edges in the loop chain is maximized. Notice that the weights on the edges in the final path corresponding to the maximum loop chain might not be the same as those initially in G'. It is very much a longest (heaviest) path problem with varying edge weights. We believe that this problem is NP-complete.

We will use the maximum weighted matching in G' to obtain an approximation solution for this problem. We claim that this provides at least $\frac{1}{2}$ number of bonds in the optimal loop chain. First assume that G' is also laid out from left to right on a line $\mathcal{L}$: the vertex in G' which corresponds to a loop touching the leftmost vertex in $G(t)$ is the leftmost on $\mathcal{L}$ and a tie is broken arbitrarily. The optimal loop chain then corresponds to a path with length k in G': $< v_{11}, v_{12}, ..., v_{1k} >$. If we take edges along this path at odd (even) positions (these two subsets of edges are disjoint), one subset of them contains at least half of the total bonds (edges in $G(t)$) in the optimal solution. Certainly the solution from the maximum weighted matching in G' is at least half of the total bonds (edges in $G(t)$) in the optimal solution for maximum loop chain.

Given a weighted graph with $|V|$ vertices and $|E|$ edges, the maximum weighted matching can be constructed in $O(|V|^3)$ time by Gabow [8], this was further improved to $O(\sqrt{|V|}|E|)$ time by Micali and Vazirani [15]. In our problem, G' clearly has $O(n)$ vertices and $O(n^2)$ edges as we are only interested in maximal nested loops. So the running time for computing the maximum weighted matching is $O(n^{2.5})$.

4 The On-Line Case: When the LCS Is Not Given

In this section, we study the problem when the LCS of a set of m ncRNA sequences are not given in advance. We study two versions of this general problem: LCSMNL and LCSBM. As computing LCS for multiple sequences is in general

NP-complete, both of these problems are NP-complete. We are interested in efficient solutions when m is relatively small. Recall that there might be too many LCS's for some given sequences, our goal is to identify a LCS with some useful property.

4.1 LCSMNL

We first consider the problem of computing the *longest common subsequence with maximum nested loop* (LCSMNL).

LCSMNL Problem. Given a set S of strings $s_1, s_2, \ldots, s_m$, each of length n, where $s_i = a_{i1}a_{i2}\ldots a_{in}$ and $a_{ij} \in \{A, C, G, U\}$, compute the longest common subsequence $s = b_1b_2\ldots b_K$ of $s_1, s_2, ..., s_m$ such that the maximum nested loop induced by s is at least L.

Theorem 3. *The* **LCSMNL** *problem can be solved in $O(n^{m+2})$ time. When $m = 2$, the problem can be solved in $O(n^4)$ time.*

Proof. We show the algorithm for $m = 2$ only. It is straightforward to extend it to general $m \geq 2$. We use a simplified notation $s_1 = a_1, a_2, \ldots, a_n$ and $s_2 = b_1, b_2, \ldots, b_n$. Let $a_{i,j}$, resp. $b_{i,j}$, denote the substring $a_i, a_{i+1}, \ldots, a_j$, resp. $b_i, b_{i+1}, \ldots, b_j$. We store four four-dimensional arrays L, D, A, B defined as follows. For $1 \leq i \leq j \leq n$ and $1 \leq k \leq l \leq n$, let $\Xi(i, j, k, l)$ denote the set of all longest common subsequences of $a_{i,j}$ and $b_{k,l}$. Then

- $L[i, j, k, l]$ is the length of the longest common subsequence of $a_{i,j}$ and $b_{k,l}$,
- $D[i, j, k, l]$ is the depth of the maximum nested loop induced by a sequence $\xi \in \Xi(i, j, k, l)$,
- $(A[i, j, k, l], B[i, j, k, l])$ is an outermost edge of a maximum nested loop induced by a sequence $\xi \in \Xi(i, j, k, l)$.

The items of the array $L[]$ can be computed in the same way as the computation of longest common subsequences. The value of $D[i, j, k, l]$ can be computed as in the pseudo-code shown in Algorithm LCSMNL. The theorem follows. □

4.2 LCSBM

Finally, we study the problem of computing a LCS which induces the maximum number of total matches, or *longest common subsequence with bounded matches* (LCSBM).

LCSBM Problem. Given a set S of strings $s_1, s_2, \ldots, s_m$, each of length n, where $s_i = a_{i1}a_{i2}\ldots a_{in}$ and $a_{ij} \in \{A, C, G, U\}$, compute the longest common subsequence $s = b_1b_2\ldots b_K$ of $s_1, s_2, ..., s_m$ such that the total number of matches among non-adjacent b_i and b_j is at least Q.

We assume that $m = 2$, and the algorithm can be easily extended to $m \geq 3$. We use a simplified notation $s_1 = a_1, a_2, \ldots, a_n$ and $s_2 = b_1, b_2, \ldots, b_n$. Let $b : \{A, C, G, U\} \rightarrow \{A, C, G, U\}$ be the mapping between a character and another one such that they form a bond. Then, $b(A) = U, b(C) = G, b(G) = U$ and vice versa. Adding another character $x \in \{A, C, G, U\}$ to the end of a string, x induces a number of matches to its non-adjacent characters following the above setting.

```
Algorithm LCSMNL
// Initialize $L[..]$
for $i = 1$ to $n$
  for $j = i - 1$ to $n$
    for $k = 1$ to $n$
      $L[i, j, k, k - 1] = 0$
      $L[k, k - 1, i, j] = 0$
// Compute $L[..]$
for $i = 1$ to $n$
  for $j = i$ to $n$
    for $k = 1$ to $n$
      for $l = k$ to $n$
        if $a_j = b_k$ then
          $L[i, j, k, l] = L[i, j - 1, k, l - 1] + 1$
        else $L[i, j, k, l] = \max(L[i, j - 1, k, l], L[i, j, k, l - 1])$
// Compute $D[..]$
for $l_1 = 1$ to $n$
  for $i = 1$ to $n - l_1 + 1$
    $j = i + l_1 - 1$
    for $l_2 = 1$ to $n$
      for $k = 1$ to $n$
        $l = k + l_2 - 1$
        if $j = i$ or $k = l$ then
          $D[i, j, k, l] = 0$
        elseif $a_i \neq b_k$ then
          $D[i, j, k, l] = \max(D[i + 1, j, k, l], D[i, j, k + 1, l])$
        elseif $a_j \neq b_l$ then
          $D[i, j, k, l] = \max(D[i, j - 1, k, l], D[i, j, k, l - 1])$
        elseif $a_i$ matches $a_j$ then
          $D[i, j, k, l] = D[i + 1, j - 1, k + 1, l - 1] + 1$
        else
          $D[i, j, k, l] = D[i + 1, j - 1, k + 1, l - 1]$
```

Let $LCS[i, j]$ be the length of the longest common subsequence of the sequences $s_1 = a_1 a_2 \ldots a_i$ and $s_2 = b_1 b_2 \ldots b_j$. The array $LCS[i, j]$ can be computed in $O(n^2)$ time [1].

Let a, c, g and u be integers and x be any letter from $\{A, C, G, U\}$. A sequence s is called (a, c, g, u, x)-*sequence* if s contains a letters A, c letters C, g letters G, u letters U and the last letter of s is x.

We use 6-dimensional array $M[n, n, n, n, n, 4]$ whose elements are defined as follows. Let $1 \leq i, j, k \leq n$ and $x \in \{A, C, G, U\}$ and $l = LCS[i, j]$. Let

$0 \leq a, c, g \leq l$ be any integers such that $u = l - a - c - g \in [0, l]$. The value $M[i, j, a, c, g, x]$ is -1 if there is there is no (a, c, g, u, x)-sequence s of length l that is the common subsequence of $a_1, a_2, \ldots, a_i$ and $b_1, b_2, \ldots, b_j$ such that the last letter of s is x. If such a subsequence exists, then $M[i, j, a, c, g, x]$ is the maximum number of matches of s. The pseudo-code is listed as Algorithm LCSBM.

Algorithm LCSBM

```
Initialize $M[i, j, i_1, j_1, k_1, x]$ to -1 if $i \geq 1$ or $j \geq 1$ and to 0 if $i = 0$ or $j = 0$.
Compute $LCS[]$
for $i = 1$ to $n$
  for $j = 1$ to $n$
    $l = LCS[i, j]$
    for $a = 0$ to $l$ // $a$ is the number of $A$ in LCS
      for $c = 0$ to $l - a$ // $c$ is the number of $C$ in LCS
        for $g = 0$ to $l - a - c$ // $g$ is the number of $U$ in LCS
           $u = l - a - c$ // $u$ is the numbers of $U$ in LCS
           if $a_i = b_j$ and $a_i$ is counted at least one time in $(a, c, g, u)$ then
             Let $(a', c', g', u')$ be the the same numbers as $(a, c, g, u)$
                  with one letter $a_i$ removed
             for each $x \in \{A, C, G, U\}$
               if $M[i, j, a', c', g', x] \geq 0$ then
                 Let $z$ be the total matches induced by $a_i$
                      if added to a $(a', c', g', u', x)$-sequence
               $M[i, j, a, c, g, a_i] = \max(M[i, j, a, c, g, a_i], M[i, j, a', c', g', x] + z)$
           if $a_i \neq b_j$ then
             for each $x \in \{A, C, G, U\}$
               $M[i, j, a, c, g, x] = \max(M[i - 1, j, a, c, g, x], M[i, j - 1, a, c, g, x])$
```

Theorem 4. *The* **LCSBM** *problem can be solved in $O(n^{m+3})$ time. When $m = 2$, the problem can be solved in $O(n^5)$ time.*

5 Concluding Remarks

In this paper, we study several versions of the problem for RNA multiple structural alignment using a LCS model. There are several interesting open problems related to this work: (1) When a sequence t (say, the LCS of m RNA sequences) and a set of nested loops from the inducing linear graph are given, is the problem of computing Maximum Loop Chain NP-complete? (2) In [4], given a multiple number of linear graphs, each with n vertices, computing the maximum common non-intersecting subgraph was shown to be NP-complete. But the $O(\log^2 n)$ approximation factor is too high to make the result practically meaningful. Can it be further reduced?

References

1. T. Cormen, C. Leiserson, R. Rivest, C. Stein. *Introduction to Algorithms*, second edition, MIT Press, 2001.
2. M. Dayhoff. Computer aids to protein sequence determination. *J. Theoret. Biology,* **8(1)**:97-112, 1965.
3. M. Dayhoff. Computer analysis of protein evolution. *Scientific American*, **221(1)**:86-95, 1969.
4. E. Davydov and S. Batzoglu. A computational model for RNA multiple structural alignment. *Proc. 15th Ann. Symp. Combinatorial Pattern Matching,* LNCS 3109, pp. 254-269, 2004.
5. X. Deng, G. Li, Z. Li, B. Ma and L. Wang. A PTAS for distinguishing (sub)string selection. *Proc. ICALP'02*, pp. 740-751, 2002.
6. S.R. Eddy. Computational genomics of noncoding RNA genes. *Cell*, **109:**137-140, 2002.
7. D. Goldman, S. Istrail and C. Papadimitriou. Algorithmic aspects of protein structure similarity. *Proc. 40th Ann. Symp. Foundations of Computer Science (FOCS'99)*, pp. 512-522, 1999.
8. H. Gabow. An efficient implementation of Edmond's algorithm for maximum matching on graphs. *J. ACM*, **23(2):**221-234, 1976.
9. R. I. Greenberg. Bounds on the Number of the Longest Common Subsequence Problem. *CoRR cs.DM/0301030*, 2003.
10. D. Hirschberg. The longest common subsequence problem. *PhD Thesis*, Princeton University, 1975.
11. W.J. Hsu and M.W. Du. Computing a longest common subsequence for a set of strings. *BIT*, **24:**45-59, 1984.
12. T. Jiang and M. Li. On the approximation of shortest common supersequences and longest common subsequences. *SIAM J. Comput.*, **24(5)**:1122-1139, 1995.
13. K. Lanctot, M. Li, B. Ma, S. Wang and L. Zhang. Distinguishing string selection problems. *Proc. 6th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pp. 633-642, 1999.
14. M. Li, B. Ma and L. Wang. Finding similar regions in many strings. *Proc. 31st ACM Symp. on Theory of Computing (STOC'99)*, pp. 473-482, 1999.
15. S. Micali and V. Vazirani. An $O(\sqrt{|V|}|E|^2)$ algorithm for finding maximum matching in general graphs. *Proc. 21st Ann. Symp. Foundations of Computer Science (FOCS'80)*, pp. 17-27, 1980.
16. R. Nussinov, G. Pieczenik, J. Griggs and D. Kleitman. Algorithms for loop matching. *SIAM J. Applied Math.*, **35:**68-82, 1978.
17. C. Rick. Efficient Computation of All Longest Common Subsequences. *Proc. 7th Scandinavian Workshop on Algorithm Theory (SWAT'00)*, pp. 407-418, 2000.
18. T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *J. Molecular Biology*, **147:**195-197, 1981.
19. M. Zucker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, **9:**133-148, 1981.
20. M. Zucker. Computer prediction of RNA structure. *Methods in Enzymology*, **180:**262-288, 1989.