國立交通大學

生物資訊所

碩士論文

正規化表示式的限制型多重序列比對之研究

On the Study of Multiple Sequence Alignment with Regular Expression Constraints

研究生：李威勳

指導教授：盧錦隆　教授

中華民國九十六年六月

正規化表示式的限制型多重序列比對之研究

On the Study of Multiple Sequence Alignment with Regular Expression Constraints

研 究 生：李威勳　　Student：Wei-Hsun Lee

指導教授：盧錦隆 教授　　Advisor：Prof. Chin Lung Lu

國 立 交 通 大 學

生 物 資 訊 所

碩 士 論 文

A Thesis Submitted to Institute of Bioinformatics

College of Biological Science and Technology

National Chiao Tung University in partial Fulfillment of the Requirements

for the Degree of Master in

Biological Science and Technology

June 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年六月

中文摘要

在生物資訊及計算生物領域中，多重序列比對 (Multiple Sequences Alignment) 在發掘基因體或蛋白質序列的生物意義上是很有用的工具。通常生物學家對序列的結構／功能／演化關係已有一些初步的認識，如活化部位的殘基、分子間的雙硫鍵、受質結合的部位、酵素的活化性及保守性的 Motifs 等等。因此在做多重序列比對時，生物學家希望有一個工具能讓一些結構性的／功能性的／保留性的核甘酸或殘基可以排在一起。

2004 年我們的研究團隊已開發出一套限制型多重序列比對 (Multiple Sequence Alignment with Constraints) 的工具叫 MuSiC。至目前為止 MuSiC 已被許多生物學家證實在生物的研究上是相當有用的。然而，MuSiC 中的 Constraint 只能是允許 Mismatches 但不能允許Gap的簡單序列片段。很多生物重要的 Patterns 像是 PROSITE database 中的 Motifs 在 MuSiC 中是無法使用的。因此，在此論文中我們主要的目的為研究並開發出一套能夠使用正規表示式的限制型多重序列比對 (Multiple Sequence Alignment with Regular Expression Constraints) 的演算法與工具。

我們採用了 Progressive 的方法來解決正規表示式的限制型多重序列比對的問題。事實上，這個方法的核心在於設計出有效率的演算法來解決正規表示式的限制型兩條序列比對問題 (Pairwise Sequence Alignment with Regular Expression Constraints Problem)。我們是將正規表示式 (Regular Expression) 轉成有限狀態機 (Finite Automaton)，並使用 Dynamic Programming 與 Divide-and-Conquer 方法來設計一個在時間與空間上都有效率的演算法來求得最佳化的正規表示式限制型兩條序列比對。然後，我們再跟據此演算法發展出能夠使用多個正規表示式的限制型多重序列比對工具：**RE-MuSiC** (**Mu**ltiple **S**equence Al**i**gnment with **R**egular **E**xpression **C**onstraints)，其網址在

http://140.113.239.131/RE-MUSIC

ABSTRACT

Multiple sequence alignment (MSA) has received much attention in the fields of bioinformatics and computational biology because it is very useful for discovering the biological meanings of sequences. Usually, biologists may have advanced knowledge about the structural, functional, and /or evolutionary relationships about sequences of their interest, such as active site residues, intramolecular disulfide bonds, substrate binding sites, enzyme activities, conserved motifs (consensuses) and so on. They would expect an MSA program that is able to align these sequences such that the structural, functional, and/or conserved bases (i.e., nucleotides or residues) are aligned together.

In 2004, our research group has already developed a tool, called MuSiC, for multiple sequence alignment with constraint. Since then, it has been proven by many biologists to be useful in biological research. Nevertheless, the constraints allowed in MuSiC can only be simple substrings allowing mismatches but disallowing gaps in the occurrences. Many biologically important patterns such as motifs in the PROSITE database cannot be supported by MuSiC, either. Hence, in this thesis, we study and develop an algorithm and a tool for the problem of multiple sequence alignment with regular expression constraints (RECMSA).

We used a progressive approach to design an efficient program for solving the RECMSA problem. The kernel of this approach is an efficient algorithm for solving the problem of pairwise sequence alignment with regular expression constraints (RECPSA). We transform the regular expressions into a finite automaton and then use dynamic programming and divide-and-conquer approaches to develop a time and space efficient algorithm for optimally solving the RECPSA problem, which can be implemented effectively on a desktop PC with limited memory. Using this algorithm as the kernel, we developed a web-server called **RE-MuSiC** (**Mu**ltiple **S**equence Al**i**gnment with **R**egular **E**xpression **C**onstraints) that is available on-line at http://140.113.239.131/RE-MUSIC.

誌謝

交通大學兩年忙碌的碩士生涯終於要畫下休止符了，這一路走來跌跌撞撞，嘗盡酸甜苦辣，也因此讓我在學術研究、做人處事、產業資訊各方面能有很大的收穫與成長。我這 EE 背景的人進入 CS 領域，並能順利完成碩士學位且找到好工作，實在是因為威勳幸運，一路上有很多貴人的扶持與砥礪。

首先要感謝的是我最親愛的老爸、老媽、大哥和小妹，給予我幸福的家庭與成長環境，提供我充足經濟與精神支持，讓我可以全心全力的求學而無後顧之憂，還有每次都準備滿滿愛心的水果讓我帶回新竹。

很感謝我的指導老師盧錦隆教授，在論文的研究方向、內容、英文寫作與如何做研究上給予我相當多的指導，並且讓我有機會與清大資工所鍾允昇學長合作。很謝謝允昇學長在演算法、邏輯思考上給予我很大的幫助與啟發，無數次討論和耐心溝通，讓我在這一年多來有更多的能量去克服種種難關。盧老師與允昇學長認真、負責、嚴謹的態度，也讓 RE-MuSiC 順利發表於 NAR，與你們合作是我的榮幸。感謝口試委員李家同、唐傳義、邱顯泰教授，從你們身上，不管在做人處事、冷靜思考、生物知識上都讓我獲益匪淺。

感謝這兩年帶給我最多歡樂與成長的生物資訊演算法實驗室，讓我在生物資訊所生存下來。謝謝黃彥菱學姐熱心與我分享多年程式寫作經驗與聰明工作方法；謝謝吳家榮同學與我討論程式邏輯和讓我在很快時間就變為在地新竹人；謝謝林光倫同學帶我寫第一支 C 程式與第一次打棒球；感謝學弟張演富在 Linux 與RE- MuSiC 的 PHP 程式上給我的協助，還有實驗室的機器管理。感謝學弟張文勇與姜禮瑋提供研究建議與卡丁車陪伴我研究生活。另外感謝隔壁實驗室的賴彥龍與顏士中同學，謝謝你們程式與生物上提供我諮詢，也常帶來我們實驗室許多歡笑的種子。

此外，感謝我虎尾科技大學指導老師徐超明教授，給我資訊人應有的基本認知，並適時給予我鼓勵與祝福。感謝財金所王淑芬教授，電信所王蒞君、陳富強

教授，資工所曹孝櫟教授，清大資工所劉龍龍教授的授課；讓我在忙碌的研究生涯中亦能接觸當前熱門產業的脈動和了解未來就業後應有的態度，這也對我在找工作時有相當大的幫助。感謝交通大學與清華大學豐沛的學術資源和產業資訊，讓我可以站在巨人肩上看的更遠、更清楚。

還要感謝的是我的女朋友，廖佳馨，這兩年來陪伴著我渡過每個歡笑與低潮，更謝謝妳週末常常從台北來新竹陪我，給予我最大的動力與體諒。回憶當時，還在中山電機所與交大生資所天人交戰，如今我已將穿碩士服畢業了！再次感謝所有幫助過我的貴人，我會永遠記得這兩年的一切的點點滴滴。

最後，僅以最誠摯的感謝與祝福，給予威勳的所有家人、師長、實驗室的同學、朋友們。並將取得碩士學位的喜悅與大家分享。

李威勳

2007/06　於新竹 交通大學

Contents

List of Figures

Chapter 1
Introduction

Multiple sequence alignment (MSA) is one of the fundamental problems in bioinformatics and computational biology that have been studied extensively, because it is a useful tool in the phylogenetic analyses among various organisms, identification of conserved motifs and domains in a group of related proteins, secondary and tertiary structure prediction of a protein/RNA and so on [8, 9, 19, 36, 37]. The sum-of-pairs score is widely used for selecting an optimal MSA. This kind of MSA problem, called sum-of-pairs MSA (SPMSA) problem, can be solved by extending the dynamic programming algorithm of Needleman and Wunsch for aligning two sequence [38]. In the worst case, it needs to take $O(2^k n^k)$ time to align k sequences of length n. This exponential time limits the dynamic programming technique to align only a small number of short sequences. Actually, the SPMSA problem has been shown to be NP-complete [7, 53], which means that it seems to be impossible to design an efficient algorithm to find the mathematically optimal alignment. Hence, some approximate and heuristic methods are introduced to overcome this problem. For the approximate methods, Gusfield [18] first proposed a polynomial time approximation algorithm with performance ratio of $2-\frac{2}{k}$. Then Pevzner [40] improved the

performance ratio to $2-\frac{3}{k}$. Recently, Bafna, Lawler and Pevzner [6] further improved the performance ratio to $2-\frac{l}{k}$ for any fixed *l*. It is worth mentioning that Li, Ma and Wang [26] have given a polynomial time approximation scheme for finding a multiple sequence alignment within a constant band, which is often useful in many practical cases. For the heuristic methods, the most widely used heuristic methods are so-called progressive strategies [13, 17, 21, 49, 50].

Standard multiple sequence alignment is based solely on the information about the residues/nucleotides constituting the sequences. In addition to merely the residues/nucleotides, however, biologists often possess more knowledge regarding function, structure or conserved patterns of the sequences to be analyzed. It is generally desirable to have such information incorporated into an alignment procedure, so that the alignment result can be more biologically meaningful. For example, functionally important sites are generally expected to be aligned together, but a typical alignment tool often fails to achieve this if the sequence similarity is low. Imposing constraints representing such information turns out to be an effective manner to incorporate biological knowledge into an alignment tool.

Motivated by such demand, Tang *et al*. [48] formulated the constrained multiple sequence alignment problem, where each constraint is a single residue/nucleotide. They considered alignment of RNase sequences, which are known to have a sequence of conserved residues His (H), Lys (K), and His. Using H, K, H as constraints, in the resulting constrained alignment each of these three residues can be found aligned together in a column of the alignment, appearing in the order as specified. Chin *et al*. [11] then proposed an improved algorithm for pairwise alignment and an approximation algorithm for multiple alignment. It is also noted that there have been other formulations regarding alignment with constraints proposed from different

perspectives with various approaches [15, 29-35, 44, 45, 51].

Conserved sites of a protein/RNA/DNA family are often of several residues/nucleotides long. For these patterns, the original formulation in [48] is not expressive enough. In addition, such patterns may not appear in the exact form in general. Consequently, Tsai *et al*. [52] proposed a generalized formulation and algorithm, where each constraint is a (usually short) string pattern allowing mismatches. Lu and Huang [28] then proposed a space efficient algorithm for this formulation. Web-based systems, MuSiC [52] (available at http://genome.life.nctu.edu.tw/MUSIC) and MuSiC-ME [28] (available at http://genome.life.nctu.edu.tw/MUSICME), were also developed; from now on these two systems will be referred to as MuSiC jointly. With the aid of MuSiC, Tsai *et al.* [52] and Lu and Huang [28] successfully identified a fragment in the 3' untranslated region (3'-UTR) of a SARS (severe acute respiratory syndrome) coronavirus sequence that can fold into a pseudoknot, which is potentially responsible for self-replication of the virus. Indeed, since its release, MuSiC has been found useful in, e.g., detection of functionally and/or structurally important residues/motifs in sequences [10, 48], prediction of RNA pseudoknotted structures [23, 41, 52], prediction of protein structures [16], and so on.

There are, however, formulations of many biologically significant patterns beyond the capability of MuSiC. For example, many function-related protein sites as those collected in the PROSITE database [24] are expressed in regular expressions, which cannot be modeled using the substring-with-mismatch formulation of constraints implemented in MuSiC. An example of regular expression patterns is the EGF-like domain signature 2 (EGF_2, PS01186 in PROSITE): C-x-C-x(2)-[GP]-[FYW]-x(4,8)-C, which is related to the initiation of a signal transduction that results in DNA synthesis and cell proliferation. The meaning of this

pattern is that, the first residue is Cys, followed by one residue of any kind, then a Cys, followed by two residues of any kind, then a Gly or Pro, etc. Regular expressions are also convenient in describing variable ranges between patterns or between blocks within a pattern, which is necessary for some single patterns themselves, and useful in applications where different patterns are expected to exhibit proximity in their occurrences. In the above example of EGF_2, the "x(4,8)" symbol preceding the last Cys indicates a range of length varying from 4 to 8 between a residue of [F, Y or W] (Phe, Tyr or Trp) and that last Cys.

Due to the usefulness of regular expressions in describing biological patterns, Arslan formulated the problem of Regular Expression Constrained Sequence Alignment (RECSA for short) [1]. A feasible solution of RECSA is an alignment containing a run of contiguous columns such that both of the two substrings corresponding to these columns match the regular expression. The following example, constructed by Arslan, clearly indicates the difference between RECSA and a standard unconstrained alignment:

```
T G F P S V G K T K D D - - - - A     T - - - G F P S V G K T K D D A
|   |   | |       | | |         |     |                           |
T - F - S V A - - K D D D G K S A     T F S V A K D D D G K S - - - A
                                      * * * * * * * * *
```

where a match of identical symbols is scored 1 and all other cases are scored 0. The alignment shown left is an optimal alignment without constraint, while that shown in the right is an optimal constrained alignment. The constraint *R* is [GA]-x-x-x-G-K-[ST], the P-loop motif. The starred columns "support" the satisfaction of constraint *R*: both GFPSVGKT and AKDDDGKS match *R*. Later, Chung *et al*. [12] proposed more time and space efficient algorithms for this problem, which, unlike Arslan's algorithm, is capable of reconstructing the optimal alignment.

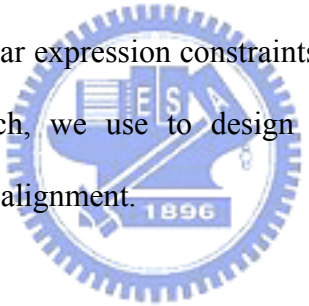
The above mentioned formulation of RECSA is for pairwise alignment with single regular expression constraints. In practice, however, it is more useful to be able to support multiple alignment with multiple constraints. In [2] Arslan extended the algorithm in [1] to support multiple alignment with multiple constraints. The algorithm proposed in [2] computes mathematically optimal constrained alignments. Unfortunately, the time complexity is extremely high, involving an exponential multiplicative factor in addition to the exponential time complexity for optimal (unconstrained) MSA computations. Even for pairwise alignment with multiple constraints, its worst case time and space requirements are intensive. In addition, the algorithms in [2] cannot find in the resulting alignment the regions responsible for the satisfactions of the constraints, either; only the alignment score, without the alignment itself, is reported. But being able to report alignments is important for practical purposes. It is therefore necessary to propose a solution more suitable for practical applications.

In this thesis, we extend the algorithm in [12] to support multiple constraints and multiple sequences. The resulting algorithm is more efficient than the one in [2] for pairwise alignment with multiple constraints. To deal with multiple sequences, a progressive method is implemented, using our improved pairwise algorithm as the kernel. This extended algorithm turns out to be more appropriate for applications than the one in [2], and we implemented a web server, RE-MuSiC (Multiple Sequence Alignment with Regular Expression Constraints), based on this algorithm. Experiments on GST proteins and on coronaviruses with phylogenetically conserved pseudoknots demonstrate that, with additional knowledge incorporated, RE-MuSiC is able to produce meaningful alignments in which important residues or structural elements can be aligned properly, even if the similarity among input sequences is low. Such ability is also useful for prediction purposes.

The rest of this thesis is organized as follows. In Chapter 2, we give the formal definition of the RECMSA (Multiple Sequence Alignment with Regular Expression Constraints) problem we study in this thesis, and also introduce weighted finite automata. In Chapter 3, we first use the dynamic programming technique to design a time-efficient algorithm for optimally solving the RECPSA (Pairwise Sequence Alignment with Regular Expression Constraints) problem. In addition, we show how to find in the resulting alignment the regions responsible for the satisfactions of the constraints, and then reconstruct the constrained sequences alignment in a space efficient manner using the divide-and-conquer approach. Based on this algorithm, we developed a program able to support multiple constraints and multiple sequences, called RE-MuSiC, using the progressive approach. In Chapter 4, we introduce the RE-MuSiC implementation and user interface. In Chapter 5, we demonstrate the applicability of our developed programs by testing them on a data set of protein and RNA sequences. Finally, we make some conclusions in Chapter 6.

Chapter 2
Preliminaries

In this chapter, we shall first formulate the problem of multiple sequence alignment with regular expression constraints. We shall then introduce the concept and basics of weighted finite automaton we use to design an efficient algorithm for the pairwise sequence alignment with regular expression constraints. Finally, we shall describe the so-called progressive approach, we use to design a heuristic algorithm for the constrained multiple sequence alignment.



2.1 Problem Formulation

Given σ sequences S_1, S_2,. . . , S_σ over alphabet Σ and a sequence of m regular expression constraints R_1, R_2 . . . , R_m, is a regular expression, the problem of the so-called multiple sequence alignment with regular expression constraints (RECMSA) is to find an alignment with the highest possible score such that all the constraints are satisfied. An alignment $\boldsymbol{A}$ of S_1, S_2,. . . , S_σ is said to satisfy all the constraints if in $\boldsymbol{A}$ there exist m regions with the following property. Let the jth region, corresponding to the jth constraint, be composed of consecutive columns k_j, k_{j+1}, . . . , k_j'. It is required that the jth region precedes the (j+1)st region without overlapping and the substring of each S_i in the jth region matches the regular expression R_j. For example, suppose we

are given two sequences S_1=cgacgta, S_2=acgcgta, as well as two regular expression constraints R_1=a and R_2=t. Then, Figure 2.1(c) shows an optimal constraint alignment in which there are two regions (i.e., 3rd and 8th columns, respectively) whose corresponding substrings of S_1 and S_2 match R_1=a and R_2=t, respectively.

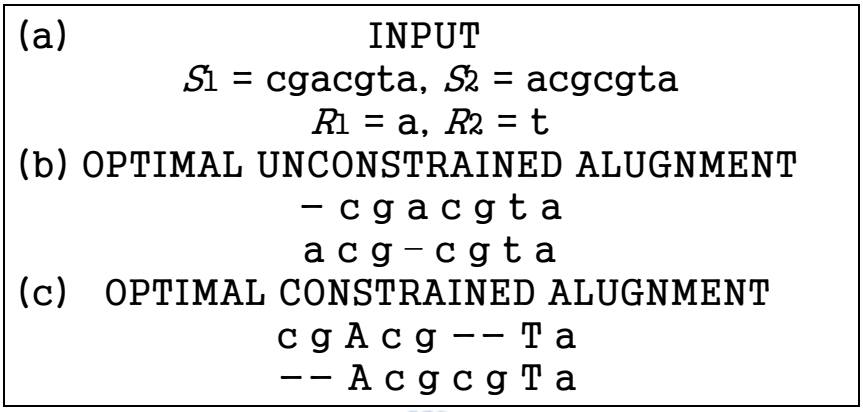


Figure 2.1: An illustration of a constrained alignment. Here a match has a score of 1, while all other cases are scored 0. Capital letters in the constrained alignment represent the columns responsible for the satisfactions of the constraints.

2.2 Constrained Alignment versus Weighted Finite Automaton

Currently, it is still a challenge to design a polynomial-time algorithm for solving the RECMSA problem, because the problem of multiple sequence alignment that can be considered a special case of RECMSA without any given constraint has been proven to be NP-hard [7, 53]. However, the pairwise version of RECMSA, simply denoted by RECPSA (Pairwise Sequence Alignment with Regular Expression Constraints), is a tractable problem, because Arslan [1] first designed a polynomial-time algorithm whose time and space complexities were further improved by Chung *et al*. [12].

The basic idea of Arslan's algorithm is to construct a weighted automaton that can be utilized to recognize any pairwise alignment of satisfying a required constraint.

For example, the automaton M in Figure 2.2(c) is able to recognize the pairwise alignment, as shown in Figure 2.2(b), which satisfies the given regular expression by the following steps. Initially, the automaton M is in the state (q_0, q_0). After reading the first two columns of $\boldsymbol{A}$ that are deletion pairs $(t, -)$ and $(c, -)$, M still stays in the state (q_0, q_0). When M reads the third column of $\boldsymbol{A}$ that is a match pair (a, a), it enters the state of (q_1, q_1) that is a final state of M. After reading the last two columns that are insertion pairs $(-, t)$ and $(-, c)$, M remains in the final state. In other words, M can enter and stay a final state from its initial state if the input pairwise alignment satisfies the given regular expression. Moreover, any non-initial state in M implies that the given regular expression is partially or completely satisfied by the input alignment. Clearly, there should be many satisfied alignments that can be accepted by M. Recall that the objective of the RECPSA problem is to find a satisfied alignment with maximum alignment score. For this purpose, a weight is assigned to each state in M for remembering the alignment that best satisfies the partial or complete constraint of regular expression. In the following, we shall formally describe how to reconstruct such a weighted automaton for recognizing the best pairwise alignment of satisfying a regular expression constraint.

Give a regular expression R, let $N = (Q, \Sigma, \delta, q_0, F)$ be an ε-free NFA (nondeterministic finite automaton) equivalent to R, which can be constructed manually or by any established algorithm [22]. We also define $\delta(Q', a)$ to be $\bigcup_{p \in Q'} \delta(p, a)$, where $Q' \subseteq Q$ and $a \in \Sigma$. In this thesis we use Q and V interchangeably. Following the notations in [23], we define a *weighted automaton* $\boldsymbol{N} \times \boldsymbol{N}$ as the finite automaton $M = (Q^M, W^M, \Sigma^M, \delta^M, q_0^M, F^M)$ which we construct as follows:

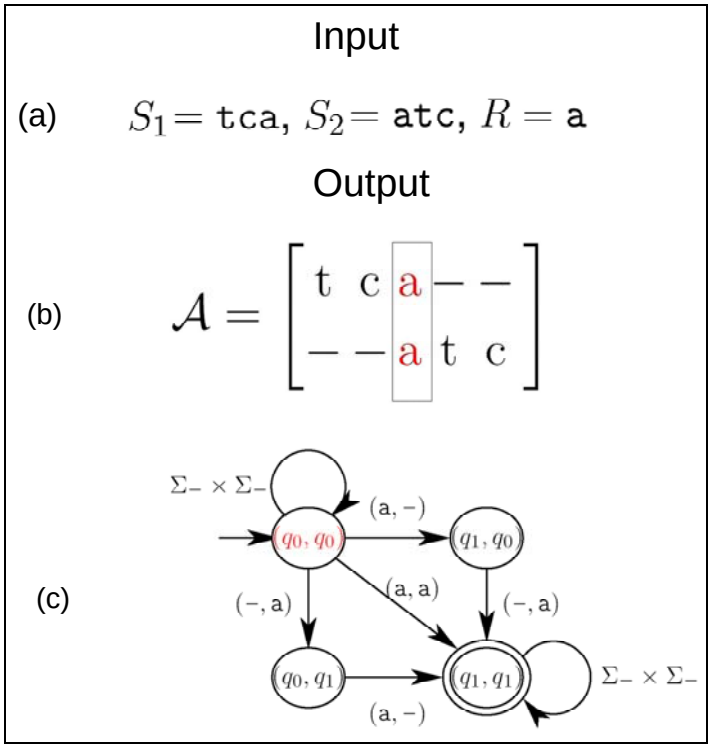


Figure 2.2: (a) Input sequences and a regular expression. (b) ***A*** is a constrained sequence alignment. (c) A simple weighted finite automaton that can accept the alignment ***A***.

- $Q^M = Q \times Q$ is the set of states. Each state of M corresponds to a pair of states in N. M remembers in each state what part of the regular expression has been seen in S_1 and S_2.
- $W^M : Q^M \rightarrow \Re$ is a function that assigns real weights to each state in Q^M, and initially all weights are -∞. We determine the active set of states of M by examining their weights. The active states of M have weights different than -∞.
- $\Sigma^M = (\Sigma \times \Sigma) - \{\varepsilon \rightarrow \varepsilon\}$. The alphabet Σ^M for M is the set of edit operations which does not include $\varepsilon \rightarrow \varepsilon$.
- $q_0^M = (q_0, q_0)$ is the start state whose initial weight is 0.
- $F^M = F \times F$ is the set of final states. If M is in a final state then M has processed an alignment that satisfies the regular expression constraint. That is, there are substrings s_1 of S_1 and s_2 of S_2 that are aligned together in an alignment, and

both s_1 and s_2 take N to final states.

– δ^M is a transition function of M that

$$\delta^M((p,q),(a,b)) = \begin{cases} \delta(p,a) \times \delta(q,b) & \text{if } (p,q) \notin F^M \cup \{(q_0,q_0)\} \\ \delta(p,a) \times \delta(q,b) \cup \{(p,q)\} & \text{otherwise.} \end{cases}$$

Function δ^M can be naturally extended to be defined on the cross product of Q^M and the set of all possible alignments. A state (p,q) of M_{i_1,i_2} is active iff there exists some alignment A of $S_1[1..i_1]$ and $S_2[1..i_2]$ such that $(p,q) \in \delta^M(q_0^M, A)$. Each state (p,q) in M_{i_1,i_2} has a score $W_{i_1,i_2}(p,q)$ assigned to it. If (p,q) is active, then $W_{i_1,i_2}(p,q)$ is the score of an optimal alignment A of $S_1[1..i_1]$ and $S_2[1..i_2]$ such that $(p,q) \in \delta^M(q_0^M, A)$. Otherwise no such alignment exists and $W_{i_1,i_2}(p, q) = -\infty$.

2.3 Progressive Multiple Sequence Alignment

The progressive approach is one of the widely used heuristics for efficiently finding a good MSA of several sequences. The ideas behind it are as follows [13, 17, 21, 49, 50].

1. Compute the distance matrix by aligning all pairs of sequences: Usually, this distance matrix is obtained by applying FASTA [27, 39] or the dynamic programming algorithm of Needleman and Wunsch [38] to each pair of sequences.
2. Construct the guide tree from the distance matrix: For the existing progressive methods, they mainly differ in the method used to build the guide tree for directing the order of alignment of sequence. To build the guide tree, for example, PILEUP (a program of GCG packages) uses UPGMA (Unweighted

Pair-Group Method using Arithmetic mean) method [46] and CLUSTAL W [50] uses NJ (Neighbor-Joining) method [43].

3. Progressively align the sequences according to the branching order in the guide tree: Initially, the closest two sequences in the tree are aligned using the normal dynamic programming algorithm. After aligning, this pair of sequences is fixed and any introduced gaps cannot be shifted later (i,e., once a gap, always a gap). Then the next two closest pre-aligned groups of sequences are joined in the same way until all sequences have been aligned. (Here, we may consider a sequence as an aligned group of a sequence.) To align two groups of the pre-aligned sequences, the score between any two positions in these two groups is usually the arithmetic average of the scores for all possible character comparisons at those positions. We call this kind of scoring methods as a set-to-set scoring.

In fact, MST has been used as a significant tool for data classification in the fields of biological data analysis. In [48], Tang *et al.* proposed a variant of progressive method by using the Kruskal MST to construct the guide tree, called Kruskal merging order tree. The Kruskal merging order tree of k sequences is constructed as follows. First, we create a complete graph $G = (V,E)$ of k sequences in a way that each vertex of V represents a sequence and each edge e of E is associated with a weight $d(e)$ to represent the distance between the corresponding sequences of its end-vertices. Then we use the Kruskal's algorithm [25] to construct the Kruskal MST of G, denoted by T .For completeness, we describe the Kruskal method for constructing T as follows.

1. Sort all edges of E in non-decreasing order according to their distances.

2. Initially, T is empty. Then we repeatedly add the edges of E in non-decreasing order to T in a way that if the currently adding edge e to T dose not create a cycle in T, then we add e to T; otherwise, we discard e.

Next, according to the Kruskal MST T, we build the Kruskal merging order tree T_K as follows.

1. Let $V = \{v_1, v_2, \ldots, v_k\}$ and $e_1, e_2, \ldots, e_{k-1}$ be the edges of T with $d(e_1) \leqq d(e_2) \leqq \ldots \leqq d(e_{k-1})$.
2. For each vertex $v_i \in V$, we create a tree T_i such that T_i contains only a node v_i. For the purpose of merging trees, we consider T_i as a rooted tree by designating v_i as its root, and define the merge of two tree T_i and T_j respectively rooted at v_i and v_j to be a new tree rooted at a new vertex u such that v_i and v_j become the children of u.
3. For each $e_k = (v_i, v_j)$, where K increases from 1 to k - 1, we find the trees T_i and T_j containing v_i and v_j respectively and then merge them into a new tree. This process is continued until the remaining is only one tree. Then this final tree is the so-called Kruskal merging order tree T_K.

The construction of G for k sequences can be done in $\boldsymbol{O}(k^2)$ time and the computation of the Kruskal's MST T of G can be done in $\boldsymbol{O}(k^2 \log k)$ time. Then the construction of T_K from T can be implemented by the disjoint set union and find algorithm proposed by Gabow and Tarjan [33] in $\boldsymbol{O}(m + k)$ time, where m denotes the number of union and find operations. It is not hard to see that $m = \boldsymbol{O}(k)$ and hence the construction of T_K takes $\boldsymbol{O}(k)$ time. Therefore, the total time complexity of constructing T_K is $\boldsymbol{O}(k^2 \log k)$.

Chapter 3
Algorithm

In this chapter, we shall propose an algorithm to solve the problem of pairwise sequence alignment with several regular expression constraints (RECPSA). Then, we shall present how to extend this algorithm for dealing with multiple sequences with several regular expression constraints (RECMSA).

3.1 Algorithm for RECPSA (Pairwise Sequence Alignment with Regular Expression Constraints)

First consider the case of pairwise alignment. For simplicity let the two input sequences have equal lengths of n. Let $A_h = (Q_h, \Sigma, \delta_h, q_h, F_h)$ be an ε-free NFA equivalent to R_h (q_h is the initial state of A_h), and $M_h = (Q^{M_h}, W^{M_h}, \Sigma^{M_h}, \delta^{M_h}, q_0^{M_h}, F^{M_h})$ be the weighted automaton corresponding to A_h, where $Q^{M_h} = Q_h \times Q_h$, $\Sigma^{M_h} = (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \setminus \{(-,-)\}$, $q_0^{M_h} = (q_h \times q_h)$, $F^{M_h} = (F_h \times F_h)$, and, for $(p,q) \in Q^{M_h}$ and $(a,b) \in \Sigma^{M_h}$,

$$\delta^{M_h}((p,q),(a,b)) = \begin{cases} \delta_h(p,a) \times \delta_h(q,b) & \text{if } (p,q) \notin F^{M_h} \cup \{q_0^{M_h}\} \\ \delta_h(p,a) \times \delta_h(q,b) \cup \{(p,q)\} & \text{otherwise} \end{cases}$$

On each pair (i_1, i_2) of indices of the sequences, denote as $M_h^{i_1,i_2}$ the corresponding weighted automaton. Also let M be an NFA obtained by "concatenating" $M_1, \ldots, M_m$

by adding an "empty move" from each final state of M_h to the initial state of M_{h+1}, $1 \leq h < m$. The initial state of M is set to be the initial state of M_1, and the final states of M are the final states of M_m. Hence M accepts all feasible constrained alignments (see Fig. 3.3 for an example). The score held in state (p, q) of $M_h^{i_1,i_2}$ (resp. M^{i_1,i_2}) is denoted as $W_h^{i_1,i_2}(p,q)$ (resp. $W^{i_1,i_2}(p,q)$). The score $W_h^{i_1,i_2}(p,q)$ represents the score of a best alignment A of $S_1[1..i_1]$ and $S_2[1..i_2]$ such that all of $R_1, \ldots, R_{h-1}$ are satisfied and that state (p, q) of M_h is reached if A is given to M_h as input. It is equal to $W^{i_1,i_2}(p,q)$, the score of optimally aligning $S_1[1..i_1]$ and $S_2[1..i_2]$ such that the alignment can lead us from the initial state of $M^{i_1,i_2}(p,q)$ to state (p,q). The goal of the algorithm is to compute $W_m^{n,n}$, since $\max_{(p,q)\in F_m \times F_m} W_m^{n,n}(p,q)$ is the optimal score of aligning S_1 and S_2 with all m regular expression constraints satisfied.

The algorithm iterates over all pairs (i_1, i_2) of indices of sequences S_1 and S_2, row by row. It computes $M_h^{i_1,i_2}$ for all $1 \leq i_1, i_2 \leq n$ and $1 \leq h \leq m$ throughout its execution. Let V_h and E_h be the set of states and set of arcs in automaton A_h, respectively. Denote as $L_h^{i_1,i_2}$ a $|V_h| \times |V_h|$ table used to hold $W_h^{i_1,i_2}$. On each sequence index pair (i_1, i_2), the algorithm computes $L_h^{i_1,i_2}, \ldots, L_m^{i_1,i_2}$, in the order. For all $(p, q) \in Q^{M_1}$, $L_1^{i_1,i_2}[p,q]$ is computed using the (first) algorithm Chung *et al.* proposed in [12]. The same is performed for $L_h^{i_1,i_2}$, $h > 1$, except that $L_h^{i_1,i_2}[q_h,q_h]$ is initialized to the maximum of $L_{h-1}^{i_1,i_2}[p, q]$ rather than $-\infty$ as the other states are, the maximum taken over all final states (p, q) of M_{h-1}.

The above procedure takes $O\left(\sum_{h=1}^{m} |E_h||V_h|n^2\right)$ time, since each $L_h^{i_1,i_2}$ takes $O(|E_h||V_h|)$ time by [12]. In what follows we present how to reconstruct the optimal

alignment in $O\left(\sum_{h=1}^{m} h|V_h|^2 n\right)$ space without affecting the time complexity, by a generalization of the method Chung *et al.* proposed in [12].

Consider an optimal alignment ***A*** satisfying all constraints. Intuitively, if we know the substrings of S_1 and S_2 responsible for ***A***'s satisfaction of the constraints, then ***A*** (or another optimal solution) can be reconstructed efficiently. Suppose that the substrings of S_1 and S_2 responsible for A's satisfying the *l*th regular expression are $S_1[b_1^l \cdots e_1^l]$ and $S_2[b_2^l \cdots e_2^l]$, $1 \le l \le m$. Then we can align $S_1[b_1^l \cdots e_1^l]$ and $S_2[b_2^l \cdots e_2^l]$ for all l = 1,…,*h*, align $S_1[1 \cdots b_1^1 - 1]$ and $S_2[1 \cdots b_2^1 - 1]$, align $S_1[e_1^l + 1 \cdots b_1^{l+1} - 1]$ and $S_2[e_2^l + 1 \cdots b_2^{l+1} - 1]$ for l = 1, 2, . . . , $m-1$, and align $S_1[e_1^m + 1 \cdots n]$ and $S_2[e_2^m + 1 \cdots n]$. These alignments can be concatenated in the proper order to obtain ***A***. Each alignment can be computed in linear space by Hirschberg's celebrated divide-and-conquer algorithm [14]. In the example of Fig.3.1, $b_1^1 = 1$, $e_1^1 = 1, b_2^1 = 1$, $e_2^1 = 1, b_1^2 = 3$, $e_1^2 = 3, b_2^2 = 2$, $e_2^2 = 2$.

Two types of $|V_h| \times |V_h|$ tables, $\beta_{l,h}^{i_1,i_2}$ and $\eta_{l,h}^{i_1,i_2}$, $1 \le l \le h$, are maintained for this purpose. Let *A* be the alignment underlying the score $W_h^{i_1,i_2}(p,q)$. Then $\beta_{l,h}^{i_1,i_2}[p,q]$ keeps the indices of S_1 and S_2 corresponding to the column immediately before the first column of leaving the initial state of M_l. If $l = h$ (resp. $l < h$), then keeps the indices of S_1 and S_2 corresponding to the first column of *A* which leads us to arrive at state (*p, q*)(resp. a final state) of M_l. It can be seen that, if (*p, q*) is the final state of $M_m^{n,n}$ with the best $W_m^{n,n}(p,q)$ value, then $\beta_{l,m}^{n,n}[p,q]$ stores $b_1^l - 1$ and $b_2^l - 1$, and $\eta_{l,m}^{n,n}[p,q]$ stores e_1^l and e_2^l, where the meanings of these *b* and *e* values are as discussed in the last paragraph.[1]

[1] When (*p, q*) is not a final state, the values of $\eta_{h,h}^{i_1,i_2}[p,q]$ are are actually not relevant (but the $\beta_{h,h}^{i_1,i_2}[p,q]$ values are still critical).

The computation of $\beta_{h,h}^{i_1,i_2}$ and $\eta_{h,h}^{i_1,i_2}$, h=1,…, m, proceeds in the same manner as in [12].[2] As to $\beta_{l,h}^{i_1,i_2}[q_h,q_h]$ and $\eta_{l,h}^{i_1,i_2}[q_h,q_h]$, $l \neq h$, we initialize them to $\beta_{l,h-1}^{i_1,i_2}[p,q]$ and $\eta_{l,h-1}^{i_1,i_2}[p,q]$, respectively, where ($p$, q) is the final state of $M_{h-1}^{i_1,i_2}$ with the best $W_{h-1}^{i_1,i_2}(p,q)$ score among all final states. In general, when $l \neq h$, for (initial or non-initial) state (p,q), each time when $L_h^{i_1,i_2}[p,q]$ is updated to, say, $L_h^{i_1',i_2'}[p',q']+\gamma(x,y)$ where $(i_1',i_2') \in \{(i_1-1,i_2-1),(i_1-1,i_2),(i_1,i_2-1)\}$ and (x,y) is the corresponding edit operation (e.g.,(x, y) = ($S_1[i_1]$,-) if $i_1' = i_1 - 1$ and $i_2' = i_2$), $\beta_{l,h}^{i_1,i_2}[p,q]$ and $\eta_{l,h}^{i_1,i_2}[p,q]$ are also updated to $\beta_{l,h}^{i_1',i_2'}[p',q']$ and $\eta_{l,h}^{i_1',i_2'}[p',q']$ respectively.

When row i_1 is being considered, all tables for rows less than i_1 are not necessary and can be discarded. Therefore the overall space requirement, including the reconstruction of the optimal alignment, is $O\left(\sum_{h=1}^{m} h|V_h|^2 n\right)$.

In [2], an algorithm extending the one in [1] to support multiple constraints and multiple sequences, is proposed. In the pairwise case, the algorithm in [2] has time complexity $O\left(\sum_{h=1}^{m}|E_h|^2 n^2\right)$ and space complexity $O\left(\sum_{h=1}^{m}|E_h|^2 n\right)$. It is clear that $|V_h| = O(|E_h|)$, hence the algorithm presented here never takes more time than the one in [2]. In the worst case, $|E_h|$ can be proportional to $|V_h|^2$, and the algorithm in [2] takes $O\left(\sum_{h=1}^{m}|V_h|^4 n^2\right)$ time, while the one presented here takes $O\left(\sum_{h=1}^{m}|V_h|^3 n^2\right)$ time. Hence the time complexity of the algorithm presented here compares favorably with

[2]If (p, q) is not a final state, then the value held in $\eta_{h,h}^{i_1,i_2}[p,q]$ may not be as described in the last paragraph. But as just mentioned, this does not affect the correctness.

the one in [2]. As to the space complexity, if $|E_h| = O(|V_h|)$, then the algorithm here may take more space than the one in [2]. However, if $\sum_{h=1}^{m} |E_h|^2 = \Omega\left(\sum_{h=1}^{m} h|V_h|\right)$, then the algorithm here is more space efficient. More importantly, the stated space complexity for the algorithm in [2] does not include the reconstruction of the alignment; only the alignment score can be computed. It is clearly an important issue for a web-server to report the alignment. If a naive backtracking method is used to augment the algorithm in [2] to reconstruct the alignment, the space requirement would be $O\left(\sum_{h=1}^{m} |E_h|^2 n^2\right)$, which is too high for practical use.

Now, we apply this algorithm to solve the following example.

```
        Input                    Scoring matrix
S1 = acgt, S2 = atcg               Match:3
     R1 = a, R2 = t              Mismatch:-1
                                    Gap=-2

     Optimal unconstrained alignment
                a - c g t
                a t c g -
     Optimal constrained alignment
                A c g T - -
                A - - T c g
```

Figure 3.1：An illustration of a constrained alignment.

	S2	a	t	c	g
S1	$M_{0,0}$	$M_{0,1}$	$M_{0,2}$	$M_{0,3}$	$M_{1,4}$
a	$M_{1,0}$	$M_{1,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$
c	$M_{2,0}$	$M_{2,1}$	$M_{2,2}$	$M_{2,3}$	$M_{2,4}$
g	$M_{3,0}$	$M_{3,1}$	$M_{3,2}$	$M_{3,3}$	$M_{3,4}$
t	$M_{4,0}$	$M_{4,1}$	$M_{4,2}$	$M_{4,3}$	$M_{4,4}$

Figure 3.2：The matrix of weighted automata.

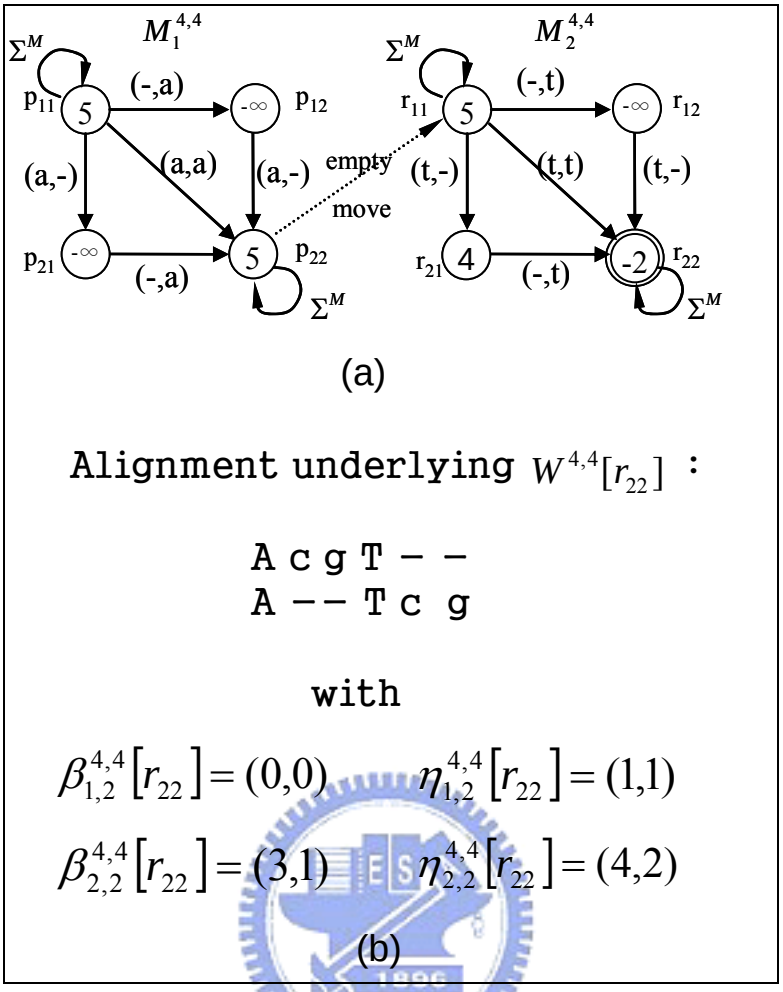


Figure 3.3: An illustration of the weighted automaton M corresponding to the example in Fig.3.1. State (p_2, p_1), etc., is denoted as p_{21}, etc., for brevity. (a) Automaton $M^{4,4}$, which is the concatenation of $M_1^{4,4}$ and $M_2^{4,4}$. Numbers in the states are the scores $W^{4,4}[p_{11}]$, etc. Initial states of $M_1^{4,4}$ and $M_2^{4,4}$ are p_{11} and r_{11}, respectively, while the final states are p_{22} and r_{22}, respectively. The initial and final states of $M^{4,4}$ are p_{11} and r_{22}, respectively. (b) The alignment underlying $W^{4,4}[r_{22}]$ is the optimal alignment of $S_1[1..4]$ and $S_2[1..4]$ such that state r_{22} is reached.

3.2 Algorithm for RECMSA (Multiple Sequence Alignment with Regular Expression Constraints)

In this section, we use Algorithm RECPSA in previous section as kernels to design an RECMSA algorithm, for progressively aligning the input sequences into a RECMSA according to the branching order of a guide tree, where the guide tree we use here is the so-called Kruskal merging order tree. We refer the reader to Section 2 for the details of constructing the Kruskal merging order tree. Except for the adopted RECPSA kernel, the execution steps, which are described as follows.

1. Compute the distance matrix D by globally aligning all pairs of sequences using Algorithm RECPSA, where $D(i, j)$ denotes the distance between sequences S_i and S_j .
2. Create a complete graph G from the distance matrix D and then compute the Kruskal merging order tree T_k from G to serve as the guide tree.
3. Progressively align the sequences according to the branching order of the guide tree T_k in a way that the currently two closest pre-aligned groups of sequences are joined by applying Algorithm RECPSA to these two groups of sequences, where the score between any two positions in these two groups is the arithmetic average of the scores for all possible character comparisons at those positions.

In the following, we analyze the time complexity of Algorithm RECPSA. It is not hard to see that step 1 costs $O(k^2n^2)$ time, where n is the maximum of the lengths of k sequences. According to the paper of Tang *et al.* [48] , step 2 can be done in $O(k^2 log\ k)$ time. In step 3, there are at most $O(k)$ iterations for calling Algorithm RECPSA, whose time complexity is $O\left(\sum_{h=1}^{m}|V_h|^3 n^2\right)$, to join two pre-aligned groups of sequences.

Hence, the time complexity of step 3 is $O\left(\sum_{h=1}^{m}|V_h|^3 kn^2\right)$. It is usually that $|k| = O\left(\sum_{h=1}^{m}|V_h|^3\right)$, hence the cost of Algorithm RECMSA is dominated by step 3 and hence its time complexity is $O\left(\sum_{h=1}^{m}|V_h|^3 kn^2\right)$.

Chapter 4
Implementation of RE-MuSiC

In this chapter, we shall introduce the implementation of RE-MuSiC, as well as its web interface, and then describe how to use it in details. Besides, we shall introduce the syntax of regular expression utilized in RE-MuSiC.

4.1 RE-MuSiC

The kernel of RE-MuSiC (short of Multiple Sequence Alignment with Regular Expression Constraints) was implemented by C and its web interface by PHP and HTML. RE-MuSiC (http://140.113.239.131/RE-MUSIC) can be easily accessed via a simple web interface (see Figure 4.1). The input of the RE-MuSiC web server consists of a set of Protein/DNA/RNA sequences and a set of user-specified regular expression constraints. The output of RE-MuSiC is a multiple sequence alignment with regular expression constraints.

4.2 Usage of RE-MuSiC

In this section, we shall describe the usage of RE-MuSiC step by step, the output of RE-MuSiC and other information including scoring matrices, gap-penalty, and syntax of regular expressions currently used in RE-MuSiC.

4.2.1 Input of RE-MuSiC

1. Input a set of genomic sequences in the FASTA format in the top blank field **(1)**.
2. Enter one or more regular expressions that are separated by spaces in the "Regular expression constraints" field **(2)**. Note that each constraint of regular expression should be put in quotes first. For example, if users have two regular expressions, say [ST]-x(2)-[DE] and G-{EDRKHPFYW}-x(2)-[STAGCN]-{P}, then they may key in the following line in the "Regular expression constraints" field:

   "[ST]-x(2)-[DE]" "G-{EDRKHPFYW}-x(2)-[STAGCN]-{P}"

   If no constraint is specified, then RE-MuSiC produces an unconstrained alignment.

3. Select the type of input sequences that can be either protein or DNA/RNA **(3)**.
4. Just click "Execute RE-MuSiC" button **(4)** if users would like to run RE-MuSiC with default parameters; otherwise, they continue with the following parameter settings.
5. Select a suitable scoring matrix for protein or DNA/RNA sequences from a list of predefined matrices **(5)**.
6. Key in two real values for gap open penalty **(6)** and gap extension penalty **(7)**, respectively, since the RE-MuSiC web server penalizes the gaps using the affine gap penalty function.
7. Check the checkbox and enter an email address **(8)** if the user would also like to receive an email that contains a hyperlink to the RE-MuSiC result from the server. Note that the RE-MuSiC result will be kept on the server only for 24 hours.
8. Click "Execute RE-MuSiC" button to run RE-MuSiC **(4)**.

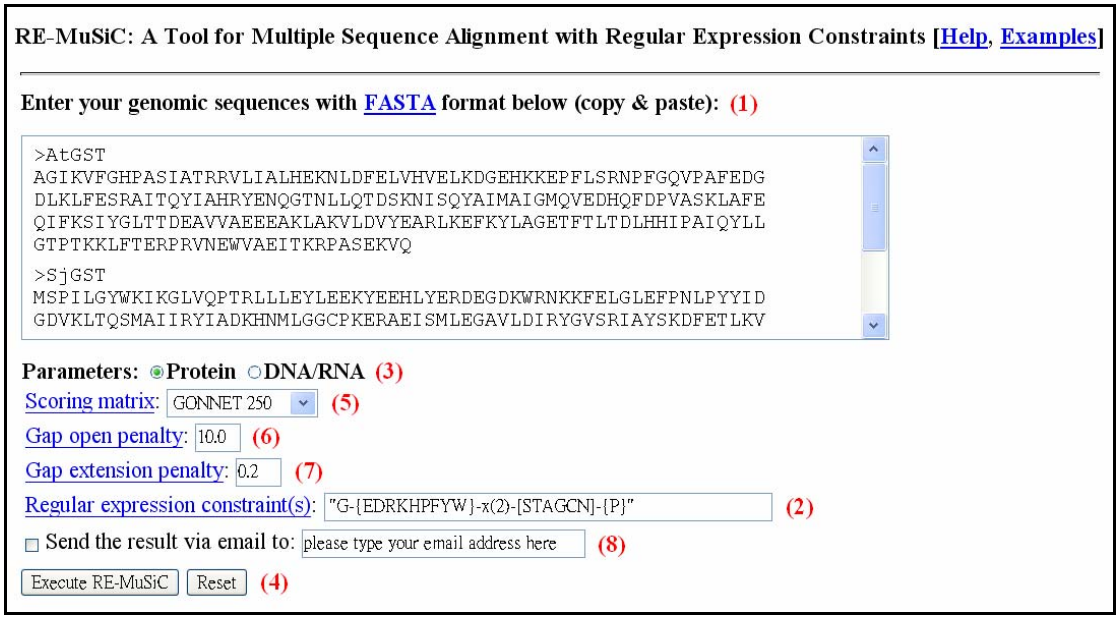


Figure 4.1: The web interface of RE-MuSiC.

4.2.2 Output of RE-MuSiC

In the first part of the output page, RE-MuSiC shows the user-specified parameters, including scoring matrix, gap open and extension penalties, regular expression constraints and so on. Next, RE-MuSiC outputs its result of the constrained sequence alignment, in which the columns whose residues/nucleotides match regular expression constraints are shaded in yellow (refer to Figures 4.2 and 4.3 for examples). On addition, RE-MuSiC allows users to download the RE-MuSiC alignments in FASTA format or ClustalW format.

The Resulting RE-MuSiC:

The user-specified parameters are:
Scoring matrix = GONNET 250
Gap open penalty = 10.0
Gap extension penalty = 0.2
Regular expression(s) = "[ST]-x-[RK]" "G-{EDRKHPFYW}-x(2)-[STAGCN]-{P}"
Number of regular expression constraint(s) = 2
Running time = 0 seconds.

Download the alignment in: [FASTA format] [ClustalW format]

```
P46850|RTCB_ECOLI       WAELEAGYQWL--------------------TQKYPRFLNTNNYKHLGTLGTGNHFIEIC 40
Q58095|Y682_METJA       WA-VKEGYGWKEDLEFIEEHGCLKDADASYVSDK----AKERGRVQLGSLGSGNHFLEVQ 55
                        ** :: ** *                      ::*     : ..  :**:**:****:*:

P46850|RTCB_ECOLI       L 41
Q58095|Y682_METJA       Y 56
```

Download the alignment in: [FASTA format] [ClustalW format]

Figure 4.2: An example of the output of RE-MuSiC for protein sequences, where the residues in the first block of columns shaded in yellow match the first regular expression of "[ST]-x-[RK]", and those in the second block of columns shaded in yellow match the second regular expression of "G-{EDRKHPFYW}-x(2)-[STAGCN]-{P}".

The Resulting RE-MuSiC:

The user-specified parameters are:
Scoring matrix = Identity
Gap open penalty = 15.0
Gap extension penalty = 6.66
Regular expression(s) = "C-U-x(4)-C-x(15,16)-U-G-x(2)-A" "G-x(4)-A-G" "U-x(3)-A"
Number of regular expression constraint(s) = 3
Running time = 2 seconds.

Download the alignment in: [FASTA format] [ClustalW format]

```
PEDV            ---GCGGUCUUGGUCUUGCACACAACGGUAAGCCAG-UGGUAAUGUCAGUGCAAGAAGGA 56
BCoV            GAUAAGGCACU---CUCUAUCAGAAUGGAUGUCUUGCUGCUA-UAAUAGAUAGAGAAGGU 56
                     **   *   **    ** ** **    *  * ** ** *   **    ******

PEDV            UAUUACCAUAGCACUGUCA 75
BCoV            UA-UAGCA---GACUA--- 68
                ** ** **    ***
```

Download the alignment in: [FASTA format] [ClustalW format]

Figure 4.3: An example of the output of RE-MuSiC for RNA sequences. Notice that a gap appears in the block of matching the 1st regular expression constraint, which is not allowed to happen in MuSiC.

4.2.3 Scoring Matrices

For protein sequences, three inbuilt series of scoring matrices are used in RE-MuSiC system: (1) GONNET 250 (default), (2) BLOSUM 30, 45, 62, and 80, and (3) PAM 30, 70, 120, 250, and 350. For DNA/RNA sequences, RE-MuSiC provides identity matrix only.

4.2.4 Gap Penalty

The RE-MuSiC web server penalizes the gaps with the so-called affine gap penalty function, which charges the score of "Gap open penalty" for the existence of a gap and the score of "Gap extension penalty" for each residue/nucleotide in the gap. The default values of "Gap open penalty" for protein and DNA/RNA sequences are 10.0 and 15.0, respectively, and those of "Gap extension penalty" are 0.2 and 6.66, respectively. All these default values can be modified by the user, depending on the evolutionary distance between the input sequences of interest.

4.2.5 Syntax of Regular Expression Used in RE-MuSiC

Regular expression is a pattern-defining notation that describes a string (or, equivalently, sequence here) or a set of strings. In RE-MuSiC, the conventions of describing a pattern of regular expression are the same as those used in PROSITE.

- The standard IUPAC one-letter codes for the amino acids and nucleotides are used in the regular expression of RE-MuSiC. Notice here that the symbol 'x'/'X' is used for a position where any amino acid or nucleotide is accepted.

IUPAC Codes for Amino Acids			
Letter	**Meaning**	**Letter**	**Meaning**
A	A (Alanine)	N	N (Asparagine)
B	D, N	P	P (Proline)
C	C (Cystine)	Q	Q (Glutamine)
D	D (Aspartic Acid)	R	R (Arginine)
E	E (Glutamic Acid)	S	S (Serine)
F	F (Phenylalanine)	T	T (Threonine)
G	G (Glycine)	V	V (Valine)
H	H (Histidine)	W	W (Tryptophan)
I	I (Isoleucine)	X	X (Unknown or Other Amino Acid)
K	K (Lysine)	Y	Y (Tyrosine)
L	L (Leucine)	Z	E, Q
M	M (Methionine)		

IUPAC Codes for Nucleotides			
Letter	**Meaning**	**Letter**	**Meaning**
A	A (Adenine)	X/N	A, C, G, T
B	C, G, T	R	A, G (Purine)
C	C (Cytosine)	S	C, G
D	A, G, T	T	T (Thymine)
G	G (Guanine)	U	U (Uracil)
H	A, C, T	V	A, C, G
K	G, T	W	A, T
M	A, C	Y	C, T (Pyrimidine)

- The amino acids (or nucleotides) that are allowed to appear at a given position are indicated by listing them in a pair of square brackets '[ ]'.
    - For example, [ALT] stands for Ala (A), Leu (L) or Thr (T).
- The amino acids (or nucleotides) that are not accepted at a given position are indicated by listing them in a pair of braces '{ }'.
    - For example, {AM} stands for any amino acid except Ala (A) and Met (M).
- Each element in a pattern of regular expression is separated from its neighbors by a dash '-'.
    - For example, [GA]-G-K-[ST] means that the first position of the pattern can be occupied by either Gly (G) or Ala (A), the second and third positions must be Gly (G) and Lys (K), respectively, and the last position can be either Ser (S) or Thr (T).
- Repetition of an element in a pattern is indicated by appending, immediately following that element, an integer or a pair of integers (meaning the allowed range of the number of repetitions) in parentheses. For example,
    - x(3) equals to x-x-x, a meaning pattern of any three amino acids (or nucleotides).
    - A(3) equals to A-A-A, a meaning pattern of three amino acids of Ala (A).
    - x(2,4) equals to x-x, x-x-x, or x-x-x-x.

For example, based on the above conventions, [AC]-x-V-x(4)-{ED} is translated as [Ala or Cys]-any-Val-any-any-any-any-{any but Glu or Asp}. A subsequence matches a given regular expression, if it can be described by this regular expression. For example, "AgVdefgB" matches the above regular expression.

Chapter 5
Experiments

In this chapter, we shall demonstrate the applicability of our RE-MuSiC by testing it on two data sets, one with protein sequences and the other with RNA sequences.

5.1 Protein Sequences with Active Site Residues

In this experiment, we analyzed three GST (Glutathione S-Transferase) proteins as follows.

1. AtGST: a phi class GST from plant *Arabidopsis thaliana* whose PDBID is 1GNW:A.
2. SjGST: an alpha class GST from non-mammalian *Schistosoma japonicum* (flat worm) whose PDBID is 1M99:A.
3. SsGST: a pi class GST from mammalian *Sus scrofa* (pig) whose PDBID is 2GSR:A.

Notice that the structural similarity between these three proteins is very high (see Figure 5.1), although their pairwise sequence identities are extremely low. In particular, the glutathione binding sites (G-sites) of these GST proteins have been found to have a conserved architecture, and the glutathione backbone conformations adopted in these GSTs from different species are quite similar [42]. Also, the cheimical natures of their residues acting as G-site ligands and interactions facilitated with glutathione exhibit analogy [42]. Due to their low sequence identity, it is hard to use a typical alignment tool, like ClustalW, to produce an accurate alignment (see

Figure 5.2 for example). In this ClustalW alignment as shown in Figure 5.3, one of the active site residues shared by these three GST proteins was not aligned well. They actually should be aligned together, however, if we superpose the crystal structures of these three GST proteins [42]. By querying PROSITE with these three GST protein sequences, we noticed that they all share a pattern of PS00006 ("[ST]-x(2)-[DE]") in common. Using this pattern as the constraint, we aligned again the three GST protein sequences mentioned above with RE-MuSiC, resulting in an alignment as shown in Figure 5.3 that indeed satisfies the requested constraint (*i.e.*, those residues matching "[ST]-x(2)-[DE]" were lined up). In addition, it is worth mentioning here that all the active site residues shared among these GSTs were also aligned together. This suggests that, with additional information regarding some common patterns, RE-MuSiC is more reliable in aligning together biologically important residues from a set of closely related proteins, even when their sequence similarities are low. Demonstrated by this experiment, therefore, our RE-MuSiC web server is helpful in the detection of active site residues in a given set of protein sequences.

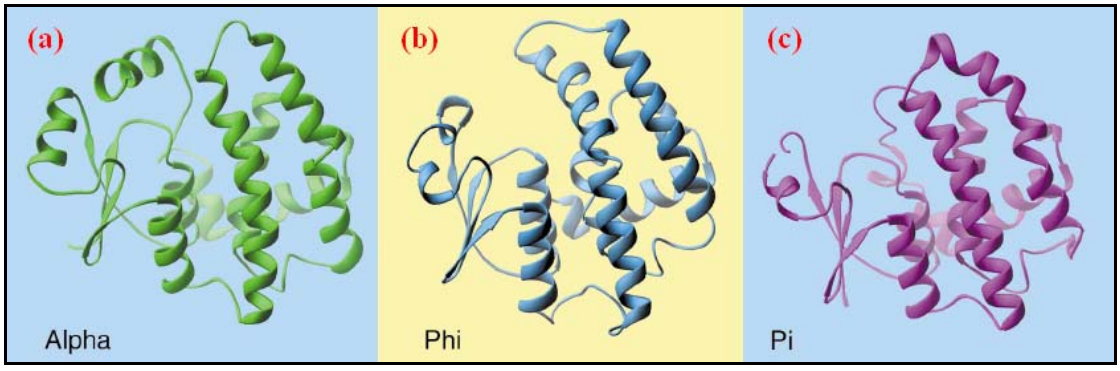


Figure 5.1: (a) Alpha class GST structure to which SjGST from non-mammalian *S. japonicum* (flat worm) belongs, (b) Phi class GST structure to which AtGST from plant *A. thaliana* belongs, (c) Pi class GST structure to which SsGST from mammalian *S. scrofa* (pig) belongs.

Figure 5.2: The active site residues shared by all three GST proteins are marked in boxes. The active site residues in green boxes are aligned together, but the others in red boxes are not.

The Resulting RE-MuSiC:

```
AtGST           --AGIKVFGHPASIATRRVLIALHEKNLDFELVHVELKDGEHKKEPFLSRNPFGQVPAFE 58
SjGST           MSPILGYWKIKGLVQPTRLLLEYLEEKYEEHLYERDEGD-KWRNKKFELGLEFPNLPYYI 59
SsGST           PPYTITYFPVRGRCEAMRMLLADQDQSWKEEVVTMET----WP--PLKPSCLFRQLPKFQ 54

AtGST           DGDLKLFESRAITQYIAHRYENQGTNLLQTDSKNISQYAIMAIGMQVEDHQFDPVASKLA 118
SjGST           DGDVKLTQSMAIIRYIADKHN-----MLGGCPKERAEISMLEG--AVLDIRYG--VSRIA 110
SsGST           DGDLTLYQSNAILRHLGRSFG-----LYGKDQKEAALVDMVND--GVEDLRCK--YATLI 105

AtGST           FEQIFKSIYGLTTDEAVVAEEEAKLAKVLDVYEARLKE----FKYLAGETFTLTD--LHH 172
SjGST           YSKDFETLK---------VDFLSKLPEMLKMFEDRLCH----KTYLNGDHVTHPDFMLYD 157
SsGST           YTN-YEAGK---------EKYVKELPEHLKPFETLLSQNQGGQAFVVGSQISFADYNLLD 155

AtGST           IPAIQYLLG------TPTKKLFTER----PRVNEWVA--EITKRPAS-------EKVQ-- 211
SjGST           ALDVVLYMDPMCLDAFPKLVCFKKRIEAIPQIDKYLKSSKYIAWPLQGWQATFGGGDHPP 217
SsGST           LLRIHQVLNPSCLDAFPLLSAYVARLSARPKIKAFLASPEHVNRPIN------GNGKQ-- 207

AtGST           - 211
SjGST           K 218
SsGST           - 207
```

Figure 5.3: The constrained sequence alignment produced by RE-MuSiC, using the pattern of "[ST]-x(2)-[DE]" (PS00006) as the constraint, in which the residues shaded in yellow match the pattern. In addition, the residues in green boxes that correspond to the active sites shared by these three GST proteins are aligned together.

5.2 RNA Sequences with Phylogenetically Conserved Pseudoknots

In this experiment, we aligned the 3' untranslated region (3'-UTR) sequences of the following four coronaviruses, where HCoV-229E and PEDV are group 1 coronaviruses, while BCoV and MHV belong to group 2.

1. HCoV-229E: human 229E coronavirus whose accession number in GenBank is af304460.
2. PEDV: porcine epidemic diarrhea virus whose accession number in GenBank is af353511.
3. BCoV: bovine coronavirus whose accession number in GenBank is af220295.
4. MHV: murine hepatitis virus whose accession number in GenBank is af201929.

It has been reported that phylogenetically conserved pseudoknots found in the 3'-UTRs of these four coronaviruses (refer to Figure 5.4) have been postulated to be involved in their RNA replication [54]. Notice that the pairwise sequence identities between sequences in the different groups are extremely low. Hence, it was difficult for ClustalW to have the sequence fragments corresponding to the conserved pseudoknots aligned together (see Figure 5.5 for example). However, as shown in Figure 5.6, this goal was achieved by RE-MuSiC when the pseudoknot consensus (as shown in Figure 5.7), derived by Williams *et al.*, [54] from the 3'-UTRs of various coronaviruses, was used as the constraint. In general, loops in pseudoknots are less conserved. To enhance the flexibility of the consensus, hence, we treat the nucleotides involved in the loop regions as "don't care" symbols ("x") and describe the consensus as

"x(5)-C-U-x(4)-C-x(15,16)-U-G-x(2)-A-x(5,7)-G-x(4)-A-G-x(7,10)-U-x(3)-A-x(5)".

This experiment suggests that RE-MuSiC is able to help locate those sequence fragments that are conserved from the structural point of view.

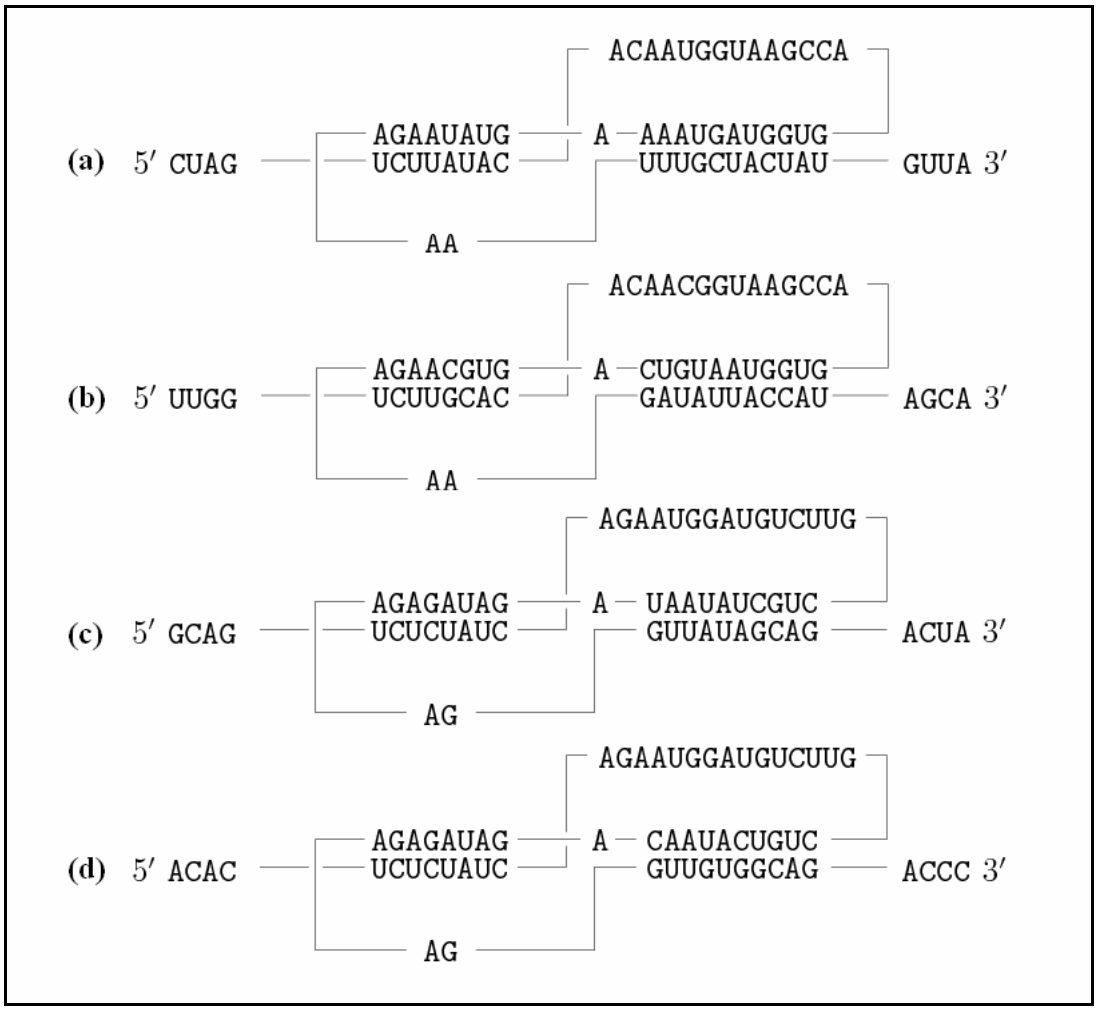


Figure 5.4: Phylogenetically conserved pseudoknots in the 3'-UTRs of four coronavirus. (a) HCoV-229E (human 229E coronavirus), (b) PEDV (porcine epidemic diarrhea virus), (c) BCoV (bovine coronavirus), (d) MHV (murine hepatitis virus).

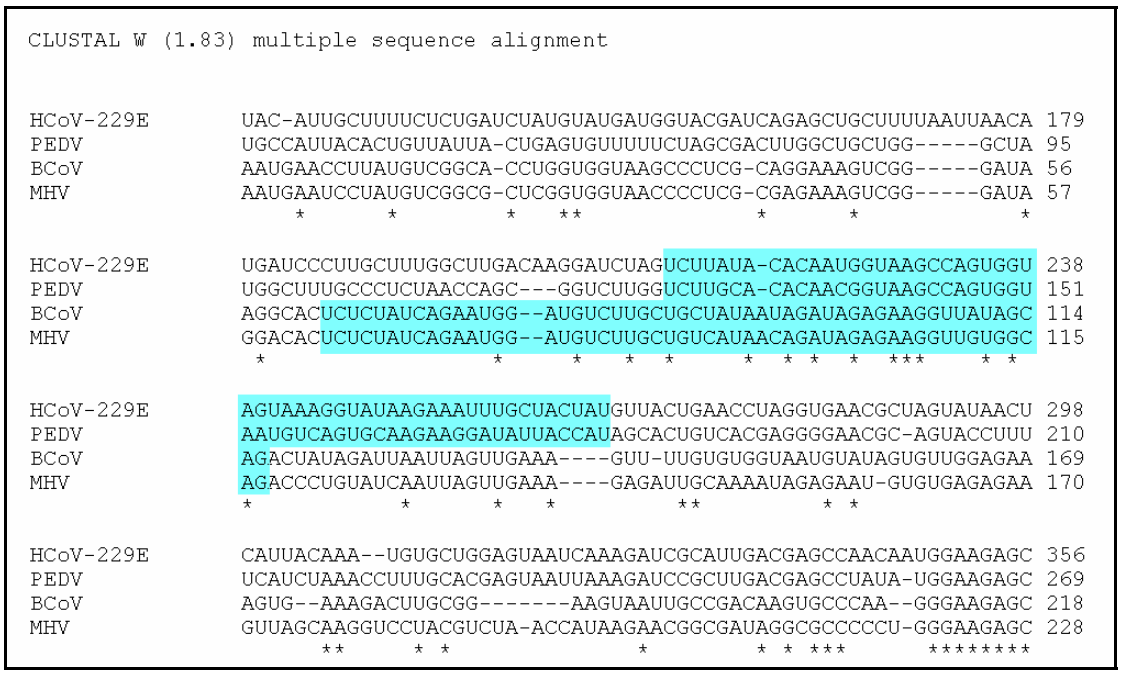


Figure 5.5: A partial view of the alignment produced by ClustalW, where the fragments shaded in light blue corresponds to the phylogenetically conserved pseudoknots in the 3'-UTRs of the four coronaviruses. Notably, these four shaded fragments were not aligned together.

The Resulting RE-MuSiC:

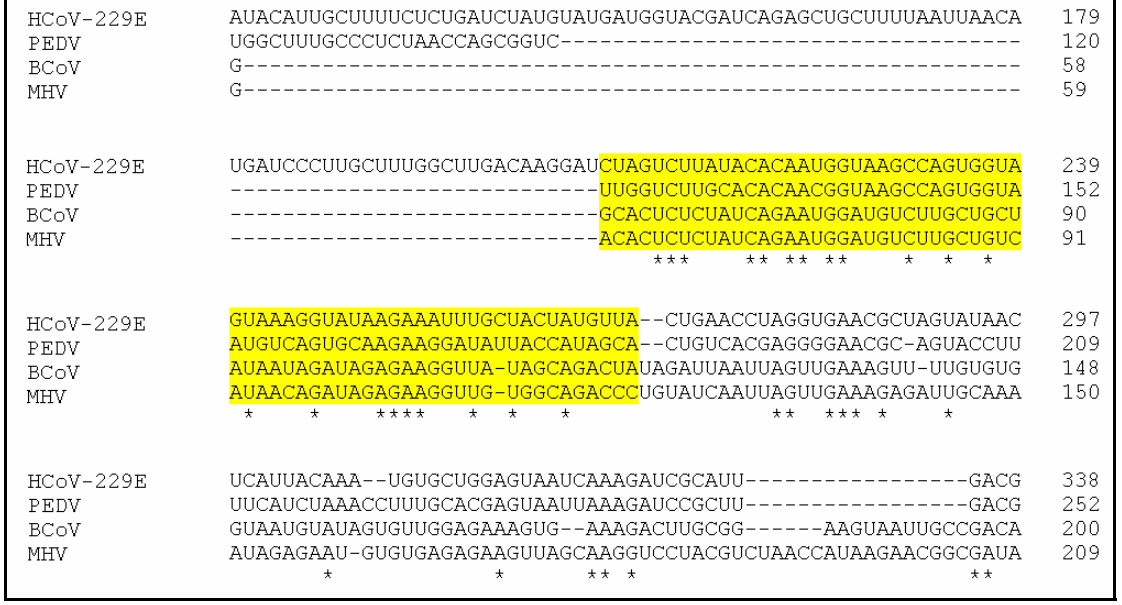


Figure 5.6: A partial view of the alignment produced by RE-MuSiC using the constraint of "x(5)-C-U-x(4)-C-x(15,16)-U-G-x(2)-A-x(5,7)-G-x(4)-A-G-x(7,10)-U-x(3)-A-x(5)", where the fragments shaded in yellow, corresponding to the phylogenetically conserved pseudoknots in the 3'-UTRs of the four coronaviruses, are aligned together.

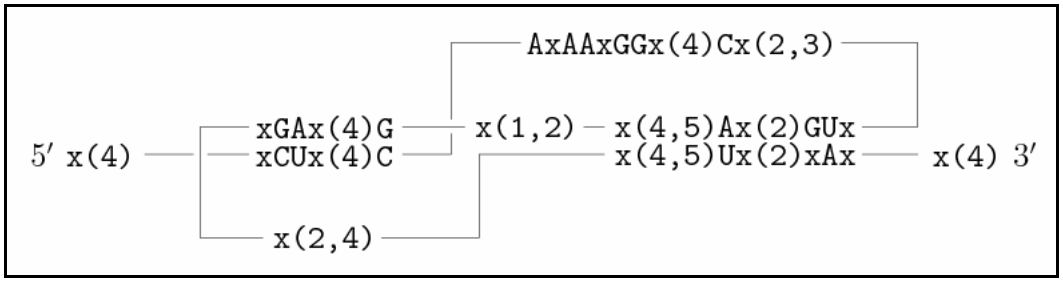


Figure 5.7: The consensuses adapted from [54], which was derived by Williams *et al.* from the 3'-UTRs of various coronaviruses, including HCoV-229E, PEDV, BCoV and MHV.

Chapter 6
Conclusions

In this thesis, we studied the RECMSA problem, whose aim is to find an RECMSA for the input sequences with several user-specified regular expression constraints such that substrings of the input sequences whose bases match regular expression constraint are aligned together. In this model, each of the user-specified constraints is a regular expression, which is useful in expressing biologically important sites such as those stored in PROSITE, as well as structural elements which often involve variable ranges in them. In contrast, the plain-strings-with-mismatches model adopted in previously available tools, MuSiC and MuSiC-ME, is not flexible enough to express such patterns.

We adopted the dynamic programming and divide-and-conquer techniques to design a time and memory efficient algorithm for optimally solving the RECPSA problem. In addition, we designed a method to find in the resulting alignment the regions responsible for the satisfactions of the constraints. Based on the algorithm, we developed a web Server RE-MuSiC for the RECMSA problem using the progressive approach. The algorithm underlying RE-MuSiC represents an improvement over the previously proposed algorithm [2], and is more appropriate for implementation in a web-server.

Experiments on GST proteins and on coronaviruses with phylogenetically

conserved pseudoknots demonstrated that, with additional knowledge incorporated, RE-MuSiC is able to produce meaningful alignments in which important residues or structural elements can be aligned properly, even if the similarity among input sequences is low. Such ability is also useful for prediction purposes.

References

[1] Arslan, A.N. (2005) Regular expression constrained sequence alignment. In *Proc. 16th Annual Symposium on Combinatorial Pattern Matching (CPM05)*, vol. 3537 of *Lecture Notes in Computer Science*, Springer, pp. 322-333.

[2] Arslan, A.N. (2005) Multiple sequence alignment containing a sequence of regular expressions. In *Proc. IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB05)*, pp.1-7.

[3] Arslan, A. N. (2006) An algorithm with linear expected running time for string editing with substitutions and substring reversals. *The Proceedings of the Biotechnology and Bioinformatics Symposium (BIOT-2006)*, pp. 90-96, Provo, Utah, October 20-21, 2006 (acceptance rate = 40%)

[4] Arslan, A. N. and He, D. (2006) An improved algorithm for the regular expression constrained multiple sequence alignment problem. *The Proceedings of the 6th IEEE Symposium on Bioinformatics and Biotechnology (BIBE 2006)*, pp. 121-126, Washington, DC, October 16-18, 2006.

[5] Arslan, A. N**.** (≥2007) Regular expression constrained sequence alignment, *Journal of Discrete Algorithms,* Elsevier (to appear)

[6] Bafna, V., Lawler, E. L. & Pevzner, P. A. (1997) Approximation algorithms for multiple sequence alignment. *Theoretical Computer Science*, **182**, 233–244.

[7] Bonizzoni, P. & Vedova, G. D. (2001) The complexity of multiple sequence alignment with SP-score that is a metric. *Theoretical Computer Science*, **259**, 63–79.

[8] Carrillo, H. & Lipman, D. (1988) The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, **48**, 1073–1082.

[9] Chan, S. C.,Wong, A. K. C. & Chiu, D. K. Y. (1992) A survey of multiple sequence comparison methods. *Bulletin of Mathematical Biology*, **54**, 563–598.

[10] Cheng, C.Y., Chang, C.H., Wu, Y.J., & Li, Y.K. (2006) Exploration of glycosyl hydrolase family 75, a chitosanase from Aspergillus fumigatus. *J. Biol. Chem.*, **281**, 3137-3144.

[11] Chin, F.Y.L., Ho, N.L., Lam, T.W., Wong, P.W.H., & Chan, M.Y. (2005) Efficient constrained multiple sequence alignment with performance guarantee. *J. Bioinform. Comput. Biol.*, **3**, 1-18.

[12] Chung, Y.-S., Lu, C.L., & Tang, C.Y. (2006) Efficient algorithms for regular expression constrained sequence alignment. In *Proc. 17th Annual Symposium on Combinatorial Pattern Matching* (CPM06), vol. 4009 of *Lecture Notes in Computer Science*, Springer, pp. 389-400.

[13] Corpet, F. (1988) Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Research*, **16**, 10881–10890.

[14] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18:341–343, 1975.

[15] Depiereux, E., & Feytmans, E. (1992) MATCH-BOX: A fundamentally new algorithm for the simultaneous alignment of several protein sequences. *Comput. Appl. Biosci.*, **8**, 501-509.

[16] Dunbrack, R.L. (2006) Sequence comparison and protein structure prediction. *Curr. Opin. Struct. Biol.*, **16**, 374-384.

[17] Feng, D. F. & Doolittle, R. F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, **25**, 351–360.

[18] Gusfield, D. (1993) Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of Mathematical Biology*, **55**, 141–154.

[19] Gusfield, D. (1997) *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press.

[20] He, D., Arslan, A. N., and Ling, A. C. H. (2006) A fast algorithm for the constrained multiple sequence alignment problem. *Acta Cybernetica*, **17**: 701-717.

[21] Higgins, D. & Sharpe, P. (1988) CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, **73**, 237–244.

[22] Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. 2nd edn. *Addison-Wesley* (2001).

[23] Huang, C.H., Lu, C.L., & Chiu, H.T. (2005) A heuristic approach for detecting RNA H-type pseudoknots. *Bioinformatics*, **21**, 3501-3508.

[24] Hulo, N., Sigrist, C.J.A., Le Saux, V., Langendijk-Genevaux, P.S., Bordoli, L., Gattiker, A., De Castro, E., Bucher, P., and Bairoch, A. (2004) Recent improvements to the PROSITE database. *Nucleic Acids Res.*, **32**, 134-137.

[25] Kruskal, J. (1956) On the shrtest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, **7**, 48–50.

[26] Li, M., Ma, B. & Wang, L. (2000) Near optimal multiple alignment within a band in polynomial time. In Proceedings of the Thirty Second Annual ACM *Symposium on Theory of Computing* (STOC 2000) pp. 425–434 ACM Press, Portland.

[27] Lipman, D. & Pearson, W. (1985) Rapid and sensitive protein simularity search. *Science*, **227**, 1435–1411.

[28] Lu, C.L., & Huang, Y.P. (2005) A memory-efficient algorithm for multiple sequence alignment with constraints. *Bioinformatics*, **21**, 20-30.

[29] Myers, G., Selznick, S., Zhang, Z., & Miller, W. (1996) Progressive multiple alignment with constraints. *J. Comput. Biol.*, **3**, 563-572.

[30] Morgenstern, B., Dress, A., & Werner, T. (1996) Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl. Acad. Sci. USA*, **93**, 12098-12103.

[31] Morgenstern, B., Frech, K., Dress, A., & Werner, T. (1998) DIALIGN: Finding local similarities by multiple sequence alignment. *Bioinformatics*, **14**, 290-294.

[32] Morgenstern, B. (1999) DIALIGN 2: Improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, **15**, 211-218.

[33] Morgenstern, B. (2004) DIALIGN: Multiple DNA and protein sequence alignment at BiBi-Serv. *Nucleic Acids Res.*, **32**, W33-W36.

[34] Morgenstern, B., Werner, N., Prohaska, S.J., Schneider, R.S.I., Subramanian, A.R., Stadler, P.F., & Weyer-Menkhoff, J. (2005) Multiple sequence alignment with user-defined constraints at GOBICS. *Bioinformatics*, **21**, 1271-1273.

[35] Morgenstern, B., Prohaska, S.J., Pohler, D., & Stadler, P.F. (2006) Multiple sequence alignment with user-defined anchor points. *Algorithms Mol. Biol*,, **1**, 6.

[36] Notredame, C. (2002) Recent progresses in multiple sequence alignment: a survey. *Pharmacogenomics*, **3**, 131-144.

[37] Nicholas, H. B., Ropelewski, A. J. & Deerfield, D. W. (2002) Strategies for multiple sequence alignment. *Biotechniques*, **32**, 592–603.

[38] Needleman, S. & Wunsch, C. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Evolution*, **48**, 443–453.

[39] Pearson, W. (1991) Searching protein sequence libraries: Computation of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithm. *Genomics*, **11**, 635–650.

[40] Pevzner, P. A. (1992) Multiple alignment, communication cost, and graph matching. *SIAM Journal on Applied Mathematics*, **52**, 1763–1779.

[41] Reeder, J., Hochsmann, M., Rehmsmeier, M., Voss, B., & Giegerich, R. (2006) Beyond Mfold: Recent advances in RNA bioinformatics. *J. Biotechnol.*, **124**, 41-55.

[42] Reinemer, P., Prade, L., Hof, P., Neuefeind, T., Huber, R., Zettl, R., Palme, K., Schell, J., Koelln, I., Bartunik, H.D. *et al*. (1996) Three-dimensional structure of glutathione S-transferase from Arabidopsis thaliana at 2.2 A resolution: Structural characterization of herbicide-conjugating plant glutathione S-transferases and a novel active site architecture. *J. Mol. Biol.*, **255**, 289-309.

[43] Saitou, N. & Nei, M. (1987) The neighbor-joining method: a new mothod for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, **4**, 406–425.

[44] Sammeth, M., Morgenstern, B., & Stoye, J. (2003) Divide-and-conquer multiple alignment with segment-based constraints. *Bioinformatics*, **19 Suppl. 2**, ii189-ii195.

[45] Shuler, G.D., Altschul, S.F., & Lipman, D.J. (1991) A workbench for multiple alignment construction and analysis. *Proteins: Struct., Funct., Genet.*, **9**, 180-190.

[46] Sneath, P. & Sokal, R. (1973) *Numerical Taxonomy. Freeman, San Francisco*, CA.

[47] Song, B., Choi, J.H., Chen, G.Y., Szymanski, J., Zhang, G.Q., Tung, A.K.H., Kang, J., Kim, S., & Yang, J. (2006) ARCS: An aggregated related column scoring scheme for aligned sequences. *Bioinformatics*, **22**, 2326-2332.

[48] Tang, C.Y., Lu, C.L., Chang, M.D.T., Tsai, Y.T., Sun, Y.J., Chao, K.M., Chang, J.M., Chiou, Y.H., Wu, C.M., Chang, H.T., and Chou, W.I. (2003) Constrained multiple sequence alignment tool development and its application to RNase family alignment. *J. Bioinform. Comput. Biol.*, **1**, 267-287.

[49] Taylor, W. R. (1987) Multiple sequence alignment by a pairwise algorithm. *CABIOS*, **3**, 81–87.

[50] Thompson, J.D., Higgins, D.G., Gibson, T.J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **22**, 4673-4680.

[51] Thompson, J.D., Plewniak, F., Thierry, J.-C., & Poch., O. (2000) DbClustal: Rapid and reliable multiple alignments of protein sequences detected by database searches. *Nucleic Acids Res.*, **28**, 2919-2926.

[52] Tsai, Y.-T., Huang, Y.P., Yu, C.T., & Lu, C.L. (2004) MuSiC: A tool for multiple sequence alignment with constraints. *Bioinformatics*, **20**, 2309-2311.

[53] Wang, L. & Jiang, T. (1994) On the complexity of multiple sequence alignment. *Journal of Computational Biology*, **1**, 337–348.

[54] Williams, G.D., Chang, R.Y. and Brian, D.A. (1999) A phylogenetically conserved hairpin-type 3' untranslated region pseudoknot functions in coronavirus RNA replication. *J. Virol.*, **73**, 8349-8355.