THE UNIVERSITY OF MANITOBA

A New Algorithm for Sequence Alignment

by

Tony Tze-Chung Wu

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE IN STATISTICS

DEPARTMENT OF STATISTICS

WINNIPEG, MANITOBA

August, 2005

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

"A New Algorithm for Sequence Alignment"

BY

Tony Tze-Chung Wu

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

Of

MASTER OF SCIENCE

Abstract

This thesis focuses on applying Markov chain Monte Carlo approach with Metropolis algorithm in molecular biology. We introduce some basic concepts including DNA, RNA, and amino acid in molecular biology and some basic concepts in stochastic process such as discrete-time discrete-space Markov chain, and Metropolis algorithm. We also discuss two topics (phylogenetic tree and sequence alignment) in molecular biology using Metropolis algorithm. We explain how to construct Markov chain model in phylogenetic tree construction based on Li et al. (2000)'s paper. However, we will skip the simulation part and focus on the probability distribution of the phylogenetic tree. Furthermore, we introduce a new method using Markov chain Monte Carlo algorithm for sequence alignment. We explain how to construct the alignment matrix and use it to run simulation via Metropolis algorithm.

Acknowledgements

I would like to first sincerely thank my initial supervisor, Dr. Dean Slonowsky, for his patience and constant support. His patience allowed me to take my time to finish this thesis. He also encouraged me to keep writing this thesis, knowing my English is not perfect. Over the years, he has taught me to use different technical programs such as MATLAB and Latex. I would like to thank my advisor, Dr. Xikui Wang for spending his time helping me finish this thesis. I would also like to thank my committee members, Dr. Jin Zhang and Dr. Jeffrey Pai for giving me helpful suggestions and comments. I wish to thank Dr. T. Visentin from University of Winnipeg for helping me solve the combinatoric problem in my thesis. Thank you to all the professors at Department of Statistics in University of Manitoba and University of Winnipeg who have guided me throughout my undergraduate and graduate studies. Furthermore, I want to say thank you to all of my colleagues who were studying and helping me throughout the years. Last but not least, I would like to thank my parents, my sister, and my brother for their constant support.

Table of Contents

Chapter 1

Biological Background

Computational biology, also known as *bioinformatics*, is concerned with the development of efficient algorithm, and statistical analysis. The first chapter contains some simple concepts of molecular biology that will enable us to understand the technical terms such as DNA, RNA, sequence alignment and so on in later chapters. We will only discuss a few of the biology concepts very briefly and use it as a tool for applying our statistical algorithm which is under Markov chain in stochastic process.

To begin this chapter, we first introduce some basic molecules used in biology.

1.1 Nucleic Acids

1.1.1 Nucleotides

DNA and RNA are composed of nucleotides; a *nucleotide* is a molecule consisting of a base, a ribose sugar (in DNA deoxyribose), and a phosphate group. The base is bound to a $1'$ carbon and the phosphate is bound to a $5'$ carbon. *Bases* are the molecules attached to each $1'$ carbon in the backbone. DNA are composed of four bases: adenine (A), guanine (G), cytosine (C), and thymine (T). The base thymine (T) is replaced by uracil (U) in RNA. Bases A and G belong to a group of substance called *purines*. On the other hand, bases C and T belong to another group of substance called *pyrimidines* (Clote and Backofen, 2000).

1.1.2 DNA

DNA molecules are composed of double strands. Each base in one strand is paired with a base in another strand so that two strands can hold together and form a double helix structure. The bases A and T are said to be complements of each other, or a pair of complementary bases whereas, bases C and G are complementary bases. This is known as the so-called Watson-Crick rule. When we refer to DNA molecules, base pairs provide the unit of length and are abbreviated **bp** (Setubal and Meidanis, 1997).

Double-stranded DNA forms a helix, where one strand goes in the direction from $5'$ to $3'$, while the second strand goes from $3'$ to $5'$; thus the second strand is the *reverse complement* of the first strand. This mechanism explains how DNA can replicate from one cell into a billion cells (Clote and Backofen, 2000).

1.1.3 RNA

In ribonucleic acids (RNA), there are no traces of thymine (T); instead, uracil (U) is present, so RNA now consists of bases adenine, cytosine, guanine, and uracil. The RNA ribose contains the composition of sugar which has an extra hydroxyl (OH) group at the $2'$ position, so it is able to form more hydrogen bonds than DNA. RNA manifests a number of catalytic properties, which plays both an information storage and enzymatic role. In contrast to DNA, RNA does not form a double helix and is only single-stranded (Clote and Backofen, 2000).

1.2 Proteins

1.2.1 Amino acids

There are 20 different amino acids and chains of these simpler molecules form a protein. The general form of an amino acid consists of chemical group (chain residue), amino group (NH_2), carboxyl group ($COOH$), and the α carbon (C_α). There are 20 different chain residues, which have different chemical properties depending on the chain of molecules (Clote and Backofen, 2000). 20 amino acids are listed in Table 1.1:

Table 1.1: Amino acids

One-letter code	Three-letter code	Name
A	Ala	Alanine
C	Cys	Cysteine
D	Asp	Aspartic Acid
E	Glu	Glutamic Acid
F	Phe	Phenylalanine
G	Gly	Glycine
H	His	Histidine
I	Ile	Isoleucine
K	Lys	Lysine
L	Leu	Leucine
M	Met	Methionine
N	Asn	Asparagine
P	Pro	Proline
Q	Gln	Glutamine
R	Arg	Arginine
S	Ser	Serine
T	Thr	Threonine
V	Val	Valine
W	Trp	Tryptophan
Y	Tyr	Tyrosine

Two amino acids can be connected by a bond called a peptide, where the carboxyl group of the first amino acid reacts with the amino group of the second. Using the peptide bond, long linear chains of amino acids called proteins can be generated (Clote and Backofen, 2000).

1.2.2 Protein structure

Once the protein is generated using a peptide bond, proteins can be roughly divided into three groups: globular (enzymes), fibril (collagen, elastin), and membrane proteins. As commonly known, enzymes are the proteins which act as catalysts of chemical reaction. A catalyst is a substance that speeds up a chemical reaction. Other examples of protein function are immune defense and oxygen transport.

There are four different levels in the structure of a protein. The primary structure is the amino acid sequence of protein, and the secondary structure describes the regions in the primary structure where secondary structure elements occur. Tertiary structure is the three dimensional structure of a protein domain in the native structure. Quaternary structure is the three dimensional, native structure of the fully functional protein (Clote and Backofen, 2000).

1.3 Phylogenetic Trees

One of the most important subjects in the study of molecular biology is to reconstruct the evolutionary history of genes and species. The reasons of phylogenetic studies are to find out the time of divergence between observed genes, species and organisms, and to reconstruct the real genealogical ties between them (Li et al., 2000). However,

in almost every experimental case, the phylogenetic tree underlying the evolutionary history of observed sequences is unknown.

A phylogenetic tree is used to illustrate the evolutionary relationships between organisms. A phylogenetic tree is a non-cycle connected graph that consists of branches and nodes. The taxonomic units are defined by the nodes. The nodes can be of either populations, genes, or species. The branching pattern of the tree represents the relationships among the units in terms of the evolutionary connection and is called topology. The evolution time or the number of mutations is usually proportional to the lengths of the branches (Li et al., 2000).

There are two kind of nodes in a phylogenetic tree which are external nodes and internal nodes. External nodes are the nodes at the tips of the trees and the rest of the nodes are called internal nodes. Internal nodes are the ancestors of the external nodes. Evolution happens independently along branches connecting with each internal node. An example is shown in Figure 1.1. Nodes A-E in both Figure 1.1 (a) and Figure 1.1 (b) are external nodes. Nodes F-I in Figure 1.1 (a) and nodes F-H in Figure 1.1 (b) are internal nodes.

There are two kind of phylogenetic trees: rooted trees [as in Figure 1.1 (a)] and unrooted trees [as in Figure 1.1 (b)]. A particular node called root [node I in Figure 1.1 (a)] is found only in the rooted tree. Only one path is connected with the root and each of the external nodes. The evolutionary times are related to the directions of these paths, and the root is the common ancestor of all external nodes. In an unrooted tree, only relationship among all external nodes or the taxa is specified. However, it does not define the evolutionary path (Li et al., 2000).

In chapter 3, we will discuss a statistical method to find out a possible phylo-

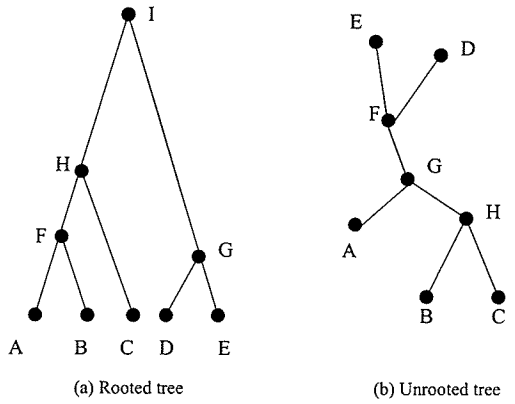

Figure 1.1: Rooted and unrooted trees

genetic tree among all possible trees. We will show how to apply an algorithm to construct the optimal branching pattern and splitting time of phylogenetic tree.

1.4 Sequence alignment

Sequence alignment is one of the primary subjects in molecular biology. Sequence alignment is usually arranging two or more sequences together and highlighting the differences or similarities. Usually, sequence alignments are filled with gaps (or presented by dash), identical or different characters placed in rows and columns. We use sequence alignments as a tool to study the evolution of sequences from a common ancestor, especially biological sequences, such as DNA, RNA, and protein sequences.

If two characters are different in the column of a sequence alignment, we say this as a *mismatch* and this usually represents a mutation, and the gap represents a deletion or insertion. Conversely, if two characters lie in the same column, we say it is a *match* (Setubal and Meidanis, 1997).

1.4.1 SP score measure

If we have two or more sequences and we want to align them, how do we know whether they are similar or different? In other word, is there a way to find an optimal sequence alignment? Before we set an algorithm to figure out the best alignment, we need to use a simple measurement called *sum-of-pairs score*. Sum-of-pairs (SP) score is restricted to a purely additive function. This scoring system is used to score two of the same characters in the column (match) as a value of 1. Since a mismatch can have two different characters or one character with one dash, we score two different characters as a value of -1, -2 respectively. Finally, we score two-dash characters as a value of 0. The reason of using this scoring system is rewarding the match characters and penalizing the mismatch characters (Setubal and Meidanis, 1997). You may wonder why we do not penalize two-dash characters if they are in the same row of the alignment. The reason is related to pairwise alignments. Sometimes, in multiple alignment, we may choose two of the sequences and check whether they are aligned to each other and neglect the rest of the sequences. In this way, we will produce a pairwise alignment with two sequences. The only difference is we may have some columns with two dashes in the same column. We can remove those columns and find a true induced pairwise alignment (Setubal and Meidanis, 1997).

Before we show a simple example to calculate the SP score, some notations used here will be introduced. First, let SP(α) be the SP score of a multiple alignment α and $p(a, b)$ be the pairwise score of characters a and b:

$$SP(\alpha) = \sum_{i<j} score(\alpha_{ij})$$

where α_{ij} is the pairwise alignment of characters a and b induced by alignment α

on sequence s_i and s_j. This scoring function is also used in Setubal and Meidanis, 1997. There are two ways to find SP(α). One way is to find the score of each induced pairwise alignment and add all pair scores together. Another way is to find the scores of each column and add the scores together. Hence, the SP score can also be written as:

$$SP(\alpha) = \sum_{1 \le i < j \le N} \sum_{k=1}^{L} p[\bar{s}_i(k), \bar{s}_j(k)]$$

where N is the number of sequences in the alignment, L is the size of the sequence and k indicates the k^{th} column of the alignment (Setubal and Meidanis, 1997). Note that the two methods of scoring alignments are identical only when we use $p(-, -)$= 0.

Example 1.4.1 Suppose we have a multiple DNA sequence alignment α as follows:

$$\begin{array}{lcccccc} s_1: & A & A & G & C & - & T \\ s_2: & A & A & T & G & G & T \\ s_3: & A & T & - & C & - & T \end{array}$$

$$\begin{aligned} SP(\alpha) &= \sum_{1 \le i < j \le 3} \sum_{k=1}^{6} p[\bar{s}_i(k), \bar{s}_j(k)] \\ &= \sum_{k=1}^{6} p[\bar{s}_1(k), \bar{s}_2(k)] + \sum_{k=1}^{6} p[\bar{s}_1(k), \bar{s}_3(k)] + \sum_{k=1}^{6} p[\bar{s}_2(k), \bar{s}_3(k)] \\ &= -1 + 0 - 4 \\ &= -5 \end{aligned}$$

This scoring function is used later in Chapter 4 when we introduce an algorithm to find an optimal sequence alignment.

Chapter 2

General MCMC Algorithm

2.1 Discrete time discrete space Markov chains

In this section, some fundamental results and notations for discrete-time discrete space Markov chains (MC) will be introduced. We begin with the basic definition:

Definition 2.1.1 *Let $X = \{X_0, X_1, X_2, \ldots\}$ be a discrete-time, discrete space stochastic process with a finite or infinitely countable state space $\mathcal{S}$. X is called a Markov chain if*

$$\boldsymbol{Pr}(X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) = \boldsymbol{Pr}(X_{n+1} = j | X_n = i)$$

for all states, $i, j, i_{n-1}, \ldots, i_1, i_0$ and time n.

If $\mathbf{Pr}(X_{n+1} = j | X_n = i) = \mathbf{K}_{i,j}$ for all n and $i, j \in \mathcal{S}$, X is called time-homogeneous. $\mathbf{K} = (\mathbf{K}_{i,j})$ is the transition probability matrix (t.p.m.), where $\mathbf{K}_{i,j}$ is the $(i,j)^{th}$ entry of the matrix $\mathbf{K}$. There are two properties of $\mathbf{K}$:

(a) $\mathbf{K}_{i,j} \geq 0$ and $i, j \in \mathcal{S}$.

(b) $\sum_{j \in \mathcal{S}} \mathbf{K}_{i,j} = 1$ which all the row sums is equal to 1.

Also, if $\mathbf{K}_{i,j} = \mathbf{K}_{j,i}$ for all $i,j \in \mathcal{S}$, then $\mathbf{K}$ is called to be symmetric t.p.m.. The n-step transition probabilities are

$$\mathbf{K}^n_{i,j} = \mathbf{Pr}(X_{m+n} = j | X_m = i)$$

for $n = 0, 1, 2, \ldots$, and all $i, j \in \mathcal{S}$. We write $\mathbf{K}_{i,j}$ to refer the one-step transition probabilities instead of writing $\mathbf{K}^1_{i,j}$.

Definition 2.1.2 *The initial distribution of a MC X is the probability mass function (p.m.f.), $\pi = (\pi_i)$, $i \in \mathcal{S}$ of X_0*

Given states $i, j \in \mathcal{S}$, we say state j is accessible from state i and write $i \to j$ if there exists $n \geq 1$ such that $\mathbf{K}^n_{i,j} > 0$. States i and j are said to communicate if $i \to j$ and $j \to i$. That is, if there exists some $n_1 > 0$, $n_2 > 0$ such that $\mathbf{K}^{n_1}_{i,j} > 0$ and $\mathbf{K}^{n_2}_{j,i} > 0$ and we label this as $i \longleftrightarrow j$.

Definition 2.1.3 *A Markov chain X is said to be irreducible if all states communicate, that is $i \longleftrightarrow j$ for all $i, j \in \mathcal{S}$.*

Definition 2.1.4 *The period of a state $i \in \mathcal{S}$, denoted $d(i)$, is the greatest common divisor* (gcd) *of $\{n \geq 1 : \boldsymbol{K}^n_{i,i} > 0\}$. If $d(i) = 1$, then i is called aperiodic. In particular, if $\boldsymbol{K}_{i,i} > 0$, then i is aperiodic.*

Theorem 2.1.5 *If $i \longleftrightarrow j$, then $d(i) = d(j)$. Therefore, if X is irreducible, then all states have the same period.*

After introducing the concepts of the aperiodicity and irreducibility, a useful result that involves these two concepts is stated as follows.

Theorem 2.1.6 *If a MC X is irreducible, aperiodic and the number of states $|\mathcal{S}|$ is finite, then there exists a unique p.m.f., $\hat{\pi}$ on $\mathcal{S}$ such that, for any state $j \in \mathcal{S}$,*

$$\lim_{n \to \infty} \boldsymbol{Pr}(X_n = j) = \hat{\pi}_j$$

independent of the initial distribution of X. *In short, if* n *is "sufficiently large", then the distribution of* X_n *is approximately* $\hat{\pi}$. *In addition,*

(1) $\hat{\pi}\boldsymbol{K} = \hat{\pi}$, *that is,* $\hat{\pi}_j = \sum_{i \in \mathcal{S}} \pi_i \boldsymbol{K}_{i,j}$ *for all* $j \in \mathcal{S}$.

(2) $\lim_n \boldsymbol{K}^n_{i,j} = \hat{\pi}_j$ *for all* $i, j \in \mathcal{S}$

(3)

$$\begin{aligned} \hat{\pi}_i &= \lim_{n \to \infty} \frac{\textit{number of times } X \textit{ equals } i}{n} \\ &= \textit{long-run proportion of time } X \textit{ spends in state } i. \end{aligned}$$

(4)

$$\begin{aligned} \frac{1}{\hat{\pi}_i} &= \textit{Expected number of jumps between visits to } i \\ &= \textit{mean time between visits to state } i \end{aligned}$$

The p.m.f. $\hat{\pi}$ in the above theorem is called the steady-state distribution of X.

This theorem can be applied as follow: if we have a target distribution $\hat{\pi}$ on $\mathcal{S}$, and if we can construct a MC X with steady-state distribution $\hat{\pi}$, then we can draw an element from $\mathcal{S}$ according to $\hat{\pi}$ by starting X in any proceed state, that is, $X_0 = i_0$ with probability 1, for some $i_0 \in \mathcal{S}$ and run X sufficiently long so that it has "burnt-in" to steady-state, which means X_n is approximately distributed as $\hat{\pi}$. In this way, any realization (simulation) of such an X_n represents an (approximate) sample x_n from $\hat{\pi}$. In section 2.4, we will introduce the *Metropolis algorithm* which allows us to construct such an MC for any target distribution, $\hat{\pi}$.

2.2 Continuous time discrete space Markov chains

In the previous section, we introduce some basic concepts of discrete-time Markov chain. In order to generalized the concept, we introduce another Markov chain called *Continuous time Markov chain* (CTMC). We will compare the difference between discrete time and continuous time Markov chains later in the section.
We begin to define:

Definition 2.2.1 *A continuous time stochastic process,* $X = \{X(t); t \geq 0\}$ *is a continuous time Markov chain (CTMC). with discrete state space,* $\mathcal{S}$*, if given time* s *and* $s+t$ *and states* $i, j \in \mathcal{S}$

$$\boldsymbol{Pr}(X(s+t) = j | X(s) = i, X(u) = i_u; 0 \leq u < s) = \boldsymbol{Pr}(X(s+t) = j | X(s) = i)$$

X is said to be a *time homogeneous* CTMC, if, for each $t > 0$, there exists a matrix, $\mathbf{K}(t) = \{K_{i,j}(t); i, j \in \mathcal{S}\}$ such that $\mathbf{Pr}(X(s+t) = j \mid X(s) = i) = K_{i,j}(t)$, for all $s > 0$. $\mathbf{K}(t)$ is called the *transition probability function* (t.p.f) of X. $K_{i,j}(t)$ is called a *transiton probablity.* It is important to note that a general CTMC can make a jump at any point t in continuous time. Therefore, a MC X can have (with probability > 0) any number of jumps in a given time interval $(s, s+t)$, no matter how small $t > 0$ is.

Theorem 2.2.2 *Let* $X = \{X(t); t \geq 0\}$ *be a CTMC with state space* $\mathcal{S}$ *and t.p.f.,* $K_{i,j}(t)$ *where* $i, j \in \mathcal{S}$, $t \geq 0$. *If* X *starts in state* i *(i.e.* $X_0 = i$*), the amount of time* X *spends in state* i *before jumping to a new state is distributed exponential* (υ_i) *where* $\upsilon_i = \lim_{t \to 0} \frac{1 - K_{i,i}(t)}{t}$. *Usually,* υ_i *is called the jumping rates.*

Comparing Theorem 2.2.2 with discrete-time MC, if $X = \{X_n\}$ where $n = 0, 1, 2, ...$ is a discrete-time MC and if $X_0 = i$, the amount of time X spends in state i before jumping to a different state is distributed geometric $(1 - K_{i,i})$.

In order to simulate a continuous time MC, we need the following set up: first, let $X = \{X(t);\ t \geq 0\}$ be a CTMC with state-space $\mathcal{S}$ and t.p.f. $K_{i,j}(t)$, $(t \geq 0)$ and let υ_i $(i \in \mathcal{S})$ be the rates defined in Theorem 2.2.2. Define an associated discrete-time MC $Y = \{Y_n;\ n = 0, 1, 2, 3, ...\}$ as follows:

(i) $Y_0 = X_0$ (initial state of X)

(ii) Y_n = state of X after n^{th} jump

Y has state-space $\mathcal{S}$. Let $\mathbf{K}_{i,j}$, $(i, j \in \mathcal{S})$ be the t.p.m. of Y. X evolves over time as follows:

1. X starts in an initial state, i_0 at time 0.

2. Then, X waits an exponential (υ_{i_0}) amount of time, then X jumps to a state, $i_1 \neq i_0$ with probability, K_{i_0,i_1} (that is, jumps according to the MC Y).

3. X waits an exponential (υ_{i_1}) amount of time, then X jumps to state $i_2 \neq i_1$ with probability K_{i_1,i_2}.

You may ask a question how to find t.p.m. $\mathbf{K}$ of Y. Here is the answer. For each pair of states $i \neq j$ in $\mathcal{S}$, we define the *instantaneous jump rate*, as,

$$r_{i,j} = \lim_{t \to 0} \frac{K_{i,j}(t)}{t}$$

Therefore, since $K_{i,j}(t) = 0$, (if $i \neq j$)

$$r_{i,j} = \left. \frac{d\, K_{i,j}(t)}{d\, t} \right|_{t=0}, \quad \forall i \neq j$$

Also, since $K_{i,i}(0) = 1$ for all i

$$\begin{aligned} -v_i &= \lim_{t\to 0} \frac{1 - K_{i,i}(t)}{-t} \\ &= \lim_{t\to 0} \frac{K_{i,i}(0) - K_{i,i}(t)}{0 - t} \\ &= \left.\frac{d\,K_{i,i}(t)}{d\,t}\right|_{t=0} \end{aligned}$$

We must be careful that $r_{i,j}$'s are *not* probabilities, they are just rates.

Definition 2.2.3 *The generator matrix* $\mathbf{R}$ *of the CTMC* X *is the matrix* $\mathbf{R} = \{R_{i,j},\, i,j \in \mathcal{S}\}$ *where* $R_{i,j} = r_{i,j}$.

Unlike t.p.m. $\mathbf{K}$ of a MC, it can be shown that all row sums of $\mathbf{R}$ are zero. That is, $\sum_{i\in\mathcal{S}} r_{i,j} = 0$ for all $j \in \mathcal{S}$. In particular,

$$v_i = -r_{i,i} = \sum_{j\neq i,\, j\in\mathcal{S}} r_{i,j}$$

Furthermore, it can be shown that

$$\begin{cases} K_{i,j} = \frac{r_{i,j}}{\sum_{j\neq i} r_{i,j}}, & \text{when } i \neq j; \\ K_{i,i} = 0, & \forall i \in \mathcal{S}. \end{cases}$$

To demonstrated, let's see a simple example:

Example 2.2.4 (Poisson Process with rate λ)

$$\mathbf{R} = \begin{pmatrix} -\lambda & \lambda & 0 & 0 & . \\ 0 & -\lambda & -\lambda & 0 & . \\ 0 & 0 & -\lambda & \lambda & . \\ . & . & . & . & . \end{pmatrix}$$

The t.p.m. of Y:

$K_{i,i+1} = \frac{\lambda}{\lambda} = 1, \forall\, i \in \mathcal{S}$

$$\mathbf{K} = \begin{pmatrix} 0 & 1 & 0 & 0 & . \\ 0 & 0 & 1 & 0 & . \\ 0 & 0 & 0 & 1 & . \\ . & . & . & . & . \end{pmatrix}$$

This is a simple MC, because X jumps from state 0 to state 1 to state 2... (i.e., $\{0 \rightarrow 1 \rightarrow 2 \rightarrow ...\}$) with probability 1.

Another example will be given in Chapter 3 with a Markov model for the mutation of a single DNA site.

In discrete-time MC, we have introduced the concept of convergence to steady state distribution of X. Similarly, there are some definitions of convergence to steady state distribution in continuous time MC as well. Here, we will make a comparison and contrast between continuous and discrete time MC as follows:

Let X= $\{X(t);\, t \geq 0\}$ be a CTMC with state space $\mathcal{S}$ and t.p.f. $K_{i,j}(t)$, for i, j $\in \mathcal{S}$ and $t \geq 0$ and let T_i = "Time of first return to i if $X(0) = i$ ", therefore, T_i will be the first time when X makes a transition to state i if $X(0) \neq i$, and this is illustrated in Figure 2.1.

(i) $i \rightarrow j$ if $K_{i,j}(t) > 0$, for some $t \geq 0$

(ii) X is *irreducible* if $i \rightarrow j$, for all $i, j \in \mathcal{S}$

(iii) *No* concept of *periodicity* of states

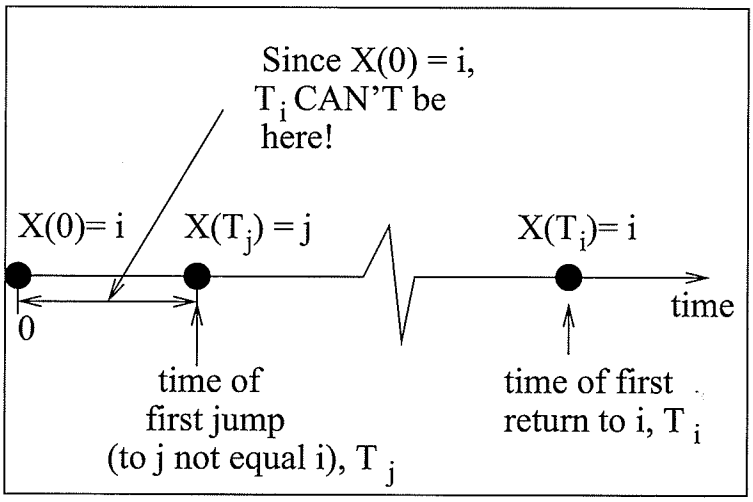

Figure 2.1: Time of first return to i if $X(0) = i$

(iv) i is called *recurrent* if $\mathbf{Pr}(T_i < \infty | X(0) = i) = 1$. (i.e., $\mathbf{Pr}(X$ will have i, then return to $i) = 1)$

(v) If i is not recurrent, it is *transient.*

(vi) i is called *positive-recurrent* if $\mathbf{E}[T_i | X(0) = i] < \infty$. (Positive-recurrent $\Rightarrow$ recurrent)

(vii) If i is recurrent, but not positive-recurrent, it is said to be *null-recurrent.*

(viii) i is recurrent if and only if $\int_0^\infty K_{i,i}(t)dt = \infty$. Therefore, i is transient $\Leftrightarrow$ $\int_0^\infty K_{i,i}(t)dt < \infty$.

(ix) $i \leftrightarrow j \Rightarrow i$, j are both transient (or both null-recurrent, or both positive-recurrent)

(x) X is irreducible and $|\mathcal{S}| < \infty \Rightarrow$ all states of X are positive-recurrent.

Theorem 2.2.5 *If X is an irreducible, positive-recurrent CTMC, then there exists a unique p.m.f. $\hat{\pi} = \{\hat{\pi}_i,\ i \in \mathcal{S}\}$ on $\mathcal{S}$ such that $\lim_{t\to\infty} K_{i,j}(t) = \lim_{t\to\infty} \boldsymbol{Pr}(X(t) = j \,|\, X(0) = i) = \hat{\pi}_j$. In addition,*

(1) $\hat{\pi}$ satisfies $\hat{\pi}\boldsymbol{R} = \overrightarrow{0}$. (zero-row-vector)

(under the condition, $\sum_{i\in\mathcal{S}} \hat{\pi}_i = 1$)

(2) Long-run proportion of time that X spends in state $i = \hat{\pi}_i$ for all $i \in \mathcal{S}$.

(3) Expected time between consecutive transitions (i.e. jumps) to state $i = \boldsymbol{E}[T_i \,|\, X(0) = i] = \frac{1}{v_i \hat{\pi}_i} = \frac{-1}{r_{i,i}\hat{\pi}_i}$

(4) $\lim_{t\to\infty} \boldsymbol{Pr}(X(t) = i) = \hat{\pi}_i$ for all $i \in \mathcal{S}$. (i.e., $X(t)$ is approximately distributed on $\hat{\pi}$ for large time t)

The p.m.f. $\hat{\pi}$ in the above theorem is called the *steady-state* distribution of X in CTMC.

One of a widely studied in CTMC is so called the *Kolmogorov backwards equations*,

$$\frac{d}{dt}\mathbf{K}(t) = \mathbf{R} \cdot \mathbf{K}(t).$$

It can be used to "solve" the t.p.f. from matrix $\mathbf{R}$. That is, we have a system of linear differential equations,

$$\frac{d}{dt} K_{i,j}(t) = \sum_{m\in\mathcal{S}} r_{i,m}\, K_{m,j}(t),\ \forall\, i, j \in \mathcal{S}\ .$$

The general solution will be:

$$\mathbf{K}(t) = \exp(t\mathbf{R}) = \sum_{n=0}^{\infty} \frac{(t\,\mathbf{R})^n}{n!}.$$

One of the application of using this Kolmogorov equations will be found in Chapter 3. We will introduce the specified $\mathbf{R}$-matrix in Chapter 3. By using Kolmogorov equations, we can find the t.p.f. $K_{i,j}(t)$ in that particular model.

2.3 Discrete time continuous space Markov chains

In the previous two sections, two types of discrete-state Markov chain (continuous time and discrete time) were introduced. Another type, called *discrete-time continuous space* Markov chain will be discussed in this section. Unlike the previous two sections, we will only introduce some basic concepts of this Markov chain. For more detailed information, we hope the reader can find it in most of the stochastic process textbook.

We begin with: let $X = \{X_n;\ n = 0, 1, 2, ...\}$ be a stochastic process where each X_n is a continuous random variable. Toward a definition of the Markov property in this setting, we select a time, n, and states, $y, x, x_{n-1}, ..., x_1, x_0 \in \mathbb{R}$. Since $X_{n+1} = y$ has probability zero, we are forced to define the Markov property via the conditional cumulative distribution function (c.d.f.). Even though it has an intuitive meaning, there is technically also the problem of conditioning with respect to the event $[X_n = x, X_{n-1} = x_{n-1}, ..., X_0 = x_0]$, since it too has zero probability. We overcome this problem by selecting a regular conditional cumulative distribution function for X_n given its past, $X_n, X_{n-1}, ..., X_1, X_0$.

$$\mathbf{Pr}\left(X_{n+1} \leq y | X_n = x, X_{n-1} = x_{n-1}, ..., X_0 = x_0\right) = \mathbf{Pr}\left(X_{n+1} \leq y | X_n = x\right).$$

If it holds for all possible selections of states and times, we refer to X as a Markov chain. If the expression on the right-hand-side does not depend of the time, n, we write

$$K(x, y) = \mathbf{Pr}\left(X_{n+1} \leq y | X_n = x\right),\ \ \forall x, y \in \mathbb{R}$$

X is called time-homogeneous. That is, for each fixed $x \in \mathbb{R}$, $K(x, y)$ is a c.d.f. in y. In literature, $K(x, y)$ is sometimes written $K(x, dy)$, to denote that it is a "measure"

on $\mathbb{R}$ for every fixed $x \in \mathbb{R}$. Furthermore, given a "nice" subset $\mathbf{A}$ of $\mathbb{R}$ (say, $\mathbf{A}$ is an interval), one often writes

$$K(x, A) = \mathbf{Pr}\,(X_{n+1} \in A | X_n = x).$$

In a more general form, many authors write

$$\mathbf{Pr}\,(X_{n+1} \in A | X_n = x, X_{n-1} = x_{n-1}, ..., X_0 = x_0) = \mathbf{Pr}\,(X_{n+1} \in A | X_n = x).$$

That is to say: " To find the probability of X jumping to state in the set $\mathbf{A}$, given the entire history of X, we just need to know the current state of X". We call $K(x, y)$ the *transition kernel* of X (a continuous-space analogue of a t.p.m.).

Example 2.3.1 (Sum of i.i.d. random variables) Let $\{\varepsilon_n\}$ where $n = (0, 1, 2, 3, ...)$ be an i.i.d sequence of continuous random variables with common c.d.f., F. Then, we get a discrete-time, continuous-space stochastic process, X with $X_0 = x_0$ and

$$X_{n+1} = X_n + \varepsilon_n, \ \ \forall n \geq 0$$

There is a concept of irreducibility, recurrence, and aperiodicity for continuous space Markov chains. The definition of these concepts are a bit technical, and are therefore omitted from these thesis. These concepts play roles similar to those of their discrete-space analogues. Namely, it can be shown that an irreducible, aperiodic, recurrent Markov chain X possesses a unique steady-state c.d.f., F. Here, steady-state means

$$\lim_{n \to \infty} \mathbf{Pr}\,(X_n \leq y | X_0 = x_0) = F(y), \ \ \forall y \in \mathbb{R}$$

for any fixed initial state, $x_0 \in \mathbb{R}$. For example, if Q is chosen wisely, the Metropolis-Hastings chain has its proposal density f as its "steady-state density". Therefore, to draw MCMC samples from f, we simply run the Metropolis-Hastings chain long enough to ensure it is burnt-in to steady-state. In any case, if X has a steady-state distribution F with density f, this density satisfies the following "stationarity" property:

$$F(y) = \int_{-\infty}^{\infty} f(x)K(x,y)dx, \ \ \forall y \in \mathbb{R}.$$

If such a density does not exist, the stationarity property of $\mathbf{F}$ can be expressed via a more general integral.

2.4 Discrete space Markov chain Monte Carlo approach

In this section, a brief explanation in the Markov Chain Monte Carlo (MCMC) approach in particular, the *Metropolis algorithm* for sampling from discrete distribution on complex space is discussed. Given a large finite state space $\mathcal{S}$, if we want to select an element i belonging to $\mathcal{S}$, there are commonly two ways to do it as follows:

(I) draw it "uniformly at random"

(II) draw it according to some "complex criterion".

If $\mathcal{S} = \{0, 1, 2, \ldots, k\}$, usually, for (I), pseudo-random number generator (PRNG) can be used; this is only a Monte Carlo approach. It does not need to use the MCMC approach. However, in general, it may be impractical to label the number of state space $|\mathcal{S}| = k+1$ elements by the numbers 0, 1, 2, ..., $k-1$, k. In order to overcome this problem, a MCMC approach is used.

Case I: First, we want to draw a sample uniformly at random from the uniform distribution on $\mathcal{S}$ ("uniformly at random"). We begin with constructing an irreducible, aperiodic MC X with state space $\mathcal{S}$ and uniform steady-state distribution $\hat{\pi} = \frac{1}{|\mathcal{S}|}$ and all $i \in \mathcal{S}$. For example, say, X has symmetric t.p.m., $\mathbf{K}$.

Suppose we start with X having a fixed state, say, i_0, (that is, $X_0 = i_0$) and let X "run" a long time, n. Then, by the definition of steady-state, X_n will be approximately distributed as $\hat{\pi}$. Therefore the state of X at a large time n is approximately a random selection element of $\mathcal{S}$.

Case II: Second, the idea of a general p.m.f., $\hat{\pi}$ on $\mathcal{S}$ is used for selecting the state i. A method called Metropolis algorithm (Metropolis et al., 1953) will be introduced as follows:

Given a "target distribution", $\hat{\pi}$ on $\mathcal{S}$ with $\hat{\pi}_i > 0$, for all $i \in \mathcal{S}$, and given an irreducible, aperiodic MC Y with symmetric t.p.m Q, Y is called the "proposal chain". Define a new MC X recursively as follows:

Given $X_n = i$, the next state X_{n+1} is determined by:

(1) Let Y "propose" the next state, j. For X, (that is, jump from i to j with probability),

$$Q_{i,j} = \mathbf{Pr}(Y_{n+1} = j \,|\, Y_n = i)$$

(2) Accept/reject the proposal, j as follows:

(i) If j is more likely than i under $\hat{\pi}$, (that is, $\hat{\pi}_j \geq \hat{\pi}_i$), then X accepts the proposal ($X_{n+1} = j$).

(ii) If $\hat{\pi}_j < \hat{\pi}_i$, X accepts the proposal j with probability, $\frac{\hat{\pi}_j}{\hat{\pi}_i} < 1$. $(X_{n+1} = j)$. Otherwise, reject j, $(X_{n+1} = X_n)$.

In total, X accepts proposal j with probability

$$\alpha_{i,j} = \min\left\{1, \frac{\hat{\pi}_j}{\hat{\pi}_i}\right\}$$

Usually, this α is called the acceptance probability (Madras, 2002).

Note that the t.p.m. of the Metropolis chain X is

$$K_{i,j} = Q_{i,j} \cdot \alpha_{i,j}, \text{ for all } i \neq j \in \mathcal{S}$$

and

$$K_{i,i} = Q_{i,i} + \sum_{i \neq j} Q_{i,j}[1 - \alpha_{i,j}], \text{ for all } i \in \mathcal{S}.$$

In fact, the Metropolis algorithm can be applied to performing a "stochastic search" for the global minimum/maximum of an objective function $V : \mathcal{S} \to R$. In particular, if we are looking for the global maximum of V where $V(i) > 0$ for all $i \in \mathcal{S}$, an associated p.m.f.

$$\hat{\pi}_i^{\beta} = \frac{1}{C_\beta} \exp[\beta V(i)], \text{for } i \in \mathcal{S}$$

is defined from the distribution $\hat{\pi}^{\beta}$ where $C_\beta = \sum_{i \in \mathcal{S}} \exp[\beta V(i)]$ is the unknown normalizing constant and β is a fixed constant greater than zero. In this way, if we run the Metropolis chain, X for a sufficiently long time, so that $X_n \sim \hat{\pi}^{\beta}$. Since $\hat{\pi}^{\beta}$ is dominated for i with $V(i)$ large, we will, with high probability (depending on the value of β) be selecting $i \in \mathcal{S}$ with large values under $V(i)$. This effect is increased for large values of $\beta > 0$, as illustrated in figure 2.2.

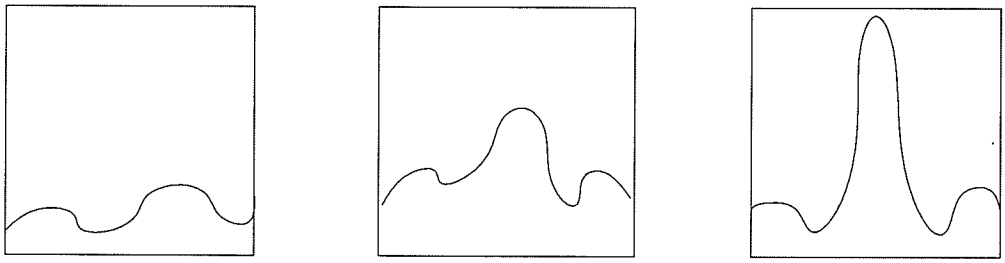

Figure 2.2: $\hat{\pi}_i^{\beta}$ with different beta values

2.5 General simulated annealing method

Simulated annealing method is used to improve the Metropolis algorithm so that a optimization solution can be achieved. Before we applied our multiple sequence alignment problem in chapter 4, a general idea of simulated annealing method is introduced as follows.

Let $\mathcal{S}$ be a set of all feasible solutions. Let V be a given value function on $\mathcal{S}$. If we want to find a point k of $\mathcal{S}$, for which $V(k)$ is maximized or close to its maximum value, we can increase the value of β slowly over the iteration time n.

Now, let Q be a irreducible, symmetric transition probability matrix on $\mathcal{S}$ and let $\beta(n)$ be the function of n by increasing the value slowly. Our new acceptance probability will be:

$$\alpha = \min\left\{1, \frac{\exp\left[\beta(n)\ V(j)\right]}{\exp\left[\beta(n)\ V(i)\right]}\right\}$$

A similar Metropolis chain is run as in the previous section except we will have a new β changing in iteration n. Note that, the Markov Chain is no longer time-homogeneous, since the transition probabilities change over time. If we set $\beta(n)$ to be a fixed constant, then the acceptance probability α will be same as the one in

previous section and this chain will be time-homogeneous. Finally, we run this chain for a long time and stop until the MC X seems to stop changing. Then the final iteration X will be an approximate optimal solution (Madras, 2002). An example of using simulated annealing method via Metropolis algorithm will be demonstrated in Chapter 4. In that chapter, Metropolis algorithm will be applied to find the optimal sequence alignment.

Chapter 3

MCMC Algorithm For Phylogenetic Tree Construction

In this chapter, a Bayesian method using Markov chain simulation is discussed to find out the phylogenetic relationship between taxa in given multiple DNA sequences based on Li et al.'s paper. A new phylogenetic method using the Markov chain Monte Carlo (MCMC) technique was developed by Li et al. (2000). Metropolis algorithm was used in the Markov chain. The reader must be careful because we are not trying to develop a new algorithm in this chapter. We just explain some of the biology applications using statistical method. However, some gaps in Li et al.'s paper will be filled so that it can help us to understand easier about the stochastic process applying in different area of the study.

3.1 The probability distribution of the phylogenetic tree

A five-parameter nucleotide substitution model is used and proposed by Hasegawa, Kishino and Yano (1985). This model allows the rates of transitional events (mutations within purines and pyrimidines) and the rates of transversional events (mutations between purines and pyrimidines) being different, and, the distribution of nucleotides is stationary (Li et al., 2000).

3.1.1 The notation of phylogenetic trees

To avoid confusion, we will use some similar notations as in the Li et al. (2000). In this chapter, only rooted trees will be considered in the model and it assumes that the rate of evolution over different parts of the tree are constant. We will divide three important components in phylogenetic tree as follows and illustration of the notations will show in fig 3.1:

1. Branching pattern (Topology)

   In rooted tree, if n taxa are given, then the total number of distinct labeled topologies will be $\prod_{i=1}^{n-1}(2i-1)$ (Edwards and Cavalli-Sforza, 1964). Let $\mathcal{T}_n$ be the set of all possible topologies with n external nodes.

2. Nodes

   n external nodes is labeled as $1, 2, ..., n$. Then, the root node is labeled as -1, follow by, the remaining $n-2$ internal nodes as $-2, -3, ..., -(n-1)$. Consider that each node inside the tree is related to a DNA sequence with length of m nucleotides long. Let the vector of DNA sequences of the external nodes be $\vec{b} = (b_1, b_2, ..., b_n)$ and the vector of DNA sequences of the internal nodes, excluding the root node, be $\vec{v} = (v_{-2}, ..., v_{-(n-1)})$. Furthermore, let $\mathcal{D}$ be the set of four nucleotides, A, C, G and T. Hence, b_i will be in $\mathcal{D}^m$ for $i = 1, 2, ..., n$ and v_{-j} will be in $\mathcal{D}^m$ for $j = 1, 2, ..., n-1$ (Li at al., 2000).

3. Branch lengths

   Assume that the time from the root node to each of the external node is a constant, τ. Let the vector of evolutionary splitting times at the internal nodes

$-2, -3, ..., -(n-1)$ be $\vec{t} = (t_{-2}, t_{-3}, ..., t_{-(n-1)})$. Obviously, t_{-k} must be less than τ and the vector $\vec{t}$ must be in the range of $(0, \tau)^{n-2}$. For simplification, we set the times of the external nodes be zero. Note that, in rooted tree, there is a unique path from the root node to each external node and the direction of these path is associated with the evolutionary time. Each of the internal node points out an evolutionary splitting. Finally, the time difference between two nodes connected with the branch is called branch length (Li at el., 2000).

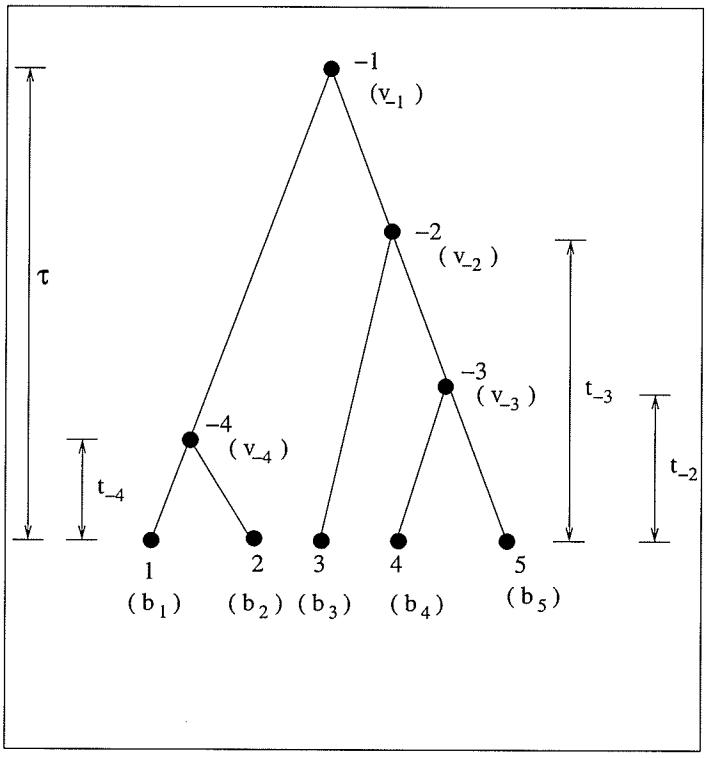

Figure 3.1: Notations of external, internal nodes and splitting times in one of the phylogenetic tree with 5 taxa.

To sum up, let Ω_n be a set of five elements $(T, \vec{t}, \vec{v}, v_{-1}, \vec{b})$ in a phylogenetic tree, where a topology $\vec{T} \in \mathcal{T}_n$, and $\vec{t} \in (0, \tau)^{n-2}$ with the order restrictions forced by T, $\vec{v} \in \mathcal{D}^{m(n-2)}$, $v_{-1} \in \mathcal{D}^m$, and $\vec{b} \in \mathcal{D}^{mn}$. We should always keep in mind that this Markov model follows an independently identical distributed processes along each branch of the tree. Moreover, paths between the root and the internal nodes split and follow independent processes (Li at el., 2000).

3.1.2 Conditional distribution given the observed DNA sequences

We continue by describing the so-called *five-parameter model.* Let $X(t)$ denote the "state" of a site on a DNA sequence at time $t \geq 0$. The possible states of $X(t)$ are A, C, G, T. Hence, the stochastic process, $X = \{X(t) | \, t \geq 0\}$ models the time evolution (that is, mutation over time) at a single site on a given DNA sequence. Serval Markov models for X appear in the literature (see Larget and Simon (1999), Mau et al. (1999) and including the five-parameter model proposed by Hasegawa, Kishino and Yano (1985) and some references from Li at el.). Let $\alpha, \kappa > 0$, and let π_A, π_C, π_G, and π_T are nonnegative numbers which sum to 1. As defined in Li at el. (2000), a "generator matrix" $\mathbf{R}$, where

$$\frac{\mathbf{R}}{\alpha} = \begin{pmatrix} \rho_A & \pi_T & \kappa\pi_G & \pi_T \\ \pi_A & \rho_C & \pi_G & \kappa\pi_T \\ \kappa\pi_A & \pi_C & \rho_G & \pi_T \\ \pi_A & \kappa\pi_C & \pi_G & \rho_T \end{pmatrix}$$

where the rows and columns of $\frac{R}{\alpha}$ are labeled A, C, G, T (from top to bottom and from left to right, respectively), and

$$\begin{cases} \rho_A = -(\kappa\pi_G + \pi_C + \pi_T), \\ \rho_C = -(\kappa\pi_T + \pi_A + \pi_G), \\ \rho_G = -(\kappa\pi_A + \pi_C + \pi_T), \\ \rho_T = -(\kappa\pi_C + \pi_A + \pi_G). \end{cases}$$

By a quick check, we can see each row sum in this matrix $\mathbf{R}$ is equal to zero which satisfies our definition 2.2.3. By using Kolmogorov backward equations, the transition probabilities matrix $\mathbf{K}(t)= (K_{i,j}(t))$ at time $t > 0$ of our Markov chain can be found. As stated in Li et al. (2000), let $\lambda_j = \pi_A + \pi_G$ if base j is a purine (A or G) and let $\lambda_j = \pi_C + \pi_G$ if base j is a pyrimidine (C or T), and let $\gamma_j = 1 + (\kappa - 1)\,\lambda_j$. Then the transition probability matrix will be:

$$K_{i,j}(t) = \begin{cases} \pi_j + \pi_j\,(\frac{1}{\lambda_j} - 1)\,e^{-\alpha t} + (\frac{\lambda_j - \pi_j}{\lambda_j})\,e^{-\alpha\gamma_j t} & i = j \\ \pi_j + \pi_j\,(\frac{1}{\lambda_j} - 1)\,e^{-\alpha t} - (\frac{\pi_j}{\lambda_j})\,e^{-\alpha\gamma_j t} & i \neq j \\ \pi_j\,(1 - e^{-\alpha t}) & i \neq j, \end{cases}$$

where the second line is the transitional event and third line is the transversional event.

Remark 3.1.1 Letting $t \to \infty$ in $K_{i,j}(t)$, (π_A, π_C, π_G, π_T) are clearly the steady-state distribution of X. The parameter α is called the overall transversion rate and the parameter κ models the relative bias toward the occurrence of transitional (mutations within purines and pyrimidines) events over tranversional (mutations between purines and pyrimidines) events (Li at el., 2000).

To define a formula for the distribution of a phylogenetic tree on Ω_n, given observed DNA sequences, $\vec{b}$, we need to impose three natural assumptions on the distribution of components of a phylogenetic tree (i.e., the five elements of phylogenetic tree $(T, \vec{t}, \vec{v}, v_{-1}, \vec{b})$). We take these directly from Li at el. (2000).

Assumption 3.1.2 The unconditional distribution of the topology T with n external nodes is uniformly distributed over all possible topologies in $\mathcal{T}_n$.

Assumption 3.1.3 If a topology T is given, the distribution of splitting times of all the internal nodes is uniform distribution over the space defined by the order restrictions imposed by T.

Assumption 3.1.4 In the root node, the unconditional distribution of nucleotides at each site is the stationary distribution ($\pi_A, \pi_C, \pi_G, \pi_T$). Also, nucleotides at different sites are changing independently followed by the t.p.m. $\mathbf{K}(t)$. Finally, the evolutionary processes along different branches are independent.

The probability distribution on the set of trees with five important elements $(T, \vec{t}, \vec{v}, v_{-1}, \vec{b})$ can be expressed. Under the above assumption, the product of three conditional and unconditional probabilities can be written as:

$$\mathbf{Pr}\,(T, \vec{t}, \vec{v}, v_{-1}, \vec{b}) = \mathbf{Pr}\,(\vec{v}, v_{-1}, \vec{b} | \vec{t}, T) \cdot \mathbf{Pr}\,(\vec{t} | T) \cdot \mathbf{Pr}\,(T),$$

where $\vec{t}$ is a splitting time for the internal nodes which satisfies the order restrictions imposed by the topology $\mathcal{T}$. In order to make things more clear, formula can be developed for the three factors in the distribution of $(T, \vec{t}, \vec{v}, v_{-1}, \vec{b})$ in turn,

(i) (uniform prior on topologies)

Using assumption 3.1.2, the topology T is uniformly distributed over all possible topologies $\mathcal{T}$. It is easy to show that

$$\mathbf{Pr}(T) = \frac{1}{\prod_{i=1}^{n-1}(2i-1)},$$

since, as shown in subsection 3.1.1, the total number of distinct labeled topologies of rooted trees with n external nodes is $\prod_{i=1}^{n-1}(2i-1)$.

(ii) (nucleotide substitution model)

Assumption 3.1.4 allows us to find probability of $\mathbf{Pr}(\vec{v}, v_{-1}, \vec{b} | \vec{t}, T)$. Consider a parent sequence v_p at node p and one of its child sequences v_c at node c, they are separated by a branch with length $t = t_p - t_c$. Totally, there are 16 different (i,j) nucleotide pairs under the m sequence long sites which are (A,A), (A,C), (A,G), (A,T), (C,C), (C,A), (C,G), (C,T), (G,G), (G,A), (G,C), (G,T), (T,T), (T,A), (T,C), (T,G). Let the parent sequence v_p at node p be $v_p = (i_1, i_2, ..., i_m)$ and its child sequence v_c at node c be $v_c = (j_1, j_2, ..., j_m)$. Hence, (i_n, j_n) are the corresponding nucleotide pairs in site n where $n = 1, 2, ..., m$ and $(i_n, j_n) \in \mathcal{D}^2$ (Li et al., 2000). Let the counts be:

$m_{i,j,p,c}$ = number of sites with nucleotide i in node v_p and nucleotide j in node v_c.

Example 3.1.5 Suppose,

$$v_p = AGGCTGAG$$

$$v_c = AGAGTCGG$$

Then, it is easy to see that $m_{G,G,p,c} = 2$, $m_{G,A,p,c} = 1$ and $m_{G,C,p,c} = 0$ and so on.

Now, let $\mathbf{Pr}_{v_p,v_c}(t)$ be the probability that the nucleotide sequence at node p evolves to the nucleotide sequence at node c with a period of time $t = t_p - t_c$. Since, the sites evolve independently and site evolve according to the t.p.m. $K_{i,j}(t)$. Therefore,

$$\begin{aligned}
\mathbf{Pr}_{v_p,v_c}(t) &= \mathbf{Pr}(\ v_p \text{ evolves to } v_c \text{ in time t }) \\
&= \prod_{n=1}^{m} \mathbf{Pr}(\text{nucleotide } i \text{ evolves to nucleotide } j \text{ at site } n) \\
&= \prod_{n=1}^{m} K_{i_n,j_n}(t)
\end{aligned}$$

Since there are only 16 categories of nucleotide pairs, then,

$$\mathbf{Pr}_{v_p,v_c}(t) = \prod_{i,j\in\mathcal{D}} K_{i,j}(t)^{m_{i,j,p,c}}$$

where $\sum_{\forall i,j\in\mathcal{D}} m_{i,j,p,c} = m$.

Now, the nucleotide sequence of the common parent node will mutate into some different nucleotide sequences of its children. Furthermore, nucleotide sequences at each site mutate according to the independent assumption. Hence, given the root node v_{-1}, the topology $T \in \mathcal{T}_n$, and times $\vec{t}$, let r_j be the number of sites in the root node for which the nucleotide is j, $j = A, C, G, T$, the probability of $\mathbf{Pr}(\vec{v}, v_{-1}, \vec{b} | \vec{t}, T)$ can be found by the steady-state distribution assumption. The reason is the probability of a tree is proportional to the product of probabilities nucleotide mutations from the root to each child and

from its children to its grandchildren and so on, until the external nodes are reached (Li et al. 2000). Hence, as stated in Li et al. (2000), the probability is

$$\mathbf{Pr}\left(\vec{v}, v_{-1}, \vec{b} \middle| T, \vec{t}\right) = \left(\prod_{j \in \mathcal{D}} \pi_j^{r_j}\right) \prod_{(c,p) \in E_T} \mathbf{Pr}_{v_p, v_c}\left(t_c - t_k\right)$$

where E_T is the set of all branches (or edges) of topology T.

The first term on the right hand side of the above equation is the stationary probability of the root node DNA sequence using assumption 3.1.4. Let r_j be the total number of sites with nucleotide j, where $j \in \{A, C, G, T\}$, and $\sum_{j \in \mathcal{D}} r_j = m$. Then the product of the stationary probability π_j in each site is

$$\prod_{\forall j \in \mathcal{D}^m} \pi_j = \prod_{j \in \mathcal{D}} \pi_j^{r_j}$$

which is the probability of the first term in the above equation. The second term on the right hand side of the above equation is the probability of set of "parent-child" relationship branches that the nucleotide sequence at node p evolves to the nucleotide sequence at node c over a period of time $t_p - t_c$. Hence, given a topology T, and a set of branches (p, c) called E_T. Using the result of $\mathbf{Pr}_{v_p, v_c}(t_p - t_c)$, we get $\prod_{(p,c) \in E_T} \mathbf{Pr}_{v_p, v_c}(t_p - t_c)$.

(iii) (uniform prior splitting time)

Assumption 3.1.3 is used to find $\mathbf{Pr}(\vec{t}|T)$. If the topology T (branching pattern) is given, it means the ancestor-offspring relationships are defined. The time of nodes are restricted by these relationships which means the time of a parent node is greater than the time of a child node all the time. The general formula of finding $\mathbf{Pr}(\vec{t}|T)$ is stated in Li et al. (2000).

Given a topology $T \in \mathcal{T}_n$ and $n \geq 2$, the probability density of the splitting time $\vec{t} = (t_{-2}, t_{-3}, ..., t_{-(n-1)})$ is

$$f(\vec{t}|T) = \frac{1}{\tau^{n-1}} \prod_{i=2}^{n-1} (n_{-i} + 1)$$

where the order of $\vec{t}$ is restricted by T, and n_{-i} is the number of descendant internal nodes of nodes $-i$, for $i = 2, 3, ..., n-1$ (Li et al., 2000).

Now combining the probabilities in (i), (ii), (iii), the probability distribution of the entire phylogenetic tree with $(T, \vec{t}, \vec{v}, v_{-1}, \vec{b}) \in \Omega_n$ is stated in Li et al., 2000 and is showing as follows:

$$\mathbf{Pr}(T, \vec{t}, \vec{v}, v_{-1}, \vec{b}) = \frac{\prod_{j \in \mathcal{D}} \pi_j^{r_j}}{\prod_{i=1}^{n-1}(2i-1)} \int_{\vec{t}} \prod_{(p,c) \in E_T} \mathbf{Pr}_{v_p, v_c}(t_p - t_c) f(\vec{t}|T)\, d\vec{t}$$

It is important to note that only v's are used in the probabilities equation, if necessary, v's can be replaced by external node b's. Furthermore, there are many different forms of probability distribution of the phylogenetic tree in Literature. The above probability distribution is only one of them and is using in Li et al. (2000).

Once, the probability distribution of the entire phylogenetic tree is defined. We are more interested in the conditional density of the tree given the observed DNA sequences which are the nucleotide sequences $\vec{b} = (b_1, b_2, ..., b_n) \in \mathcal{D}^{mn}$. The posterior density of the tree given the observed DNA sequences is

$$\mathbf{Pr}(T, \vec{t}, \vec{v}, v_{-1}|\vec{b}) = \frac{p(T, \vec{t}, \vec{v}, v_{-1}, \vec{b})}{\sum_{T \in \mathcal{T}_n} \int_{\vec{t} \in (0,\tau)^{n-2}} \sum_{(\vec{v}, v_{-1}) \in \mathcal{D}^{m(n-1)}} p(T, \vec{t}, \vec{v}, v_{-1}, \vec{b}) d\vec{t}}$$

where p is the density associated with $\mathbf{Pr}(T, \vec{t}, \vec{v}, v_{-1}, \vec{b})$ (Li et al., 2000). It is inefficient to calculate the denominator in the above equation, because it involves

with some high dimensional integrals and the calculation increases rapidly when the number of external nodes, n increases. However, this denominator is only the normalizing constant (Li et al., 2000). A brilliant algorithm called Metropolis algorithm can be used to avoid the long computational time of finding this normalizing constant. Metropolis algorithm can be used to construct a class of Markov chains with stationary distribution known only up to the normalizing constant.

3.2 Metropolis sampler on phylogenetic trees

Metropolis algorithm is used to construct a class of Markov chains. This algorithm allows us to find the desired phylogenetic tree with high probability. Samples under the conditional density of the tree given the observed DNA sequences are drawn. Running this algorithm for a long period of time will burn-in to steady state distribution.

If we want to estimate a distribution Π, first, assume Π has a density π with measure μ. Then let $Q(\cdot,\cdot)$ be a transition probability function, with density $q(\cdot,\cdot)$ with respect to measure μ. If x is a current state, a proposed state y is drawn according to $Q(x,\cdot)$. State y is accepted with probability

$$\alpha(x,y) = \min\left\{\frac{\pi(y)\,q(y,x)}{\pi(x)\,q(x,y)},\ 1\right\}.$$

State x will remain unchanged with remaining probability (Li et al., 2000).

To calculate $\alpha(x,y)$, we only need to know density π up to the normalizing constant. π is the conditional density of the tree given the observed DNA sequences (that is, $p\,(T,\vec{t},\vec{v},v_{-1}|\,\vec{b})$ in subsection 3.1.2). Tree is generated from the target distribution $\pi = \sum_{v_{-1}\in\mathcal{D}^m} p\,(T,\vec{t},\vec{v},v_{-1}|\,\vec{b})$ on $(T,\vec{t},\vec{v})$ using Markov chain.

A Metropolis algorithm based on a modification of the local rearrangement strategy is used by the Li at al. (2000). This strategy requires only changing the neighborhood of a specific internal node excluding the root node. Let us consider this internal node or target node (T) with a parent node (P), a sibling node (S), and two of its children nodes ($C1, C2$). A new topology is formed by switching one of the neighborhood nodes from the target nodes to another. For example, we switch $C1$ to S so $C1$ will become a new sibling S' and S will become a new child $C1'$ in our new topology; see figure 3.2 (b). This modification of local rearrangement method (Li at el., 2000) can move any topology to any topology in a finite number of steps and is illustrated in figure 3.2.

In other word, the local rearrangement method can be shown the property of irreducibility. Let x be the state of any branching pattern with n external nodes (i.e., $x \in \mathcal{T}_n$). x can be shown to transform to $y \in \mathcal{T}_n$ in a finite number of steps. The following lemma is stated at Li at el., 2000's appendix and some modifications are done by us:

Lemma 3.2.1 *The local rearrangement method is irreducible.*

Proof.

To show if given any tree T with n external nodes in $\mathcal{T}_n$, Tree T can transform to any other tree in a finite number of steps. Consider $n = 3$, there are only three topologies. It is obviously to see that one can transform to the other in one step in the local rearrangement method, so assume n is larger than 3. In order to prove this lemma, mathematical induction is used by assuming the statement is true for any tree of size less than n.

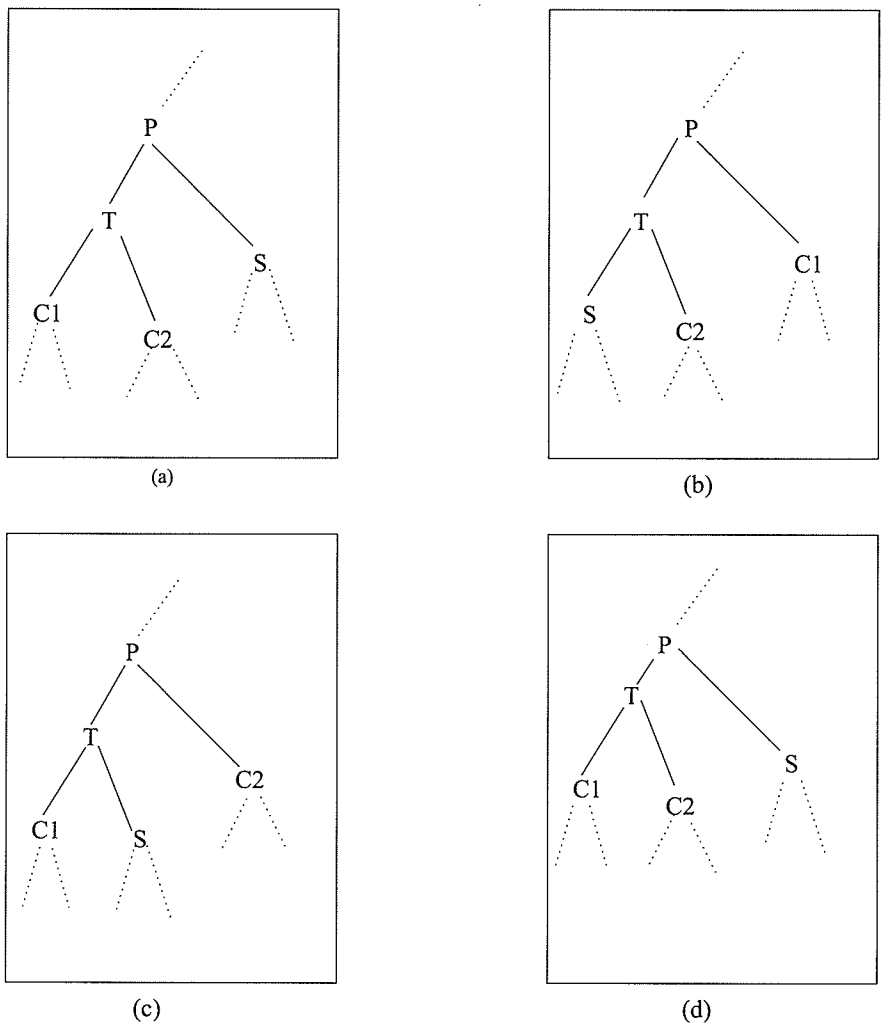

Figure 3.2: The local rearrangement method. Tree (a) is the current tree before the local rearrangement. Trees (b), (c) and (d) are three of the possible trees after the local rearrangement. The nucleotide sequence and the time of the target node, T will change in all the cases after the local rearrangement.

Suppose a initial tree is given from $\mathcal{T}_n$, this tree can be separated into two sets of branching patterns A and B with both size less than n using the local rearrangement method; see figure 3.3 (a). The set of all nodes with size n can be denoted as $A \cup B$. Tree $A \cup B$ can transform into another tree, let say, tree $C \cup D$ in a finite number of steps. This lemma can be proved by showing $A = C$ or $A = D$ at the end of the modification.

If $A = C$ or $A = D$ is started at the beginning, each of the two subtrees can be rearranged into the desired tree by induction hypothesis. Hence $A \neq C$ and $A \neq D$ are assumed. Let M and N be two nodes just under the root node, and start with a initial tree. By the induction hypothesis, the set A can be partition into two subpartition sets $A \cap C$ and $A \cap D$. Similarly, the set B can be partition into two subpartition sets $B \cap C$ and $B \cap D$; see figure 3.3 (b). The size of the all subpartition set must be less than n. Now, using the local rearrangement method, set N be a target node, $B \cap C$ and M can be interchanged, so a new tree with $B \cap C$ be our new sibling of N and M be our new child of N is formed; see figure 3.3 (c). The second rearrangement will be setting M be our new target node and interchange $A \cap C$ and $B \cap D$. Another new tree are formed; see figure 3.3 (d). The third rearrangement will be setting N be the new target node and interchange M and $B \cap C$, the new tree will be look like figure 3.3 (e). Using the induction hypothesis, $B \cap D$ and $A \cap D$ can be rearranged into a set D. Also, $B \cap C$ and $A \cap C$ can be rearranged into a set C; see figure 3.3 (f). By the mathematical induction, the statement is true for all tree with size n. That is, starting with any initial tree, it can transform to any other tree in a finite number of steps.

It is shown that the local rearrangement method is irreducible. The Markov chain

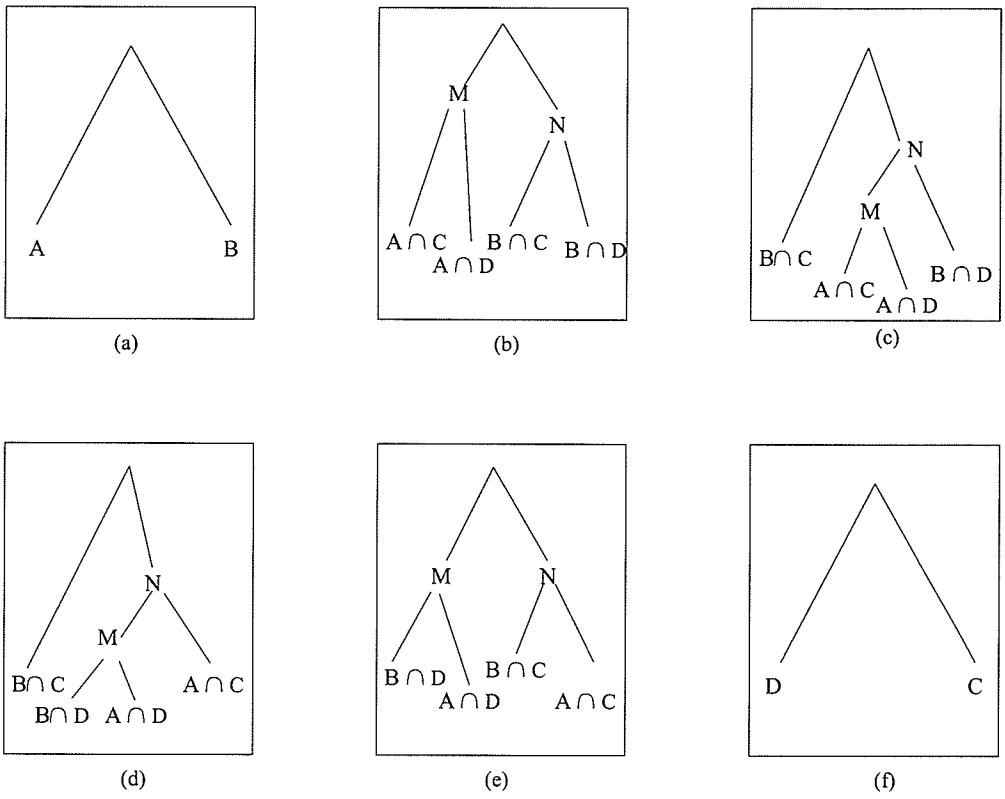

Figure 3.3: Irreducible of the tree

can be divided into three parts as follows:

Part I: (Uniform picking a internal node)

Given a set of internal nodes namely $\{-2, -3, ..., -(n-1)\}$, one of the internal node is randomly picked by using simply uniform Monte Carlo method. For example, a integer k can be generated uniformly at random from a set of integers $\{2, 3, ..., -(n-1)\}$. It means $-k$ is the internal node being picked. Then the neighborhood nodes of $-k$ is considered, which its sibling node, S, and its two children nodes, $C1$ and $C2$. Next, one of these three neighborhood nodes is picked uniformly at random and set it to be our new sibling node, S',

leave the remaining two be our new children nodes, $C1'$ and $C2'$. Now a new branching pattern is formed. There are three possible branching patterns. One of them is same as the former pattern. Hence, the probability of this uniform picking method is simply $\frac{1}{(n-2)3}$, since we will have $n-2$ internal nodes to choose and will have 3 possible branching patterns being formed (Li at el., 2000).

Part II: (Picking the time of the new target node)

To find the time, $t_{T'}$ of the new target node T', the knowledge of part (I) is required, because in order to pick the time, $t_{T'}$, we need to know the time, $t_{C1'}$ and $t_{C2'}$ of the new children nodes $C1'$, $C2'$, also, the time, t_P of the parent node, P in our density function. Obviously, $t_{T'}$ falls between t_P and $\max\{t_{C1'}, t_{C2'}\}$ (That is, $t_P \geq t_{T'} \geq \max\{t_{C1'}, t_{C2'}\}$). Now, $t_{T'}$ is picked according to density

$$g(t_{T'}) = \frac{\delta_{t_{T'}}}{\sigma_{t_{T'}}}$$

where

$$\delta_{t_{T'}} = \sum_{v \in \mathcal{D}^m} \mathbf{Pr}_{v_P,v}(t_P - t_{T'})\, \mathbf{Pr}_{v,v_{C1'}}(t_{T'} - t_{C1'})\, \mathbf{Pr}_{v,v_{C2'}}(t_{T'} - t_{C2'})$$

and

$$\sigma_{t_{T'}} = \int_{\max\{t_{C1'},t_{C2'}\}}^{t_P} \sum_{v \in \mathcal{D}^m} \mathbf{Pr}_{v_P,v}(t_P - t)\mathbf{Pr}_{v,v_{C1'}}(t - t_{C1'})\mathbf{Pr}_{v,v_{C2'}}(t - t_{C2'})dt.$$

This density is stated in Li. at el., 2000. It is important to remember that $\mathbf{Pr}_{v_i,v_j}(t_i - t_j)$ is the probability that nucleotide sequence v_i evolves to nucleotide sequence v_j in a period of time $t_i - t_j$ defined in subsection 3.1.2.

Part III: (Picking the nucleotide sequence of the new target node)

After the new time $t_{T'}$ is defined in part (II), we would like to find the new nucleotide sequence, $v_{T'}$ of T'. This new nucleotide sequence, $v_{T'}$ is chosen according to density

$$h(v_{T'}) = \frac{\phi_{v_{T'}}}{\varphi_{v_{T'}}}$$

where

$$\phi_{v_{T'}} = \mathbf{Pr}_{v_p,v_{T'}}(t_p - t_{T'})\,\mathbf{Pr}_{v_{T'},v_{C1'}}(t_{T'} - t_{C1'})\,\mathbf{Pr}_{v_{T'},v_{C2'}}(t_{T'} - t_{C2'})$$

and

$$\varphi_{v_{T'}} = \sum_{v\in\mathcal{D}^m} \mathbf{Pr}_{v_p,v}(t_p - t_{T'})\,\mathbf{Pr}_{v,v_{C1'}}(t_{T'} - t_{C1'})\,\mathbf{Pr}_{v,v_{C2'}}(t_{T'} - t_{C2'}).$$

This density is defined in Li at el., 2000. Note that the expression of the numerator, $\delta_{t_{T'}}$ in part (II) is exactly the same as the expression of the denominator, $\varphi_{v_{T'}}$ in this part.

Combining the probabilities and densities in part (I), part (II) and part (III), $q(\cdot,\cdot)$, $\alpha(\cdot,\cdot)$ in the Metropolis chain can be found. First, assume x be the current state and y be the proposed state after the local rearrangement method describe before in the chain, the transition probability density will be $k(x,y) = \alpha(x,y)\,q(x,y)$, where

$$\begin{aligned} q(x,y) &= \frac{1}{(n-2)3}\,g(t_{T'})\,h(v_{T'}) \\ &= \frac{1}{(n-2)3}\,\frac{\delta_{t_{T'}}}{\sigma_{t_{T'}}}\,\frac{\phi_{v_{T'}}}{\varphi_{v_{T'}}} \\ &= \frac{1}{(n-2)3}\,\frac{\phi_{v_{T'}}}{\sigma_{t_{T'}}}. \end{aligned}$$

with

$$\phi_{v_{T'}} = \mathbf{Pr}_{v_p,v_{T'}}(t_p - t_{T'})\,\mathbf{Pr}_{v_{T'},v_{C1'}}(t_{T'} - t_{C1'})\,\mathbf{Pr}_{v_{T'},v_{C2'}}(t_{T'} - t_{C2'})$$

and

$$\sigma_{t_{T'}} = \int_{\max\{t_{C1'},t_{C2'}\}}^{t_P} \sum_{v\in\mathcal{D}^m} \mathbf{Pr}_{v_P,v}(t_P - t)\mathbf{Pr}_{v,v_{C1'}}(t - t_{C1'})\mathbf{Pr}_{v,v_{C2'}}(t - t_{C2'})dt,$$

also,

$$\begin{aligned}\alpha(x,y) &= \min\left\{\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}, 1\right\} \\ &= \min\left\{\frac{\Lambda}{\Phi}\frac{\mathbf{Pr}_{v_P,v_{S'}}(t_P - t_{S'})(n_{T'}+1)}{\mathbf{Pr}_{v_P,v_S}(t_P - t_S)(n_T+1)}, 1\right\}\end{aligned}$$

with

$$\Lambda = \int_{\max\{t_{C1'},t_{C2'}\}}^{t_P} \sum_{v\in\mathcal{D}^m} \mathbf{Pr}_{v_P,v}(t_P - t)\mathbf{Pr}_{v,v_{C1'}}(t - t_{C1'})\mathbf{Pr}_{v,v_{C2'}}(t - t_{C2'})dt$$

and

$$\Phi = \int_{\max\{t_{C1'},t_{C2'}\}}^{t_P} \sum_{v\in\mathcal{D}^m} \mathbf{Pr}_{v_P,v}(t_P - t)\mathbf{Pr}_{v,v_{C1}}(t - t_{C1})\mathbf{Pr}_{v,v_{C2}}(t - t_{C2})dt$$

The above expressions are defined in Li at el., 2000. Note that n_T is the number of descendant internal nodes of target node T. The expression of $\alpha(x,y)$ is stated without proof by Li et al. (2000). This expression involves some cancelations of numerator $\pi(y)q(y,x)$ and denominator $\pi(x)q(x,y)$ in $\alpha(x,y)$. Furthermore, in the local rearrangement method, the root node cannot be picked as the target node. If the root node is the parent of the target node, the new sibling node can be set to be the parent node and replace v_P by $v_{S'}$ and t_P by $2\tau - t_{S'}$ (Li et al., 2000).

3.3 Discussion

As mentioned before, we only explain the statistical method using in Li et al.'s paper. We ignore to discuss how to estimate the parameters such as the overall substitution rate α, the transition/transversion bias κ, and the stationary probabilities π_A, π_C, π_G and π_T and so on. The reader can find the detail of estimating those parameters in Li et al. (2000). Furthermore, we ignore the simulation part of this model because we do not want to repeat the procedures done by Li et al. (2000)'s paper.

To sum up, the Metropolis algorithm that was used is very fast and it has a good balance between calculation speed and mixing rate of the local rearrangement strategy. Phylogenetic tree construction using Markov chain Monte Carlo method is a huge topic in literature. The model that we explain is only one of them.

Chapter 4

MCMC Algorithm For Sequence Alignment

Sequence alignment is one of the important subjects in molecular biology. There are many different algorithms and methods to find a reasonable or optimal sequence alignment in literature. This chapter will introduce a algorithm using Markov chain Monte Carlo with Metropolis method.

4.1 Uniform sampler of sequence alignments using MC

Suppose we want to draw an uniform sample from the set of all possible sequence alignment using Markov chain, first assume the state-space of all possible alignment be $\mathcal{S}$, so the number of states will be $|\mathcal{S}|$. Let X be a MC with state-space $\mathcal{S}$. A MC X can be constructed, such that X has a uniform steady-state distribution. If this chain can be run for a large number of time, say n, then X_n will be approximately one of a sequence alignment selected uniformly at random from the state space $\mathcal{S}$.

In the next few sections, we will introduce how to set-up an appropriate algorithm and notations in the sequence alignments.

4.1.1 Notation for sequence alignments

Before we apply Markov chain techniques to the sequence alignment problem, we introduce some notations that we are going to use in this section. Some notations here are quite similar to those notations used in H. Carrillo and D. Lipman (1988).

Suppose an alphabet γ of n characters is given:

$$\gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}.$$

If DNA sequence is used, then γ of 4 character is:

$$\gamma = \{A, C, G, T\}.$$

A k-sequence, of this alphabet is element of

$$\gamma^k = \prod_{i=1}^{k} \gamma$$

where $\prod$ represents the cartesian product of sets. Hence, a k-sequence s is a vector of the form

$$s = (\gamma_{n_1}, \ldots, \gamma_{n_k})$$

where for each $j = 1, 2, \ldots, k$, n_j is a natural number that satisfies $1 \leq n_i \leq n$ (Carrillo and Lipman, 1988). A new alphabet α which is obtained from γ by adding the blank or space character " $-$ " is defined:

$$\alpha = \{-\} \cup \{\gamma_1, \gamma_2, \ldots, \gamma_n\}.$$

To simplify notation, assume that γ is the set of the first n natural numbers $\gamma = \{1, 2, \ldots, n\}$ and will denote " $-$ " by 0.

An alignment of N sequences $s_1, s_2, \ldots, s_N$ in γ, not necessary of the same length is another set of sequences, $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_N$ in α all of the same length, such that each sequence $\bar{s}_i$ is obtained by adding spaces in positions of s_i where some but not all of the other sequences have a non-space character.

In particular, suppose the sequences are

$$s_1 = (n_1^1, \ldots, n_{k_1}^1)$$
$$\vdots$$
$$s_N = (n_1^N, \ldots, n_{k_N}^N),$$

then the alignment $(\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_N)$ of $(s_1, s_2, \ldots, s_N)$ is an element of the set

$$\prod_{j=1}^{N} \alpha^l$$

for some l satisfying $\max\{k_1, \ldots, k_N\} \leq l \leq \sum_{i=1}^{N} k_i$, such that, removing all $-$'s from $\bar{s}_i$ yields s_i, for $i = 1, 2, \cdots, N$ (Carrillo and Lipman, 1988). The number l must be between $\max\{k_1, \ldots, k_N\}$ and $\sum_{i=1}^{N} k_i$, because we do not want to have a vertical column with all zeros.

4.1.2 The uniform alignment sampler X

In order to apply the MC algorithm to our particular DNA sequences, an appropriate state space, $\mathcal{S}$ is defined, where $\mathcal{S}$ is the set of all possible sequence alignments. Suppose N DNA sequences $s_1, s_2, \ldots, s_N$ are given, and each sequence has a corresponding length k_j, where $j = 1, 2, \cdots, N$. To obtain an alignment, some spaces / blank elements "$-$" are inserted in each sequence s_i. In the present case, $\bar{s}_i$ is obtained from s_i by inserting spaces so that each s_i is of length $L = \sum_{j=1}^{N} k_j$. For simplification, each alphabet character of DNA can be transformed into a number as follows: *space* $= 0$, $A = 1$, $C = 2$, $G = 3$, $T = 4$.

Any alignment of $(s_1, s_2, \ldots, s_N)$ can be represented by an "alignment matrix"

$B \in \{0,1,2,3,4\}^{N\times L}$ where

$$B = \begin{pmatrix} \bar{s_1} \\ \bar{s_2} \\ \vdots \\ \bar{s_N} \end{pmatrix}_{N\times L}$$

i.e., The i^{th} row of B is the vector $\bar{s_i}$, $i = 1, 2, \ldots, N$.

A MC $X = \{X_0, X_1, \ldots\}$ can be constructed whose state space $\mathcal{S}$ is the collection of all alignments of $(s_1, s_2, \ldots, s_n)$. That is, $\mathcal{S} \subseteq \{0,1,2,3,4\}^{N\times L}$. For the initial sequence alignment, we set $X_0 = B_0$ where the i^{th} row of B_0 is $(n_1^i, n_2^i, \ldots, n_{k_i}^i, 0, \ldots, 0)$ and each row i has a corresponding length of zero which is $L - k_i$. Symbolically, we can express this matrix as

$$B_0 = \begin{pmatrix} s_1 & 0 & 0 & \cdots & 0 \\ s_2 & 0 & 0 & \cdots & 0 \\ s_3 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ s_N & 0 & 0 & \cdots & 0 \end{pmatrix}_{N\times L}$$

The MC X on $\mathcal{S}$ can be defined "recursively" as follows:

If $X_t = B \in \mathcal{S}$ then to determine the state of X at time $t+1$

(a) choose i in $\{1, 2, \cdots, N\}$ uniformly at random (i.e., select one of the rows or equivalently, select one of the sequences)

(b) choose j in $\{1, 2, \cdots, L-1\}$ uniformly at random (i.e., choose j^{th} column or pick one position in sequence i)

(c) Let $B_{i,j}$ denotes the element of i^{th} row and j^{th} column of the matrix B.

(i) If $B_{i,j} \in \{1,2,3,4\}$, then we check $B_{i,j+1}$. If $B_{i,j+1} = 0$ then we interchange the element of $B_{i,j}$ and $B_{i,j+1}$. Otherwise, we do nothing.

(ii) If $B_{i,j} = 0$, we check $B_{i,j+1}$ if $B_{i,j+1} \neq 0$ then we interchange the element of $B_{i,j}$ and $B_{i,j+1}$. Otherwise, we do nothing.

Clearly, X is a Markov Chain with state space $\mathcal{S}$.

4.1.3 Properties of the alignment sampler X

We want to show X defined in subsection 4.1.2 is a irreducible, aperiodic MC with symmetric t.p.m., K, hence has uniform steady-state distribution, and can be used to select an alignment "uniformly at random". We will discuss an MCMC algorithm for selecting optimal alignment in a later section.

Now, we start with the irreducible property. To prove X is irreducible, we want to show any alignment matrix $B \in \mathcal{S}$ can "transform" to B_0 in a finite number of steps; where $B_0 \in \mathcal{S}$ is the initial alignment matrix defined in subsection 4.1.2. We also need to show B_0 can transform back to B in a finite number of steps.

First, let $\bar{s_i}$ be the sequence vector with space element inserted in i^{th} row of matrix B and $\bar{o_i}$ be the sequence vector with space element insert in i^{th} row of the initial matrix B_0, where $i \in \{1,2,\ldots,N\}$. Second, let r_i be the minimal number of steps being used to transform $\bar{s_i}$ to $\bar{o_i}$. Hence, r_i denotes the number of steps needed to move all non-zero entries in $\bar{s_i}$ to the left-hand side of that row.

Example 4.1.1 Suppose we have,

$$s_3 = 010203$$

$$o_3 = 123000$$

To find r_3, we need to count how far the position of the non-zero elements from $\bar{s_3}$ to $\bar{o_3}$; say the element " 3 " in $\bar{s_3}$, we need three steps/jumps from position 6 back to position 3 in $\bar{o_3}$. Hence, in total, $r_3 = 1 + 2 + 3 = 6$.

Now, let $R = \sum_{i=1}^{N} r_i$ be the total number of jumps from matrix B to matrix B_0 and let p_i denote the probability of picking row i with r_i times, and in each such time, selecting precisely the entries which, transform $\bar{s_i}$ to $\bar{o_i}$ for all $i = 1, 2, ..., N$. Then,

$$p_i = \left(\frac{1}{N} \times \frac{1}{L-1}\right)^{r_i}$$

and let $K^R_{C,D}$ be the transition probability matrix (t.p.m.) from state C to D where C, D belong to state space $\mathcal{S}$.

$$\begin{aligned} K^R_{B,B_0} &= \mathbf{Pr}(X_R = B | X_0 = B_0) \\ &\geq \prod_{i=1}^{N} p_i \\ &= \prod_{i=1}^{N} \left(\frac{1}{N} \times \frac{1}{L-1}\right)^{r_i} \\ &= \left(\frac{1}{N} \times \frac{1}{L-1}\right)^{\sum_{r=1}^{N} r_i} \\ &= \left(\frac{1}{N} \times \frac{1}{L-1}\right)^{R} \\ &> 0 \end{aligned}$$

Conversely by the symmetry,

$$K^R_{B_0,B} \geq \left(\frac{1}{N \times (L-1)}\right)^{R} > 0$$

Hence, the MC X is irreducible.

We continue with the aperiodic property here. In order to show the aperiodic property, it is sufficient to show the t.p.m, $K_{B,B} > 0$ for some $B \in \mathcal{S}$. In fact, we will now calculate $K_{B,B}$, any $B \in \mathcal{S}$, since this number measures the speed of our algorithm.

Lemma 4.1.2 *For any* $B \in \mathcal{S}$,

$$K_{B,B} = 1 - \frac{2}{N\,L\,(L-1)} \sum_{i=1}^{N} k_i\,(L - k_i).$$

Furthermore, $K_{B,B} > 0$, *for any* $B \in \mathcal{S}$.

Proof.

$$\begin{aligned} K_{B,B} &= \mathbf{Pr}(B \text{ does not change in one application of X}) \\ &= \sum_{i=1}^{N} \mathbf{Pr}(B \text{ does not change} \,|\, \text{row } i \text{ is picked})(\text{row } i \text{ is picked}) \\ &= \sum_{i=1}^{N} \mathbf{Pr}(B \to B \,|\, s_i)\,\mathbf{Pr}(s_i) \end{aligned}$$

Now, $\mathbf{Pr}(B \to B \,|\, s_i)$ is equal to the probability of a "non-zero" element being picked such that the element behind it is a "non-zero" element or probability of a "zero" element being picked but the element behind it is also a "zero" element. This probabilities is equal to:

$$\begin{aligned} \mathbf{Pr}(B \to B \,|\, s_i) &= 1 - P \\ &= 1 - \frac{2\,k_i\,(L - k_i)}{L\,(L-1)} \\ &= \frac{L\,(L-1) - 2\,k_i\,(L - k_i)}{L\,(L-1)} \end{aligned}$$

where P is the probability of two consecutive elements being picked are different and the proof of this probability is provided in Appendix A. Hence,

$$\begin{aligned} K_{B,B} &= \sum_{i=1}^{N} \left(1 - \frac{2\,k_i\,(L-k_i)}{L\,(L-1)}\right) \frac{1}{N} \\ &= \frac{1}{N} \sum_{i=1}^{N} \left(1 - \frac{2\,k_i\,(L-k_i)}{L\,(L-1)}\right) \\ &= \frac{1}{N} \cdot N - \frac{1}{N} \sum_{i=1}^{N} \frac{2\,k_i\,(L-k_i)}{L\,(L-1)} \\ &= 1 - \frac{2}{N\,L\,(L-1)} \sum_{i=1}^{N} k_i\,(L-k_i) \end{aligned}$$

In order to prove $K_{B,B} > 0$, we only need to show $\frac{2\,k_i\,(L-k_i)}{L\,(L-1)} < 1$ or equivalently, to show $2\,k_i\,(L-k_i) < L\,(L-1)$. This leads to the following lemma:

Lemma 4.1.3 *There exists a i' for which $1 \leq i' \leq N$ such that $2\,k_{i'}\,(L-k_{i'}) < L\,(L-1)$*

Proof.

Recall:

$$L = \sum_{i=1}^{N} k_i \geq k_i \geq 1,\ \forall\, j.$$

If $N > 2$, let i' be such that

$$k_{i'} = \min\left\{k_i : 1 \leq i \leq N\right\},$$

then

$$2k_{i'} \leq k_1 + k_{i'} \leq L$$

since

$$L - k_{i'} \leq L - 1$$

therefore,

$$2k_{i'}(L - k_{i'}) < L(L-1)$$

Let us see a simple example to show the probability that B does not change is greater than zero.

Example 4.1.4 Suppose, 4 sequences are given and each sequence has equal length of size 5. Then according to our notation, we have $k_i = 5$ for all i, $N = 4$ and $L = 20$.

$$\begin{aligned} K_{B,B} &= 1 - \frac{2}{(4)(20)(20-1)} \sum_{i=1}^{4} k_i(20 - n_i) \\ &= 1 - \frac{1}{760} \sum_{i=1}^{4} 5\,(20-5) \\ &= 1 - \frac{300}{760} \\ &= 0.61 \end{aligned}$$

Hence, B has 61 % chance to remain unchanged. Conversely, B will change to any other matrix $C \in \mathcal{S}$ with probability of 0.39 in this particular example.

Finally, we want to show the transition probability matrix K of X is symmetric. First, consider two different matrices B and C drawn from state space $\mathcal{S}$. However, all elements in B and C are the same except the $(i,j)^{th}$ element and $(i,j+1)^{th}$ element. The $(i,j)^{th}$ element in B is the same as the $(i,j+1)^{th}$ element in C. Similarly, the $(i,j+1)^{th}$ element in B is the same as the $(i,j)^{th}$ element in C. Therefore, B and C are only differed on interchanging the $(i,j)^{th}$ element and $(i,j+1)^{th}$ element. (Note: one of the $(i,j)^{th}$ and $(i,j+1)^{th}$ must be a "space" element and one of them must

be a "alphabet" element.) Now,

$$\begin{aligned} K_{B,C} &= \mathbf{Pr}(\text{ two consecutive elements being picked are different}) \\ &= \frac{2\,k_i\,(L-k_i)}{L\,(L-1)} \\ &> 0 \end{aligned}$$

The result in the second last line of the equation is used from the Appendix A. Similarly, it is easy to see that $K_{C,B} = K_{B,C} > 0$. This implies for all B, $C \in \mathcal{S}$, $K_{C,B} = K_{B,C}$.

By the above properties, X must have a uniform steady-state distribution on $\mathcal{S}$.

4.2 Applying Metropolis algorithm in the sequence problem

We have already shown that the alignment sampler X has an uniform steady-state distribution. Hence, we can apply Metropolis algorithm in our sequence alignment problem. Our goal here is to find an alignment matrix B where B has the maximum sum-of-pairs (SP) score. In order to find this alignment matrix $B \in \mathcal{S}$, our approach will be to fix a constant $\beta > 0$ and to try to sample from the distribution $\hat{\pi}^{\beta}$ given by

$$\hat{\pi}_B^{\beta} = \frac{1}{C_\beta} \exp[\beta \cdot score(B)] \;\; (B \in \mathcal{S}),$$

where $C_\beta = \sum_{C \in \mathcal{S}} \exp[\beta \cdot score(C)]$ is the unknown normalizing constant.

To sample from this distribution, we use the Metropolis algorithm whose proposal matrix Q is the transition probability matrix for the chain in our sequence alignment. The resulting Metropolis chain for $\hat{\pi}^{\beta}$ may be described as follows. Given $X_0 = B_0$ where B_0 is the initial sequence matrix described in subsection 4.1.2,

Step I: Start at the initial alignment matrix B_0, after n-steps set $B_n = B_{curr.} \in \mathcal{S}$ where $B_{curr.}$ is the current alignment matrix.

Step II: Select proposal alignment matrix, $B_{prop.} \in \mathcal{S}$ according to the proposal chain described in subsection 4.1.2. (That is, choose a $(i,j)^{th}$ entry in $B_{curr.}$ uniformly at random, interchange the $(i,j)^{th}$ and the $(i,j+1)^{th}$ if one of them is a zero element and one of them is a nonzero element.)

Step III: Calculate acceptance probability

$$\begin{aligned}
\alpha &= \min\left\{1, \frac{\hat{\pi}^{\beta}_{B_{prop.}}}{\hat{\pi}^{\beta}_{B_{curr.}}}\right\} \\
&= \min\left\{1, \frac{\frac{1}{C_\beta}\exp[\beta \cdot score(B_{prop.})]}{\frac{1}{C_\beta}\exp[\beta \cdot score(B_{curr.})]}\right\} \\
&= \min\left\{1, \frac{\exp[\beta \cdot score(B_{prop.})]}{\exp[\beta \cdot score(B_{curr.})]}\right\}
\end{aligned}$$

Step IV: Generate a random number, U uniformly in $[0,1]$ If $U \leq \alpha$, accept $B_{prop.}$ and set $B_{n+1} = B_{prop.}$. If $U > \alpha$, reject $B_{prop.}$ and set $B_{n+1} = B_n = B_{curr.}$.

Step V: If n is large, we quit the procedure. Otherwise, return to step II, replacing n by $n+1$.

4.2.1 The simplified form of the acceptance probability α

When we calculate the acceptance probability

$$\alpha = \min\left\{1, \frac{\exp[\beta \cdot score(B_{prop.})]}{\exp[\beta \cdot score(B_{curr.})]}\right\},$$

it is easy to see that most of the pairwise scores from the proposal matrix $B_{prop.}$ are the same as the pairwise scores from the current matrix $B_{curr.}$. Therefore, we can

cancel out those identical pairwise scores and remain with some different pairwise scores. If $B_{curr.} = B_{prop.}$, then $\alpha = 1$, so we assume $B_{curr.} \neq B_{prop.}$ and it leads to the following lemma.

Lemma 4.2.1 *Suppose we have an alignment matrix B, if we only interchange two elements, $B_{a,b}$ and $B_{a,b+1}$ in B where $B_{a,b}$ is character number located in row a and column b of matrix B with $a = \{1, 2, ..., N\}$ and $b = \{1, 2, ..., L-1\}$, then, let us call this new alignment matrix as B'. For simplification, let s_i be the sequence i or the i^{th} row in the alignment matrix B and s'_i be the sequence i in the alignment matrix B'. We have*

$$\begin{aligned}
\alpha &= \min\left\{1, \frac{\exp[\beta \cdot score(B')]}{\exp[\beta \cdot score(B)]}\right\} \\
&= \min\left\{1, \frac{\exp\{\beta \cdot \sum_{k=1}^{L} \sum_{1\leq i<j\leq N} p\left[s'_i(k), s'_j(k)\right]\}}{\exp\{\beta \cdot \sum_{k=1}^{L} \sum_{1\leq i<j\leq N} p\left[s_i(k), s_j(k)\right]\}}\right\} \\
&= \min\left\{1, \frac{\exp\{\beta \cdot \sum_{k=b}^{b+1} \sum_{1\leq i\leq N, i\neq a} p\left[s'_i(k), s'_a(k)\right]\}}{\exp\{\beta \cdot \sum_{k=b}^{b+1} \sum_{1\leq i\leq N, i\neq a} p\left[s_i(k), s_a(k)\right]\}}\right\}
\end{aligned}$$

where $p\left[s'_i(k), s'_j(k)\right]$ is the pairwise SP score of the $(i,k)^{th}$ entry and $(j,k)^{th}$ entry in matrix B', and $p\left[s_i(k), s_j(k)\right]$ is the pairwise SP score of the $(i,k)^{th}$ entry and $(j,k)^{th}$ entry in matrix B.

Proof.

$$\begin{aligned}
\frac{\exp[\beta \cdot score(B')]}{\exp[\beta \cdot score(B)]} &= \frac{\exp\left\{\beta \sum_{1\leq i<j\leq N} \sum_{k=1}^{L} p\left[s'_i(k), s'_j(k)\right]\right\}}{\exp\left\{\beta \sum_{1\leq i<j\leq N} \sum_{k=1}^{L} p\left[s_i(k), s_j(k)\right]\right\}} \\
&= \frac{\prod_{k=1}^{L} \exp\left\{\beta \sum_{1\leq i<j\leq N} p\left[s'_i(k), s'_j(k)\right]\right\}}{\prod_{k=1}^{L} \exp\left\{\beta \sum_{1\leq i<j\leq N} p\left[s_i(k), s_j(k)\right]\right\}}
\end{aligned}$$

where $\exp\left\{\beta \sum_{1\le i<j\le N} p\left[s'_i(k), s'_j(k)\right]\right\} = \exp\left\{\beta \sum_{1\le i<j\le N} p[s_i(k), s_j(k)]\right\}$, if and only if $k \neq b$ and $k \neq b+1$.

$$= \frac{\prod_{k=b}^{b+1} \exp\left\{\beta \sum_{1\le i<j\le N} p\left[s'_i(k), s'_j(k)\right]\right\}}{\prod_{k=b}^{b+1} \exp\left\{\beta \sum_{1\le i<j\le N} p\left[s_i(k), s_j(k)\right]\right\}}$$

$$= \frac{\prod_{k=b}^{b+1} \prod_{1\le i<j\le N} \exp\left\{\beta\, p\left[s'_i(k), s'_j(k)\right]\right\}}{\prod_{k=b}^{b+1} \prod_{1\le i<j\le N} \exp\left\{\beta\, p\left[s_i(k), s_j(k)\right]\right\}}$$

Also, $\exp\left\{\beta\, p\left[s'_i(k), s'_j(k)\right]\right\} = \exp\left\{\beta\, p\left[s_i(k), s_j(k)\right]\right\}$, if and only if $i,\ j \neq a$.

$$= \frac{\prod_{k=b}^{b+1} \prod_{1\le i<j\le N,\, i,j=a} \exp\left\{\beta\, p\left[s'_i(k), s'_j(k)\right]\right\}}{\prod_{k=b}^{b+1} \prod_{1\le i<i\le N,\, i,j=a} \exp\left\{\beta\ \ p\left[s_i(k), s_j(k)\right]\right\}}$$

$$= \frac{\prod_{k=b}^{b+1} \prod_{1\le i\le N,\, i\neq a} \exp\left\{\beta\, p\left[s'_i(k), s'_a(k)\right]\right\}}{\prod_{k=b}^{b+1} \prod_{1\le i\le N,\, i\neq a} \exp\left\{\beta\, p\left[s_i(k), s_a(k)\right]\right\}}$$

$$= \frac{\exp\left\{\beta \sum_{k=b}^{b+1} \sum_{1\le i\le N,\, i\neq a} p\left[s'_i(k), s'_a(k)\right]\right\}}{\exp\left\{\beta \sum_{k=b}^{b+1} \sum_{1\le i\le N,\, i\neq a} p\left[s_i(k), s_a(k)\right]\right\}}$$

The calculation of the pairwise SP score in the nominator reduces to $2(N-1)$ terms. Similarly, only $2(N-1)$ terms in the denominator have to calculate. Let us see a simple example to illustrate our result.

Example 4.2.2 Suppose we have an alignment matrix B where

$$B = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix},$$

then say we pick element "e" and interchange with element "f" and form a new

matrix B' where

$$B' = \begin{pmatrix} a & b & c \\ d & f & e \\ g & h & i \end{pmatrix}.$$

In order to find the acceptance probability α, instead of calculate all possible pairwise SP score in B and B', we only need to calculate $p(b,e)$, $p(h,e)$, $p(c,f)$, $p(i,f)$ in B and $p(b,f)$, $p(h,f)$, $p(c,e)$, $p(i,e)$ in B'. If we calculate all possible pairwise SP scores in B and B', there are in total 18 pairs to calculate. We would have saved 11 units of computational time. Imagine, if we have an alignment matrix with the number of sequence(N) is large, then we will save a lot of computational time by using our formula.

4.2.2 Simulated Annealing for the sequence alignment problem

We have discussed the general simulated annealing method in section 2.5 to improve the optimization problem. We can also apply it to our sequence alignment problem. Consider running the Metropolis chain in section 4.2, letting β increase over time. In other word, choose a function $\beta(n)$ that increases slowly with time. Then define the acceptance probabilities as follows:

$$\alpha = \min\left\{1, \frac{\exp[\beta(n) \cdot score(B_{prop.})]}{\exp[\beta(n) \cdot score(B_{curr.})]}\right\},$$

By theory, as $\beta(n) \to \infty$, the distribution $\hat{\pi}^{\beta}$ will stay at the optimal solution. We should be careful that we want β increasing slowly enough so that the chain will not be stuck in local maxima.

In the next section, we will apply simulated annealing in our simulation. We will

use a very simple example of sequence alignment to demonstrate how to apply our Metropolis algorithm. In that particular example, we will set $\beta(n)$ as a sine function and the reason will be shown later in that section.

4.3 Simulation results

Simulation is a common tool for computer system performance analysis. A simulation model usually gives a convenient way to predict the performance or compare some alternatives. Sometimes, simulation models fail and produce misleading or useless results. The reason may be the model developers are good at statistical technique but bad in computer software development, or they are good at software development but bad in statistical skills. There are few common mistakes we usually make in simulations, such as improper programming language, invalid models, poor random number generator and so on. For more details, reader can see in R. Jain (1991).

4.3.1 Random number generator

When we talk of generating random numbers, we are usually referring to the generation of random sequences of numbers. We are concerned with the generation of pseudo-random numbers by completely deterministic methods. Although the use of deterministic algorithms to simulate randomness may be objectional on philosophical grounds, it does have some advantages:

(1) The properties of pseudo-random sequences can be tailored, in advance, to suit the needs of a particular application.

(2) A pseudo-random sequence can be replicated.

(3) Techniques for generation pseudo-random sequences are amenable to mathematical analysis.

There are three common methods to generate pseudo-random fractions between 0 and 1, which are multiplicative congruential method, linear congruential method and combined method. Usually, we generate integers x_n between zero and some number m and form fractions

$$u_n = \frac{x_n}{m}.$$

Therefore, u_n will lie between zero and one.

(I) Multiplicative congruential method:

choose 3 magic numbers (integers)

$$\begin{cases} m = \text{modulus}, & m > 0; \\ a = \text{multiplier}, & 0 < a < m; \\ x_0 = \text{initial value} = \text{seed}, & 0 < x_0 < m. \end{cases}$$

We recursively compute successive values x_n for $n \geq 1$, using the relation

$$x_n = (a\, x_{n-1}) \bmod m.$$

For each x_n, compute the fraction

$$u_n = \frac{x_n}{m}.$$

(II) Linear congruential method:

Linear congruential method was proposed by D. H. Lehmer in 1951.

choose 4 magic numbers (integers)

$$\begin{cases} m = \text{modulus}, & m > 0; \\ a = \text{multiplier}, & 0 \le a < m; \\ c = \text{increrrent}, & 0 \le c < m; \\ x_0 = \text{ seed} = \text{initial value}, & 0 \le x_0 < m. \end{cases}$$

We compute successive values x_n, for $n \ge 1$, using the recursive relation

$$x_n = (ax_{n-1} + c) \bmod m.$$

For each x_n value, compute

$$u_n = \frac{x_n}{m}.$$

$$\begin{cases} a \neq 1, c \neq 0, & \text{Mixed congrential generator;} \\ a = 1, c \neq 0, & \text{Additive congrential generator;} \\ a \neq 1, c = 0, & \text{Multiplicative congrential generator.} \end{cases}$$

(III) Combined method:

Two or more random generators can be combined to make one new generator (Jain, 1991). There are three such techniques:

- Adding random numbers obtained by two or more generators;
- Exclusive-ir random numbers obtained by two or more generators;
- Shuffle.

We usually concern about the *cycle* of the random generator. Most random numbers $\{u_i\}$ are generated according to a recursive relation

$$u_{n+1} = f(u_n)$$

where u_i have some fixed number of digits. Because there are only finitely many (say k) different values that the u_i can assume, eventually (by $n = k$), there will be some

j and $n > j$ such that $u_j = u_n$, $u_{j+1} = u_{n+1}$, and so on. The random numbers will repeat themselves in the cycle

$$u_j, u_{j+1}, ...u_n.$$

The *period* of a random number generator is the minimum λ such that $u_j = u_{j+\lambda}$. Random number generators with cycles of short period are undesirable.

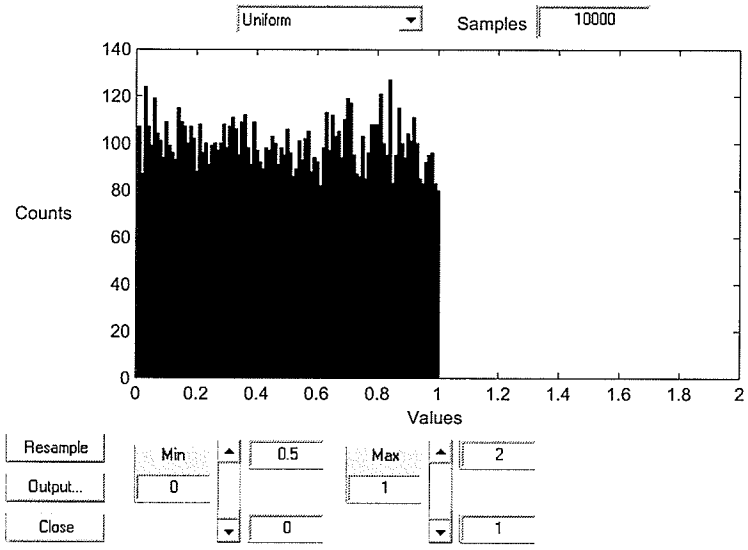

Figure 4.1: A sample of 10,000 uniform (0,1) random numbers in MATLAB

The program we will use in simulating sequence alignment problem is called MATLAB®. The random number generator in MATLAB uses a lagged Fibonacci generator combined with other generator. The function "RAND" in MATLAB is used to generate uniform random number between zero and one. A demonstration of drawing $10,000$ uniform random number is shown in figure 4.1. This function has a huge period. More than 2^{1492} numbers are generated before the sequences start to repeat itself. Another built-in function in MATLAB is called "RANDN", it is

used to generate normally distributed random numbers with mean 0 and standard deviation 1.

4.3.2 Simulation results in sequence alignment

We apply our Metropolis algorithm to sequence alignment problem. We set a very simple example to test this algorithm. In this example, we focus on how the SP score changes when the iteration changes. Furthermore, in order to get a better result, we apply simulated annealing method by using a suitable beta function in our Metropolis chain.

Suppose we have three DNA sequences as follows:

$$s_1 = ACGT \Longrightarrow 1234$$

$$s_2 = AGGT \Longrightarrow 1334$$

$$s_3 = GCCT \Longrightarrow 3224$$

According to our notation, we begin to construct an initial sequence matrix, B_0 by inserting some blank characters or zeros.

$$B_0 = \begin{pmatrix} 1 & 2 & 3 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 3 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 2 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Hence, in this particular example, $N = 3$ is the total number of sequences, $L = 12$ is the length of each sequence. We use MATLAB to simulate this Metropolis algorithm by using different β values in our acceptance probability α. We will sample $n = 30,000$ times and the result will show in figures 4.2, 4.3, and 4.4.

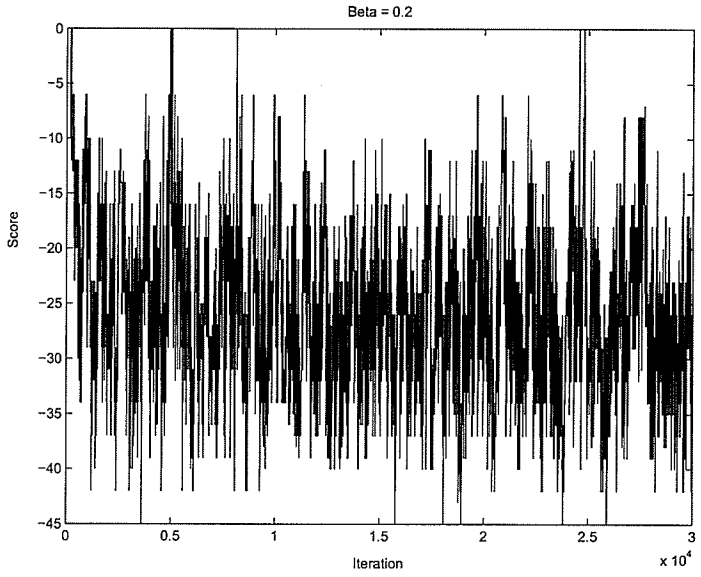

Figure 4.2: SP score of alignment matrix B with $\beta = 0.2$

In figure 4.2, the score(B) is bounded rapidly because when β is closed to zero, the probability of accepting the proposed alignment matrix increases. At this $\beta = 0.2$ value, the algorithm is close to an uniform random search algorithm. In figure 4.3, the score(B) is bounded moderately compared with the score(B) in figure 4.2. However, in figure 4.4, the score(B) gets stuck in the maximum value (which is score(B) $= 0$) in the early iteration and never has a chance to move to different values of score(B).

In order to improve this algorithm, we use simulated annealing method. We can see in this particular example, β is large when it is close to 1. Therefore, we want to increase β between 0 and 1 slowly when the iteration increases. A sine function is a good choice here. Since in this example, we do $30,000$ iterations and use the

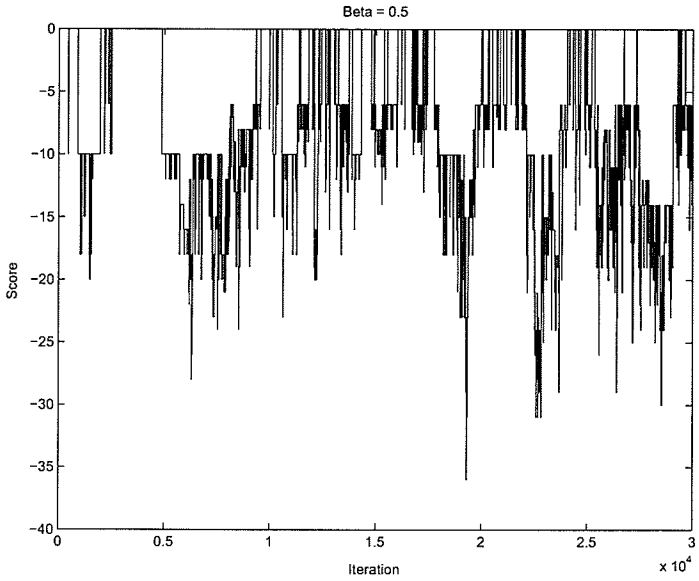

Figure 4.3: SP score of alignment matrix B with $\beta = 0.5$

function

$$\begin{aligned} \beta(n) &= \sin\left(\frac{90°n}{30000}\frac{\pi}{180°}\right) \\ &= \sin\left(\frac{n\pi}{60000}\right). \end{aligned}$$

The result is illustrated in figure 4.5.

In figure 4.5, as expected, the graph is bounded rapidly in the first 15,000 iterations and becomes more stable after that.

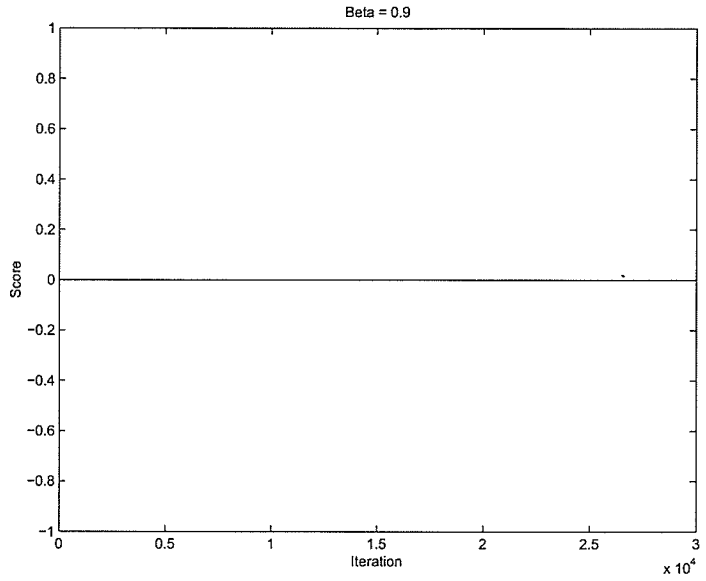

Figure 4.4: SP score of alignment matrix B with $\beta = 0.9$

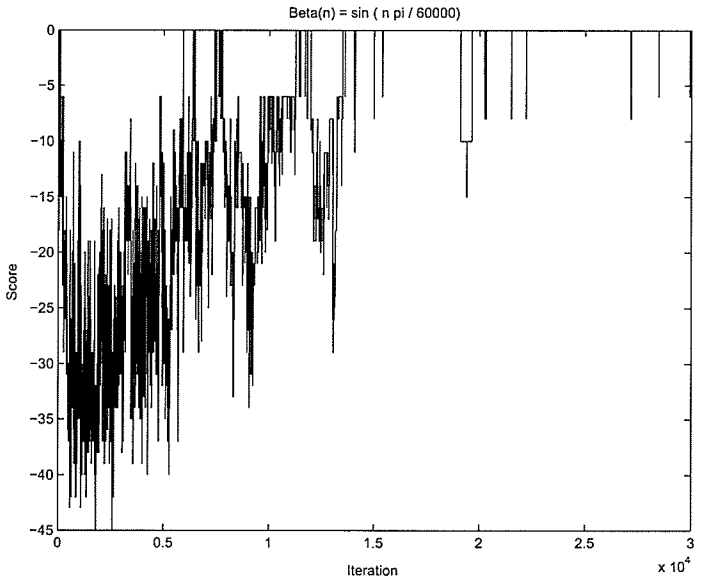

Figure 4.5: Simulating annealing with $\beta(n) = \sin\left(\frac{n\,\pi}{60000}\right)$

Chapter 5

Discussion And Future research

The main purpose of this thesis is to apply Markov chain Monte Carlo approach with Metropolis algorithm in molecular biology. Two different examples (phylogenetic tree, sequence alignment) using statistical method were discussed. However, the example in sequence alignment problem is only a basic demonstration. It seems that our Metropolis algorithm is quite acceptable when the number of sequences and the length of the sequence are small. One of the problems in our algorithm is balancing the rate of mixing and speed of calculation. It means that the initial alignment matrix will remain unchanged most of the time in the beginning of iteration because in order to change the initial matrix, there are only some limited entries inside this matrix that can be picked (the nonzero entries in the front of the first zeros column). Therefore, in the random picked entries of matrix, we may spend most of the iteration time by just picking the useless entry (that is, the entry that we picked cannot use to transform the alignment matrix). An improvement may be using a semi-random search by only searching those "right" entries and updating the new entries after each iteration. Further investigation can be done in future research. Another consideration can be using more complex scoring function. The Sum-of-pairs (SP) function that we used is only a basic function. Other scoring functions may consider such as scoring transitional and transversional events to get different score values.

In conclusion, sequence alignment methods have steadily increased over the last

20 years. There are numerous number of computer packages in literature such MultAlign, ClustalW and so on. Programmers have spend a long time developing these packages. Our program is only a starting tool to learn this huge "sequence alignment world".

Bibliography

[1] Altschul, S. F. and D. J. Lipman (1989). Trees, Stars, and Multiple Biological Sequence Alignment. *SIAM Journal on Applied Mathematics*, 49, 197-209.

[2] Bonizzoni, P. and G. D. Vedova (2001). The Complexity of Multiple Sequence Alignment with SP-score That is a Metric. *Theoretical Computer Science*, 259, 63-79.

[3] Carrillo, H. and D. Lipman (1988). The Multiple Sequence Alignment Problem in Biology. *SIAM Journal on Applied Mathematics*, 48, 1073-1082.

[4] Cheung, L. W.-K. (2002). "Statistical Pattern Recognition in Genomic DNA Sequences", PH.D. Thesis, University of Manitoba, Department of Statistics.

[5] Clote, P. and R. Backofen (2000). *Commputational Molecular Biology An Introduction.* West Sussex, England: John Wiley & Sons Ltd.

[6] Edwards, A. and L. Cavalli-Sforza (1964). *Reconstruction of Evolutionary Trees in Phenetic and Phylogenetic Classification.* (Edited by V. Heywood & J. McNeil), 67-76. Systematics Association, London.

[7] Gilks, W.R.; S. Richardson; and D. J. Spiegelhalter (1996). *Markov Chain Monte Carlo in Practice.* Boca Raton, Florida: Chapman & Hall.

[8] Hasegawa, M.; H. Kishino; and T. Yano (1985). A New Molecular Clock of Mitochondrial DNA and the Evolution of Hominoids. *Proceedings of the Japanese Academy of Sciences*, 60, 95-98.

[9] Jain, R. (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* New York: John Wiley & Sons Inc.

[10] Larget, B. and D. L. Simon (1999). Markov Chain Monte Carlo Algorithms for the Bayesian Analysis of Phylogenetic trees. *Mol. Biol. Evol.*, 16, 750-759.

[11] Lehmer, D. H. (1951). Mathematical Methods in Large-Scale Computing Units. *Harvard University Computation Laboratory Annals*, 26, 141-146.

[12] Li, S.; D. K. Pearl; and H. Doss (2000). Phylogenetic Tree Construction Using Markov Chain Monte Carlo. *Journal of the American Statistical Association*, 95, 493-508.

[13] Li, W.-H. and D. Graur (1991). *Fundamentals of Molecular Evolution.* Sunderland, Mass.: Sinauer Associates Inc.

[14] Madras, N. (2002). *Fields Institute Monographs: Lectures on Monte Carlo Methods.* Rhode Island: American Mathematical Society.

[15] MathWorks Inc. (2002). *Statistics toolbox for use with MATLAB: user's guide.* Natick, Mass.: MathWorks Inc.

[16] Metropolis, N.; A. Rosenbluth; M. Rosenbluth; A. Teller; and E. Teller (1953). Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.*, 21, 1087-1092

[17] Morgenstern, B.; A. Dress; and T. Werner (1996). Multiple DNA and Protein Sequence Alignment Based on Segment-to-segment Comparison. *Proc. Natl.*

Acad. Sci. USA, 93, 12098-12103.

[18] Rannala, B. and Z. Yang (1996). Probability Distribution of Molecular Evolutionary Trees: A New Method of Phylogenetic Inference. *J. Mol. Evol.*, 43, 304-311.

[19] Ross, S. M. (2002). *Simulation.* San Diego, C.A.: Academic Press.

[20] Ross, S. M. (2003). *Introduction to Probability Models.* San Diego, C.A.: Academic Press.

[21] Schard, M. A.; R. E. Weiss; and J. S. Sinsheimer (2001). Bayesian Selection of Continuous-Time Markov Chain Evolutionary Models. *Mol. Biol. Evol.*, 18, 1001-1013.

[22] Setubal, J. and J. Meidanis (1997). *Introduction to Computational Molecular Biology.* Pacific Grove, CA: Brooks Cole Publishing Company.

[23] Stoye, J.; S. W. Perrey; and A. W. M. Dress (1997). Improving the Divide-and-Conquer Approach to Sum-of-Pairs Multiple Sequence Alignment. *Appl. Math. Lett.*, 2, 67-73.

[24] Thorne, J. L.; H. Kishino; and I. S. Painter (1998). Estimating the Rate of Evolution of the Rate of Molecular Evolution. *Mol. Biol. Evol.*, 15, 1647-1657.

[25] Yang, Z. and B. Rannala (1997). Bayesian Phylogenetic Inference Using DNA Sequences: A Markov Chain Monte Carlo Method. *Mol. Biol. Evol.*, 14, 717-724.

Appendix A

Finding The Probability Of Two Consecutive Elements Being Picked Are Different

We are using Combinatorics to solve our problem. For simplification, we look at our sequence as a binary string with k of 1's to represent the non-zero elements (i.e., A, C, G, T or $1, 2, 3, 4$) and $L-k$ of 0's to represent the zero element (i.e., blank "–" or 0). Suppose the binary string is randomly chosen with uniform probability among all $\binom{L}{k}$ such strings. Within this string, one of the $L-1$ pairs of contiguous symbols is selected with uniform probability. Our question is what is the probability that we select a 01 or 10 pair?

Let $b_{L,k,n}$ be the number of binary strings of length L with k 1's and n blocks (of 0's or 1's). A string with n blocks has $n-1$ occurrences of 0 1 or 1 0. Let P be the probability we seek, then

$$\begin{aligned} P &= \frac{\sum_{n\geq 1}[(n-1)\, b_{L,k,n}]}{(L-1)\binom{L}{k}} \\ &= \frac{\left(\sum_{n\geq 1}[n\, b_{L,k,n}]\right) - \binom{L}{k}}{(L-1)\binom{L}{k}}. \end{aligned}$$

Let $B(u) = \sum b_{a,b,c}\, x^a y^b u^c$ be the generating function for binary strings with x marking length, y marking number of 1's and u marking number of blocks then

$$\sum_{n\geq 1}[n \cdot b_{L,k,n}] = [x^L y^k] B'(1).$$

Using $(0,1)^* = 0^*(11^*00^*)^*1^*$, we have

$$\begin{aligned} B(u) &= \left(1+\frac{xu}{1-x}\right)\left(\frac{1}{1-\left(\frac{xyu}{1-xy}\right)\left(\frac{xu}{1-x}\right)}\right)\left(1+\frac{xyu}{1-xy}\right) \\ &= \frac{(1+(u-1)x)(1+(u-1)xy)}{1-x-xy-(u^2-1)x^2y} \end{aligned}$$

We take derivative with respect to u and substitute $u = 1$. After a bit of simplification, we get

$$B'(1) = \frac{x(1+y)-x^2(1+y^2)}{(1-x(1+y))^2}$$

$$\begin{aligned} [x^L y^k]\, B'(1) &= [y^k](1+y)\,[x^{L-1}]\,(1-x(1+y))^{-2} - [y^k](1+y^2)\,[x^{L-2}]\,(1-x(1+y))^{-2} \\ &= [y^k]L(1+y)^L - [y^k](L-1)(1+y^2)(1+y)^{L-2} \\ &= L\binom{L}{k} - (L-1)\left[\binom{L-2}{k} + \binom{L-2}{k-2}\right]. \end{aligned}$$

Hence,

$$\begin{aligned} P &= 1 - \frac{\binom{L-2}{k} + \binom{L-2}{k-2}}{\binom{L}{k}} \\ &= \frac{2k(L-k)}{L(L-1)}. \end{aligned}$$

There must be a simpler way of obtaining such a nice answer. Here is the simple explanation.

Given k 1's and $L-k$ 0's, we can simply choose 2 symbols without replacement to create an ordered pair of symbols.

$$P(0,1) = \left(\frac{L-k}{L}\right)\left(\frac{k}{L-1}\right)$$

and

$$P(1,0) = \left(\frac{k}{L}\right)\left(\frac{L-k}{L-1}\right)$$

Hence,

$$P((0,1) or (1,0)) = \frac{2k(L-k)}{L(L-1)}.$$

Appendix B

Computer Code

Here are the MATLAB code used for the simulations.

We first construct a function called z to be our SP score function.

```
  function z = score(x,y)
%
%    SP- score function
%    x, y are DNA characters
%    x, y belong to {0,1,2,3} => {A, C, G, T}
%


% if a = b = space
  if        (x == 0) & (y == 0)
             z= 0;
% if a = b and a,b ~=space
  elseif    (x==y)& ((x~=0)|(y~=0))
             z= 1;
% if a ~= b and a,b ~=space
  elseif    (x~= y)&((x~=0)&(y~= 0))
             z= -1;
 % if a ~= b and a or b = space
```

```
  else
            z= -2;


  end;
```

Then, a function called T is used to calculate the score of the sequences alignment.

```
% Total score of the sequences alignment using SP-score

function T = Tscore(K)

    [p,q] = size(K);

    T = 0;

for m= 1:p; for n= (m+1):p ; for c= 1:q;

T =  T + score(K(m,c),K(n,c));
end;
end;
end;
```

The main program to simulate our sequence alignment:

```
% S1= ACGT => 1, 2, 3 ,4.

% S2= AGGT => 1, 3, 3, 4.

% S3= GCCT => 3, 2, 2, 4.


i=0 B= [1 2 3 4 0 0 0 0 0 0 0 0
    1 3 3 4 0 0 0 0 0 0 0 0
    3 2 2 4 0 0 0 0 0 0 0 0]


N = 3;

L= 12;

I= 30000;

Beta = .2;

B_0=B;

Total_0 = Tscore(B_0)

T=Total_0;

fid = fopen('tscore_2.txt','w');


for i= 1: I;


    %Beta = sin((i/333.4) * (pi/180));


    B_curr= B;


    % pick a column uniformly at random.
```

```
  a= ceil(N* rand);

  % pick a row uniformly at random.
  b= ceil((L-1)* rand);

 if (B_curr(a,b)== 0)&(B_curr(a,b+1)~=0)
    B_prop = B_curr;
    B_prop(a,b) = B_curr(a, b+1);
    B_prop(a,b+1) = B_curr(a, b);
elseif (B_curr(a,b)~=0)&(B_curr(a,b+1)==0)
    B_prop = B_curr;
    B_prop(a,b) = B_curr(a, b+1);
    B_prop(a,b+1) = B_curr(a, b);
else;
    B_prop = B_curr;
end;

% Partial SP score for B_prop.
Total_prop = 0;
for k=1:N;
  if k ~= a
    Total_prop = Total_prop + score(B_prop(a,b),B_prop(k,b))
    + score(B_prop(a,b+1),B_prop(k,b+1));
  else;
```

```
  end;
end;

% Partial SP score for B_curr.
Total_curr = 0;
for l=1:N;
  if l~= a
    Total_curr = Total_curr + score(B_curr(a,b),B_curr(l,b))
    + score(B_curr(a, b+1),B_curr(l,b+1));
  else;
  end;
end;

% Calculation of Alpha.
num = exp((Beta)*Total_prop);
den = exp((Beta)*Total_curr);
Alpha = min ( [1,(num/den)] );

%Generate a uniform random number between [0,1].
x = rand;

% Acceptance of B_prop.
if Alpha > x
   B = B_prop;
```

```
    T= T + (Total_prop - Total_curr);
  else
    B = B_curr;
    T= T;
  end;

fprintf(fid,'%6.0f\n' ,T);
end;
fclose(fid);
```