1-1-2005

Parametric analysis for ungapped Markov models of evolution

Balaji Venkatachalam
Iowa State University

Parametric analysis for ungapped Markov models of evolution

by

Balaji Venkatachalam

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
David Fernández-Baca, Major Professor
Maria Axenovich
Oliver Eulenstein

Iowa State University

Ames, Iowa

2005

Graduate College
Iowa State University

This is to certify that the Master's thesis of

Balaji Venkatachalam

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

We present efficient sensitivity-analysis algorithms for two problems involving Markov models of sequence evolution: ancestral reconstruction in evolutionary trees and local ungapped alignment under log-odds scoring. Our algorithms generate complete descriptions of the optimum solutions for all possible values of the evolutionary distance. The running time of these algorithms are comparable to the running time of the algorithms for a fixed parameter value. The running time for the parametric ancestral reconstruction problem under the Kimura 2-parameter model is $O(kn + kn^{2/3} \log k)$, where n is the number of sequences and k is their length, assuming all edges have the same length. For the parametric gapless alignment problem under the Jukes-Cantor model, the running time is $O(mn + mn^{2/3} \log m)$, where m and n are the sequence lengths and $n \leq m$.

CHAPTER 1. INTRODUCTION

Understanding the evolution of biological sequences is one of the basic problems in biology. Biological data available suggests that the organisms have evolved from a common ancestor. We would like to know the relation and differences between species. Phylogenies are basic structures to analyze and infer the relationships between species. The relationship between species is represented by a phylogenetic tree. Various methods for inferring phylogenies have been developed. Mutation is a natural evolutionary process which results in modification of characters in DNA sequences. Similarity between DNA sequences of organisms can provide a clue to their evolutionary origin. Sequence analysis of DNA helps in learning the similarity and evolutionary relation between species.

Inference problems require a mathematical model. In this thesis we consider a simple but widely-used Markov model, where sequences evolve only through random mutation. Mutation probabilities are themselves non-linear functions of evolutionary distance (or time). Markov models of evolution are the basis for many of the scoring schemes used in sequence comparison. The goal of an inference problem is to find the most likely explanation for the input data, given the model. The inference is dependent on the parameters of the model. Our work is motivated by the observation that the answer provided by any such model is highly sensitive to the model parameters. Slight variations in these values can result in completely different answers.

Given a model, ancestors of the given set of species can be inferred assuming the parameters of the model are known. Several scoring schemes implicitly assume knowledge of the evolutionary distance between the sequences being compared. However, sequences are compared to determine their similarity, which allows one to infer something about the evolutionary distance between them. There is an implicit circularity. Parametric analysis considers the problem for

all possible parameter values. It provides solution with respect to all parameter values thereby avoiding the circularity. This also allows us to identify solutions that would be missed if a fixed parameter value is assumed.

Our main results are algorithms that compute the solution to the ancestral inference problems in phylogeny and similarity regions in sequence alignment at *all* possible evolutionary distances. The algorithm incurs only a slight overhead relative to the effort needed to compute the answer for any *fixed* distance. In the process, we gain some insight into the structure of these problems.

1.1 Background

In the Markov model of DNA evolution considered here, sites evolve independently according to identical random processes. The model is *gapless* in that insertion and deletion events are not allowed. Mutation probabilities for one unit of evolutionary time are given in the form of a 4×4 matrix M. By the Markov property, the mutation probability matrix $M^{(t)}$ for t time units is simply M^t (that is, M raised to the power t). Parametric problems arise from the attempt to understand the sensitivity of the model to changes in the entries of M and the value of t. The nature of this dependence is clearly non-linear.

The Markov model of mutation is the basis for the two problems considered here: phylogeny construction and local alignment. Let S be a set of species, each represented by a sequence of the same length. An *evolutionary tree* or *phylogeny* is a rooted tree T whose leaves are labeled by the elements of S. Tree T is a representation of the evolutionary history of S; its internal nodes represent ancestral species. An *internal labeling* for T is an assignment of sequences to the internal nodes of T, representing a set of hypothetical ancestors for S. Each edge of T has a length, which is the evolutionary distance (time) between its two endpoints. The *ancestral reconstruction problem* is to find the most probable internal labeling for the tree. The probability of this optimum reconstruction is a function of the edge lengths, with different lengths possibly leading to different reconstructions. The probability of the most likely labeling is a measure of the likelihood of the tree (23). An alternative measure of likelihood is

the *total probability* of the tree, which is the sum of the probabilities of all possible all internal labellings for the tree. While the second notion of likelihood also depends on edge lengths, the dependency is more complex than for the first notion (4; 18; 19; 22). We shall not consider this approach further here.

Markov models are also used to identify contiguous regions of high similarity between two sequences. Similarity in this case is measured using *log-odds* scoring (7), which is computed as the logarithm of the ratio of the probability that the similarity occurred by evolution (according to a Markov model) to the probability that this was a purely random event. The underlying mechanism exhibits the same kind of time-dependency as the phylogeny problem described above.

1.2 Our Contributions

We present fast algorithms for computing the optimum solution to parametric ancestral reconstruction and local ungapped alignment for all possible evolutionary distances. Their running times are comparable to those of the algorithms to find optimum solutions for fixed distance. For the parametric ancestral reconstruction problem under the uniform Kimura 2-parameter model (where all edge lengths are equal), we achieve a running time of $O(kn + kn^{2/3} \log k)$, where n is the number of sequences and k is their length. This is comparable to the running time for the fixed-parameter problem. In contrast, typical approaches are slower by a factor at least equal to the number of distinct optima. For the parametric gapless alignment problem under the Jukes-Cantor model, we achieve a running time of $O(mn + mn^{2/3} \log m)$, where m and n are the sequence lengths and $n \leq m$. This is comparable to the time needed to solve the fixed-parameter problem by dynamic programming.

The algorithms are based on two ideas, which may be useful in solving other problems. The first is that, despite the non-linearity of the time dependency, the well-known *log-transform* allows us to view the problems as linear ones, at the expense of increasing the number of parameters. The second idea takes advantage of the independence between the positions. We use the idea of "lifting" the execution of a fixed-parameter algorithm so that it computes the

optimum solution at all evolutionary distances. Implemented in the obvious way, this results in a slowdown proportional to the number of different optimum solutions. We avoid this overhead because the problems we study are amenable to divide-and-conquer.

1.3 Related work

Our work is closely related to *parametric sequence alignment*, which explores the effect of parameter variation on the optimum solution to sequence alignment problems over a *range* of parameter values. The goal is to build a decomposition of the parameter space into *optimality regions*, that is, maximal connected regions within which the optimum alignment is unique. Parametric pairwise alignment was first considered by Fitch and Smith (12). Waterman et al. (24) proposed a systematic way of finding the optimality regions. Gusfield et al. (15) gave the first bounds on the number of regions. Fernández-Baca et al. (10) extended Gusfield et al.'s work to a broader class of problems characterized by their scoring system; these problems include parametric multiple sequence alignment and phylogeny construction.

The importance of parameter choice in stochastic models of sequence evolution has been known for some time (see (1) and the references cited therein). Agarwal and States (1) give an example of a pair of sequences for which local ungapped alignment varies as the distance increases. In two companion papers (19; 18), Pachter and Sturmfels show the connection between parameter choice in statistical models and parametric problems with feature-based scoring schemes. Among other contributions, they give general bounds on the number of optimality regions and describe how dynamic programming algorithms for the fixed-parameter versions of these problems can be "lifted" to solve these same problems parametrically. Their method — the *polytope propagation algorithm* — is closely related to the approach used here.

1.4 Organization of the Thesis

Chapter 2 reviews Markov models of sequence evolution and their application to phylogenies and scoring sequence alignments. Chapter 3 studies the effect of parameter variation on these models. We introduce the notion of linearization and show how to reduce the number of

parameters for the ancestral reconstruction problem. Algorithms for the parametric problems are presented in chapters 4 and 5. Chapter 6 discusses open problems.

CHAPTER 2. MODELS OF SEQUENCE EVOLUTION

In this chapter we discuss simple models of DNA sequence evolution and show how these models are used in scoring alignments and phylogeny problems.

2.1 A Markov model for substitution

The basis for the model of sequence evolution is a 4×4 *substitution* matrix $M = [m_{ab}]$, where m_{ab} is the probability that nucleotide a is substituted by nucleotide b in one unit of evolutionary time. Let $M^{(t)} = [m_{ab}^{(t)}]$ be the matrix where $m_{ab}^{(t)}$ is the probability that nucleotide a is substituted by nucleotide b in t units of time. By the Markov property, $M^{(t)} = M^t$.

We consider two important cases where the elements of $M^{(t)}$ can be expressed in closed form. The *Jukes-Cantor* (*JC*) model (16) is the simplest model of DNA evolution. In this model, the rate of a mutation event is the same at all sites of the sequence. Each character can mutate to any other character with equal probability. If α is the rate of transition at a site, then the rate of change to another character is $\alpha/3$. This is symbolically represented in the Figure 2.1. If between two sequences an evolutionary time (distance) of t has elapsed, the probability that no mutation event has occurred is $e^{-\frac{4}{3}\alpha t}$. The probability that there was at least one event is $1 - e^{-\frac{4}{3}\alpha t}$. Thus the probability that the character `A` mutates to a `T` at the end of the time t when the rate of mutation is α is

$$\Pr(\mathtt{A}|\mathtt{T}, \alpha, t) = \frac{1}{4}(1 - e^{-\frac{4}{3}\alpha t}).$$

In the matrix $M^{(t)}$, $m_{ab}^{(t)} = r(t)$ when $a = b$ (for a *match*) and $m_{ab}^{(t)} = s(t)$ when $a \neq b$ (a *mismatch*). Functions $r(t)$ and $s(t)$ are given by

$$r(t) = \frac{1}{4}\left(1 + 3e^{-4\alpha t}\right) \qquad \text{and} \qquad s(t) = \frac{1}{4}\left(1 - e^{-4\alpha t}\right), \tag{2.1}$$

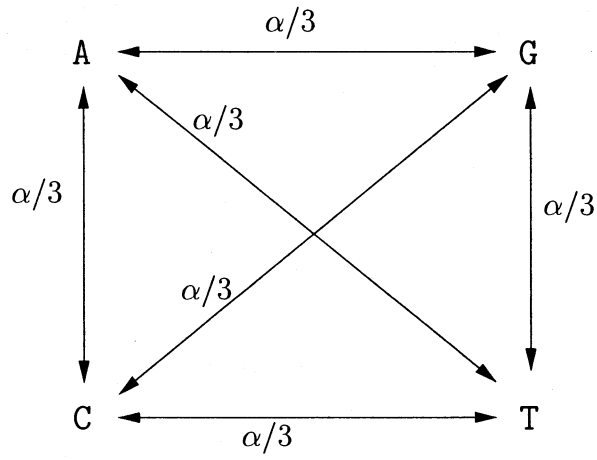

Figure 2.1 Rate of mutation between characters in Jukes-Cantor model

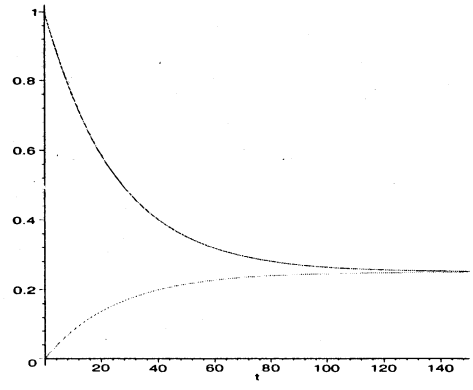

Figure 2.2 The probability of match and mismatch event wrt t

Thus, mismatches have a certain probability and matches have a different (lower) probability, but these probabilities are independent of the nucleotides involved. The probability of these events with respect to time is shown in Figure 2.1. Note that the curve of match (mismatch) probability is smooth with decreasing (increasing), and the value of match and mismatch are equal to 1/4 at infinity suggesting that after long enough evolution the sequences appear random.

A *transition* is a substitution of a purine (A or G) by a different purine or a pyrimidine (C or T) by a different pyrimidine. A *transversion* is a substitution of a purine by a pyrimidine or vice versa. The *Kimura 2-parameter* (*K2P*) model (17) allows for different probabilities for transitions and transversions (the former being more likely than the latter). The entries of the K2P substitution matrix are given by $m_{ab}^{(t)} = r(t)$ when $a = b$, $m_{ab}^{(t)} = s(t)$ when $a \to b$ is a

transversion, and $m_{ab}^{(t)} = u(t)$ when $a \to b$ is a transition. Functions $r(t)$, $s(t)$, and $u(t)$ are given by

$$s(t) = \frac{1}{4}\left(1 - e^{-4\beta t}\right), \tag{2.2}$$

$$u(t) = \frac{1}{4}\left(1 + e^{-4\beta t} - 2e^{-2(\alpha+\beta)t}\right), \tag{2.3}$$

$$r(t) = 1 - 2s(t) - u(t), \tag{2.4}$$

where α, β are transition and transversion rate parameters (7). Since the functions $r(t), s(t), u(t)$ are probabilities and hence satisfy the condition sum of the row in the matrix $M^{(t)}$ is 1. Note that Jukes-Cantor model is a special case of K2P with $\alpha = \beta$.

We often work with logarithms (here assumed to be base 2) of probabilities rather than with the probabilities themselves, since, as will be explained below, this yields linear expressions. For the JC and K2P models, it is convenient to use the following notation

$$\kappa(t) = \log r(t) \qquad \lambda(t) = \log s(t) \qquad \mu(t) = \log u(t). \tag{2.5}$$

Let $A = a_1 a_2 \ldots a_k$ and $B = b_1 b_2 \ldots b_k$ be two sequences. For each $i \in \{1, \ldots, k\}$, a_i (b_i) is referred to as *position* i or *site* i of A (B). Assume that site i of B evolved from site i of A through substitution, independent of and at the same rate as the other positions according to the Markov model just described. Then, the probability that B evolved from sequence A in t time units is given by

$$\Pr(B|A,t) = h_{AB}(t) = \prod_{i=1}^{k} m_{a_i b_i}^{(t)} = r(t)^x s(t)^y \tag{2.6}$$

where x is the number of matches and y is the number of mismatches. This expression can be simplified for the JC and K2P models. For the JC model, taking logarithms and using (2.5), we have

$$\log h_{AB}(t) = \kappa(t) \cdot x + \lambda(t) \cdot y, \tag{2.7}$$

Similarly, for the K2P model, we have

$$\log h_{AB}(t) = \kappa(t) \cdot x + \lambda(t) \cdot y + \mu(t) \cdot z, \tag{2.8}$$

where x, y, z are the number of matches, transversions and transitions, respectively.

2.2 Phylogenies for molecular sequences

Let S be a multiset $\{A_1, \ldots, A_n\}$ of DNA sequences of length k. An *evolutionary tree* or *phylogeny* for S is a rooted tree T whose leaves are labeled by S. The root denotes the common ancestor. Each edge e of T has a *length* l_e that gives the evolutionary distance between its two endpoints.

The internal nodes of a phylogeny T for S represent unknown ancestral sequences. An *internal labeling* for T is a mapping X that assigns a sequence X_v of length k to each node v of T. For each leaf w, X_w must equal the sequence from S that labels w.

Assume that evolution along the edges of T obeys a Markov process, as described in the previous section. Let $\rho(T)$ denote the root of T; for every $v \in T - \rho(T)$, $p(v)$ denotes the parent of v and $e(v)$ is the edge $(p(v), v)$. The *likelihood* of an internal labeling X for T is

$$L_X = \prod_{v \in T - \rho(T)} h_{X_{p(v)} X_v}(l_{e(v)}) \tag{2.9}$$

The *score* of an internal labeling X for T is the logarithm of the L_X. The score is clearly a function of the edge lengths. Here we consider only the simplest case, the *uniform* model, where all edges have the same length t. In this case, if evolution proceeds according to the JC model, the score of an internal labeling X is

$$score(X, t) = \kappa(t) \cdot x + \lambda(t) \cdot y, \tag{2.10}$$

where x and y are the number of matches and mismatches in X, respectively. Under the K2P model, the score becomes

$$score(X, t) = \kappa(t) \cdot x + \lambda(t) \cdot y + \mu(t) \cdot z, \tag{2.11}$$

where x, y, and z are the number of matches, transitions, and transversions in X.

Given a phylogeny T and a set of transition probabilities, the *ancestral reconstruction problem* is to find the most likely (that is, highest-scoring) internal labeling for T. The problem can be solved using a dynamic programming algorithm due to the Felsenstein (8; 7), which computes the solution in $O(nk)$ time.

2.3 Log-odds scoring and gapless local alignment

Local alignment refers to the alignment between the subsequences of the given input sequences. Local alignment helps in identifying highly related subsequences of the input sequences which are conserved during evolution or which show a similarity in structure or form.

Let s be a function that assigns a real-valued score $s(a, b)$ to every pair a, b of characters. A *gapless local alignment* (or simply an *alignment* for short) of two sequences $A = a_1 a_2 \ldots a_n$ and $B = b_1 b_2 \ldots b_m$ is a pair of equal-length consecutive subsequences $a_i a_{i+1} \ldots a_{i+l-1}$ and $b_j b_{j+1} \ldots b_{j+l-1}$ from A and B; its *score* is $\sum_{r=0}^{l-1} s(a_{i+r}, b_{j+r})$. The *optimum gapless alignment problem* is to find a maximum-score gapless alignment. The optimum gapless local alignment can be solved in $O(nm)$ time by dynamic programming (14).

Log-odds scoring schemes are based on the Markov model of Section 2.1. The log-odds score measures the relative likelihood that the sequences are related compared to being unrelated. The log-odds score of the pair a, b is the logarithm of the ratio of the probability that a and b are related through evolution and the probability that they are aligned by pure chance. Thus, if the evolutionary distance between the two sequences is t, the log-odds score for a, b is $s(a, b) = \log m_{ab}^{(t)} / q_a q_b$, where q_a, and q_b are the probabilities that bases a and b appear in any position of a sequence.

The actual $s(a, b)$ values depend on t and the model of evolution. We again consider the JC and K2P models. As we noted in section 2, in both of these models $q_a = 1/4$ for all a (7). Let

$$\tilde{\kappa}(t) = \kappa(t) + 4 \qquad \tilde{\lambda}(t) = \lambda(t) + 4 \qquad \tilde{\mu}(t) = \mu(t) + 4. \tag{2.12}$$

Thus, for the JC model, the score of an alignment $\mathcal{A}$ is given by

$$score(\mathcal{A}, t) = \tilde{\kappa}(t) \cdot x + \tilde{\lambda}(t) \cdot y, \tag{2.13}$$

where x and y are the number of matches and mismatches. For the K2P model, the score is

$$score(\mathcal{A}, t) = \tilde{\kappa}(t) \cdot x + \tilde{\lambda}(t) \cdot y + \tilde{\mu}(t) \cdot z, \tag{2.14}$$

where x, y, and z are the number of matches, transitions, and transversions in $\mathcal{A}$.

CHAPTER 3. PARAMETRIC PROBLEMS IN SEQUENCE EVOLUTION

The problems introduced in the previous chapter involve finding a feasible solution with maximum score. In each case, this solution depends on t. We also described a dynamic programming based algorithm that computes the optimum solution for the fixed-parameter problem. For the parametric problem we would like to solve the problem for all values of the parameter t. Our ultimate goal is to partition the positive time axis into *intervals of optimality*; that is, into maximal intervals of time within which a single solution is optimum. This partition, along with the optimum solution within each interval, gives a complete description of the score of the optimum solution as a function of t, for all $t \geq 0$. Building this parameter space decomposition is what we call the *construction problem.*

The dependence of the optimum score on time is non-linear. By working in the log domain and temporarily ignoring certain aspects of this dependency, we obtain a *linearized problem.* This auxiliary problem is easier to analyze, but nevertheless still contains all the information needed to solve the construction problem. In this chapter we study linearization and its implications (section 3.1). We will also discuss the re-parameterization of the problem where we reduce the number of parameters of the ancestral reconstruction problem (section 3.2).

3.1 Linearization

The optimization problems encountered in Section 2 have the following form. Let $\mathcal{F}$ denote a (finite) set of feasible solutions. Each feasible solution $X \in \mathcal{F}$ has a *feature vector* $p(X) = (p_1(X), \ldots, p_d(X)) \in \mathbb{Z}^d$; each coordinate of $p(X)$ is called a *feature.* The features are weighted

according to a *parameter vector* $\gamma(t) = (\gamma_1(t), \ldots, \gamma_d(t))$ to yield the *score* of X as follows:

$$score(X, t) = p(X) \cdot \gamma(t), \tag{3.1}$$

where "$\cdot$" denotes the dot product. The problem is to find the highest-scoring feasible solution. Its score is given by the *optimum score function*,

$$Z(t) = \max_{X \in \mathcal{F}} score(X, t). \tag{3.2}$$

If we ignore the constraints imposed by the dependence on t, we obtain a *linearized* problem. Abusing notation, the score function for $X \in \mathcal{F}$ in this problem is

$$score(X, \gamma) = p(X) \cdot \gamma, \tag{3.3}$$

and the optimum score function is

$$Z(\gamma) = \max_{X \in \mathcal{F}} score(X, \gamma). \tag{3.4}$$

By its definition, the $Z(\gamma)$ function for the linearized problem is the *upper envelope* (that is, the point-wise maximum) of a set of linear functions (2). Therefore, $Z(\gamma)$ induces a decomposition of the parameter space, $\mathbb{R}^d$, into convex polyhedral *optimality regions*, each of which is the maximal connected set of points for which a particular feasible solution has maximum score. Since $score(X, \mathbf{0}) = 0$ for all X, the optimality regions are cones that meet at the origin. Furthermore, the following result is known.

Theorem 1 (Pachter and Sturmfels (19)). *Let A be d-parameter linearized problem whose set of feasible solutions is $\mathcal{F}$ and let $\mathcal{P} = \{p(x) : x \in \mathcal{F}\}$. Suppose that there exist integers $n_1, \ldots, n_d$ such that $\mathcal{P} \subseteq [0, n_1] \times \cdots \times [0, n_d]$. Then the optimum score function of A induces $O\left(\left(\prod_{i=1}^{d} n_i\right)^{(d-1)/(d+1)}\right)$ optimality regions.*

The decomposition induced by $Z(\gamma)$ contains all the information needed to solve the original parametric problem in t: As t varies from 0 to $+\infty$, $\gamma(t)$ traces a curve through the parameter space defined by a set of non-linear parametric equations in t. To solve the parametric construction problem, we must identify the regions in the linearized problem that are traversed by this curve.

3.2 Parametric ancestral reconstruction

As seen in Section 2, under the K2P model, the score of an internal labeling X for a phylogeny T is a function of three parameters, which weigh the number of matches, transitions, and transversions in X. We can apply Theorem 1 directly to the linearized problem to obtain a bound. We get a better bound by reducing the number of parameters. Let m denote the number of edges in T. Then, $x + y + z = km$ where k, as before, is the sequence length. Therefore, the linearized version of equation (2.11) for the score can be re-expressed as

$$score(\mathcal{A}, \kappa, \lambda, \mu) = \kappa \cdot km + (\lambda - \kappa) \cdot y + (\mu - \kappa) \cdot z. \tag{3.5}$$

Since the term $\kappa \cdot km$ is common to all internal labellings, we can eliminate it from the score. Define $\lambda' = \lambda - \kappa$ and $\mu' = \mu - \kappa$. Then, the score function can be redefined as

$$score'(\mathcal{A}, \lambda', \mu') = \lambda' \cdot y + \mu' \cdot z. \tag{3.6}$$

Note that $y, z \leq mk$ and that $m = O(n)$, where n is the number of sequences. Thus, Theorem 1 implies that the number of regions induced by Z is $O(k^{2/3}n^{2/3})$. In fact, it is convenient to examine each position's contribution to the total score of T. Denote position i's contribution by Z_i. By the independence assumption between the sites, $Z = \sum_i Z_i$. Then, by arguments similar to the one just given, Z_i induces $O(n^{2/3})$ optimality regions.

Now consider the role of t. By Equations (2.2), (2.3), (2.4), (2.12), and (2.13), as t varies from 0 to $+\infty$, $(\lambda'(t), \mu'(t))$ traces a monotonically increasing curve on the negative quadrant that goes from $(-\infty, -\infty)$ to $(0, 0)$. The derivative of this curve is a continuously decreasing function of t that goes from 1 to 0 (Fig. 3.1). On the other hand, by the structure of the linearized score function, the boundaries between optimality regions in the linearized problem are straight line segments that meet at $(0, 0)$. Therefore, the curve intersects every region lying within the cone $\lambda' \leq \mu' \leq 0$ exactly once. Thus, given the space decomposition induced by the linearized problem, one can easily solve the parametric problem relative to t.

The Jukes-Cantor model only counts matches and mismatches. As we did for K2P, we can re-parameterize to obtain a redefined problem and score function: $Z(\lambda') = \max_{\mathcal{A}} score'(\mathcal{A}, \lambda') =$

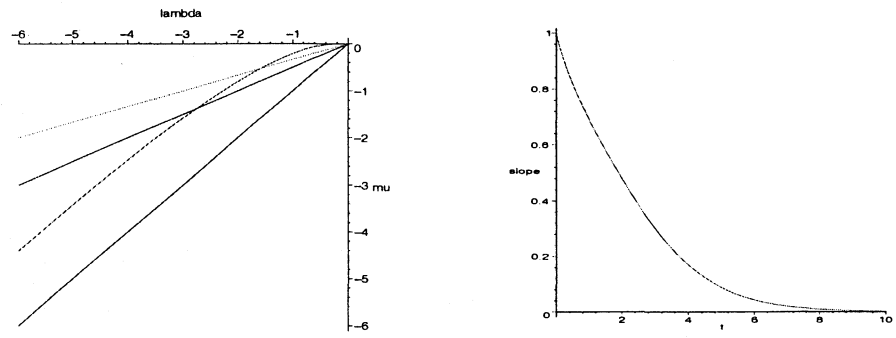

Figure 3.1 Left: λ as a function of μ. The straight lines represent boundaries between regions of the decomposition. Right: Derivative of the λ-μ curve as a function of t.

$\lambda' \cdot y$, where y is the number of mismatches. Hence, for all $\lambda' < 0$, the problem is to minimize the number of mismatches; the value of λ', and thus t, is unimportant. Therefore, there is no parametric problem to study in this case.

3.3 Parametric ungapped alignment

Consider ungapped local alignment with log-odds scoring under the JC model. There are two parameters: the weight $\tilde{\kappa}(t)$ of the matches and the weight $\tilde{\lambda}(t)$ of the mismatches. The linearized score of an alignment $\mathcal{A}$ is $score(\mathcal{A}, \tilde{\kappa}, \tilde{\lambda}) = \tilde{\kappa} \cdot x + \tilde{\lambda} \cdot y$, where x and y are the number of matches and mismatches. Unlike the case for phylogenies, it is not possible to establish a linear relationship between x and y, so we cannot eliminate parameters. Since $0 \leq x, y \leq n$, where $n \leq m$, Theorem 1 implies that the number of optimality regions is $O(n^{2/3})$. Note that, as for the ancestral reconstruction problem, all optimality regions meet at $(0, 0)$; the boundary lines between them are rays emanating from this point.

Now consider the original non-linear problem in t. By Equations (2.1), (2.12), and (2.13), as t is increased from 0 to $+\infty$, $(\tilde{\kappa}(t), \tilde{\lambda}(t))$ traces a curve that goes from $(+4, -\infty)$ to $(2, 2)$, crossing the $\tilde{\lambda}$-axis at $t = (1/4\alpha)\ln(4/3)$. In the process, since $\tilde{\lambda}(t)$ is monotonically increasing, the curve intersects every optimality region of the linearized problem in the $(+, -)$ quadrant exactly once. In the positive quadrant, this curve intersects every optimality region that lies below the line $\tilde{\lambda} = \tilde{\kappa}$ (Figure 3.3).

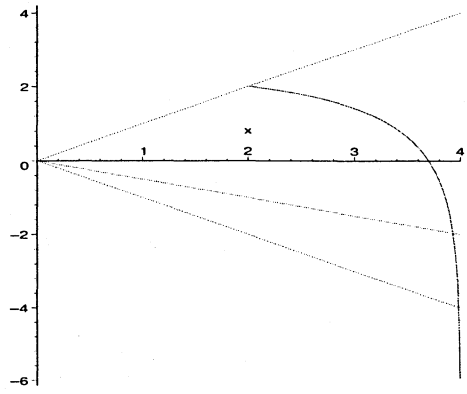

Figure 3.2 Parameter space decomposition for the parametric local alignment

3.4 Overview of the algorithms

We now present an overview of algorithms for parametric phylogenetic analysis and parametric ungapped alignment. Both procedures are based on the same technique. The first step is to build the parameter-space decomposition induced by the linearized version of the problem. As we discussed in section 3.1 this problem ignores the constraints that probabilities must satisfy and thus considers a larger set of parameter values than actually needed. In the second step, these constraints are re-introduced. The parameters in the linearized problem are expressed as parametric equations in t. These equations define a curve that intersects with successive regions of the decomposition as t is varied from 0 to $+\infty$. To solve the parametric construction problem, we simply find these regions.

As we mentioned before, the linearized problems have dynamic programming algorithms for their fixed-parameter versions. These algorithms use two operations on real numbers: addition and finding the maximum. To construct the space decomposition induced by the parametric problem, instead of running the algorithms for a single parameter, we execute them for *all* parameter values. Instead of manipulating real numbers, the parametric algorithms work with piecewise linear convex functions. Instead of addition of real numbers, the modified algorithms add piecewise linear convex functions of the parameters, and instead of taking maxima of numbers, they compute upper envelopes of piecewise linear convex functions. This is similar to Pachter and Sturmfels' polytope propagation method (19; 18), with two differences. A minor

one is that polytope propagation works in the dual space of convex hulls, while our method works in the original space of upper envelopes. A more significant difference is that the absence of gaps allows us to use divide-and-conquer. This leads to algorithms that solve optimization problems for all evolutionary distances in time that closely matches that needed to solve them for a fixed distance. We describe the algorithm for parametric ancestral reconstruction problem and the parametric local alignment problem in the next two chapters respectively.

CHAPTER 4. PARAMETRIC ANCESTRAL RECONSTRUCTION

In this chapter we provide algorithms for parametric ancestral reconstruction problem. We will first consider the special case of balanced binary trees. The more general case is consider in section 4.2

4.1 Balanced trees

In this section we consider the case of perfectly balanced evolutionary tree. Each node has exactly two children and all leaves are at the same level.

We will first describe the fixed-parameter case which is a dynamic programming algorithm due to Felsenstein (8). Due to the independence assumption between sites, we can process each position separately. Let $Z_i^{(u)}(\lambda, \mu)$ denote the optimum score for site i at the node u with respect to parameters λ, μ and $Z_i(\lambda, \mu)$ denote optimum score for the site i for the entire tree. It is well known (see, e.g., (8)) that for each fixed choice of λ and μ,

$$Z_i^{(v)}(a, \lambda, \mu) = \max_b \left(\delta_{ab} + Z_i^{(u)}(b, \lambda, \mu)\right) + \max_b \left(\delta_{ab} + Z_i^{(w)}(b, \lambda, \mu)\right), \tag{4.1}$$

where v is the parent of u, w and δ_{ab} equals 0 if $a = b$, μ if $a \to b$ is a transition, and λ if $a \to b$ is a transversion. To compute the score of the tree, we use dynamic programming with the above recurrence and compute the value of Z_i^u in the bottom-up approach (from leaves to the root). The optimal score for the tree Z is obtained by summing the values of Z_i.

As noted in chapter 3, by reparameterization we have a parametric problem in two parameters λ, μ. We are only interested in the negative quadrant of the λ–μ plane. Observe that, since the score of any internal labeling is 0 at $(0, 0)$, all optimality regions are cones that meet at the origin. Now, a line with one fixed parameter value intersects all the cones. We fix λ,

at any negative value, say -1, while varying the other. This reduces the dimensionality of the problem by one, and leaves us with convex piecewise linear functions of one parameter. Such functions consist of a succession of line segments. Each segment represents a region of optimality. The point of intersection of line segments is called a *breakpoint.* The projection of these breakpoints onto the μ axis partitions it into a sequence of intervals.

An important property of convex piecewise linear functions of one parameter is that the upper envelope and the sum of two such functions can be computed in time linear in their total number of breakpoints. Suppose we have $Z_i(\lambda, \mu)$ for each i. Then, the optimum score function is $Z(\lambda, \mu) = \sum_i Z_i(\lambda, \mu)$. As seen in Section 3, the number of optimality regions induced by Z_i is $O(n^{2/3})$. We can compute the sum of the Z_i's by a process similar to merging.

Algorithm 1 Algorithm to solve the parametric ancestral reconstruction problem

1: **for** each site $i \in \{1, \ldots, k\}$ **do**
2: **for** each node v in the post-order traversal order **do**
3: **for** each character $a \in \{\mathtt{A}, \mathtt{C}, \mathtt{T}, \mathtt{G}\}$ **do**
4: $Z_i^{(v)}(a, \lambda, \mu) = \max_b \left(\delta_{ab} + Z_i^{(u)}(b, \lambda, \mu)\right) + \max_b \left(\delta_{ab} + Z_i^{(w)}(b, \lambda, \mu)\right)$
5: **end for**
6: **end for**
7: **end for**
8: **for** every pair of functions Z, Z' **do**
9: $Z(\lambda, \mu) = Z(\lambda, \mu) + Z'(\lambda, \mu)$
10: **end for**

Theorem 2. *Algorithm 1 solves the parametric ancestral reconstruction problem for the Kimura 2-parameter model in* $O(kn + kn^{2/3} \log k)$ *time.*

Proof. **Correctness:** Recurrence (4.1) holds for the single parameter case. The above algorithm computes the recurrence for all values of the parameters in step 4 and produces $Z_i(\lambda, \mu)$, the upper envelope which provides the solution for the site i for all parameters λ, μ. These upper envelopes are added pairwise and the final solution is obtained in steps 8–10. This process, called *lifting*, obtains the solution to all the parameter values simultaneously.

Run-time Analysis: We will consider the time spent in the loop of step 1 to compute $Z_i^{(\rho)}(a, \lambda, \mu)$ for each a. Consider the subtree rooted at node v with n nodes. For each choice of

b, $Z_i^{(u)}(a,\lambda,\mu)$ induces $O(n^{2/3})$ regions, since the subtree rooted at u has $n/2$ nodes. Therefore time to construct $\max_b\left(\delta_{ab} + Z_i^{(u)}(b,\lambda,\mu)\right)$ is $O(n^{2/3})$ regions. Similarly time to construct $\max_b\left(\delta_{ab} + Z_i^{(w)}(b,\lambda,\mu)\right)$ also takes $O(n^{2/3})$ time. Each of these functions have $O(n^{2/3})$ regions. Computing their sum to obtain the upper envelope takes will take $O(n^{2/3})$ time.

Since the tree is balanced, $T(n) = 2T(n/2) + O(n^{2/3})$. Thus $T(n) = O(n)$. The running time for the loop in step 1 over all sites i takes $O(kn)$ time.

Now consider the loop in step 1. Each site i induces $O(n^{2/3})$ regions for Z_i. In the first step we merge the Z_i pairwise. There are k functions each with $O(n^{2/3})$ regions. This merging takes $O(kn^{2/3})$ time. We repeat the merging process for the resultant upper envelopes. We repeat the process till the final iteration produces one upper envelope, which forms the final solution. There are $\log k$ iterations. In general at the jth iteration there are $k/2^{j-1}$ functions each with $O(2^{j-1}n^{2/3})$. Therefore total time per iteration is $O(kn^{2/3})$. Since there are $\log k$ iterations, the total time for step 1 is $O(kn^{2/3}\log k)$. Therefore the run-time of algorithm 4.1 is $O(kn + kn^{2/3}\log k)$. □

4.2 Unbalanced trees

The analysis of the theorem in the previous section to obtain a linear time solution holds for the perfectly balanced tree case. The crucial part for the linear execution time is the recurrence of $T(n) = 2T(n/2) + O(n^{2/3})$. It is easy to see that for an unbalanced tree the above recurrence does not lead to a linear time solution. In this section we show the use of centroid decomposition, which will help in maintaining the linear run-time.

Centroid decomposition works on the basis of tree decomposition. A *tree decomposition of a graph* G is a labeled tree (T, X) where T is a tree and X is a labeling for T, such that for all $i \in V(T)$, $X(i) = X_i \subseteq V(G)$, and such that: (1) $\cup_{i\in V(T)} X_i = V(G)$; (2) for every $(u,v) \in E(G), \{v,u\} \in X_i$ for some $i \in V(T)$; (3) if j likes on a path of T from i to k, then $X_i \cap X_k \subseteq X_j$. It is well known that given a tree, a balanced tree decomposition can be obtained in linear time (13).

A graph G is a *k-terminal graph* if it is given together with a list $\texttt{terms}(G) = t_1, \ldots, t_s, 1 \leq$

$s \leq k$. A *k-terminal operator* φ *of arity* $r \geq 0$ is defined as follows. If $r = 0$, then φ is a k-terminal graph; else φ joins r k-terminal graphs $G_1, \ldots, G_r$ to obtain a new k-terminal graph $G = \varphi(G_1, \ldots, G_r)$ by identifying the terminals of the graph in a pre-determined way. Given a set $\mathcal{R}$ of k-terminal operators of arity r, let $\mathcal{R}_0$ be the operators of arity 0. Thus $\mathcal{R}_0$ is the set of k-terminal graphs. A *parse tree*, T, of a graph G is a labeled tree which is constructed as follows. If $G \in \mathcal{R}_0$, T consists of a single node v with the label $\delta(v) = G$. Otherwise, let $G = \varphi(G_1, \ldots, G_r)$ be the decomposition of G with respect to R. Then T consists of a root v, with the label $\delta(v) = \varphi$ and subtrees that are parse trees of $G_1, \ldots, G_r$. For any node $v \in T$, we write T_v to denote the subtree of T rooted at v. The subtree represents a graph G_v such that (1) $G_v = \delta(v)$, if v is a leaf, or (2) $G_v = \delta(v)(G_{v_1}, \ldots, G_{v_r})$ if v has children $v_1, \ldots, v_r$. Fernández-Baca and Slutzki (11) showed how a balanced tree decomposition can be used to obtain a balanced parse tree in linear time.

We now sketch how a parse tree can be used to obtain a linear time solution for the ancestral reconstruction problem. The details follow closely to the parametric search algorithm in Fernández-Baca and Slutzki (11). Given a phylogenetic tree G, let T be a bounded-degree linear size parse tree of G, and let $v \in V(T)$. (Note that we are deviating from the general notational convention; here G denotes the input tree and T denotes the parse tree.)

Let z_v denote the optimum value of the score for the subtree T_v and the optimum score for the graph G_v. We need to compute the optimum score z_G for the entire tree G. The algorithm to computing the score is as follows:

Step 1: If v is a leaf and G_v is a primitive graph, then we can compute the score z_v by exhaustive enumeration. Under the assumption that primitive graphs have constant bounded size, this computation takes constant time.

Step 2: Otherwise for each node v, we have a finite number of ways in which G_v can be expressed as a combination of G_u, where u is a child of v in T. If we know the value of z_u, we can combine these values in constant time.

After all the nodes v have been processed, the cost z_G is given by the cost of the root of the parse tree.

Analysis: The parse tree T of the input tree G is linear in the size of G and can be constructed in linear time. The computation of the optimum score of input tree is obtained by computing the optimum score of the nodes of the parse tree from the leaves to the root. Since the parse tree is balanced, the runtime analysis of the previous section can be used for the computation on the parse tree. The size of the parse tree is linear in the size of the input tree and by the analysis of the previous section, the overall running time to compute the cost of the input tree is linear in the size of the input tree.

CHAPTER 5. PARAMETRIC GAPLESS LOCAL ALIGNMENT

In this chapter we consider the parametric local alignment problem. We consider the case where gaps (i.e., the deletions and insertions of characters) are not allowed. We will use the techniques described before in section 3.4 and used in the previous chapter to develop an algorithm. The absence of gaps allows us to use divide-and-conquer to obtain a faster algorithm. We show an algorithm to obtain the best parametric local alignment in time almost equal to the fixed parameter case.

5.1 The algorithm

Let $A = a_1 \ldots a_n$ and $B = b_1 \ldots b_m$ be the input sequences. Given a pair of indices $i \leq n, j \leq m$, let $Z(\kappa, \lambda, i, j)$ denote the score of an optimum alignment between subsequences of $A[1..i], B[1..j]$. In the fixed parameter case, the local alignment problem can be solved by dynamic programming. Construct the usual $(n+1) \times (m+1)$ dynamic programming matrix $C = [c_{ij}]$ for the problem. The entry c_{ij} stores the cost of the optimum local alignment ending at a_i and b_j. Since no gaps are allowed, the value of any entry depends only on the entries along the same diagonal of the matrix. We fill out the dynamic programming matrix C using the following recurrence

$$c_{i,j}(\kappa, \lambda) = \max\{0, c_{i-1,j-1}(\kappa, \lambda) + s(\kappa, \lambda, a_i, b_j)\} \tag{5.1}$$

where $s(\kappa, \lambda, a_i, b_j)$ is the score of the (mis)match as a function of parameters κ, λ as shown in equation 2.13.

As we noted in chapter 3, the linearized version of the parametric problem has two parameters, viz. κ, λ. The parameter space decomposition is a set of cones that meet at the origin.

When we fix one parameter, say, κ at 2, the line intersects all the regions of optimality in the parameter space. We lift the dynamic programming algorithm to solve the problem for all parameters. We can improve the running time of the algorithm by using divide-and-conquer to compute the upper envelope of each diagonal .

Algorithm 2 Algorithm to compute the parametric ungapped local alignment

1: $align[n+m+1]$ //To store the optimal alignment for the $m+n+1$ diagonals
2: $align_1$ //Temporary variable to store the optimal alignment returned by the procedure
3: **for** each diagonal i of the dynamic programming matrix C **do**
4: $align_1 \Leftarrow diagonal(i_1, i_2, j_1, j_2)$ // where i_1, i_2, j_1, j_2 are the endpoints of the diagonal i //$diagonal()$ is described in Procedure 3
5: $align[i] \Leftarrow \max\{align_1[1], align_1[2], align_1[3], align_1[4], 0\}$
6: **end for**
7: **for** each pair of functions Z, Z' **do**
8: $Z(\kappa, \lambda) \Leftarrow \max\{Z(\kappa, \lambda), Z'(\kappa, \lambda)\}$
//for each pair of upper envelopes, starting with the entries from $align[m+n+1]$
9: **end for**

Description of the algorithm Algorithm 2 computes the optimum parametric gapless alignment. As we argued before due to the independence between the diagonals we can consider each diagonal separately. We compute the parametric optimal alignment of each diagonal by divide-and-conquer of procedure 3. The final score of the alignment is obtained by merging the upper envelopes of diagonals in pairs in lines 6–8. For optimality (see analysis in theorem below) we first merge the diagonals pairwise to get a new set of upper envelopes. The resulting functions are paired together and their maximum is obtained. We repeat this process to obtain the upper envelope which is the to the parametric gapless alignment.

In procedure 3, we split a diagonal into two halves and compute the optimum alignment of each sub-diagonal and merge the results of the two halves. We repeat the process recursively to compute the optimal alignment of the sub-diagonals. Since an optimal alignment can straddle the two halves (or multiple splits of the diagonal), in addition to the optimal alignment, we compute the optimal alignment containing the boundary element in each half. We then compute the optimal alignment for a diagonal by performing all combinations of merges of the alignment. In the procedure $diagonal(.)$, $align[1]$ maintains the optimal alignment of the

Procedure 3 *diagonal*(i_1, i_2, j_1, j_2) – The divide-and-conquer algorithm

Require: $0 \leq i_1, i_2 \leq n, 0 \leq j_1, j_2 \leq m, i_2 - i_1 = j_2 - j_1$

//Computes the alignment of $A[i_1 \ldots i_2]$ and $B[j_1 \ldots j_2]$

1: $align[4] \Leftarrow 0$
2: **if** $i_1 = i_2$ **then**
3: $align[1] \Leftarrow \max\{0, score(a_{i_1}, b_{j_1})\}$
4: $align[2] \Leftarrow align[3] \Leftarrow align[4] \Leftarrow score(a_{i_1}, b_{j_1})$
5: **else if** $i_2 = i_1 + 1$ **then** //*length* $= 2$
6: Let $s \Leftarrow score(i_1, j_1, \kappa, \lambda) + score(i_2, j_2, \kappa, \lambda)$
7: $align[1] \Leftarrow \max\{s, score(i_1, j_1, \kappa, \lambda), score(i_2, j_2, \kappa, \lambda), 0\}$ //Best alignment
8: $align[2] \Leftarrow \max\{s, score(i_1, j_1, \kappa, \lambda)\}$ //Left alignment
9: $align[3] \Leftarrow \max\{s, score(i_2, j_2, \kappa, \lambda)\}$ //Right alignment
10: $align[4] \Leftarrow score(i_1, j_1, \kappa, \lambda), score(i_2, j_2, \kappa, \lambda)$ //All elements
11: **else** //*length* > 2
12: $align_1 \Leftarrow diagonal(i_1, i_1 + \frac{(i_2 - i_1)}{2}, j_1, \frac{(j_1 + j_2)}{2})$
13: $align_2 \Leftarrow diagonal(i_1 + \frac{(i_2 - i_1)}{2} + 1, i2, j_1 + \frac{(j_1 + j_2)}{2} + 1, j_2)$
14: $align[1] \Leftarrow \max\{align_1[1], align_2[1], align_1[2] + align_2[4], align_1[2] + align_2[3], align_2[3] + align_1[4], 0\}$
15: $align[2] \Leftarrow \max\{align_1[2], align_1[4] + align_2[2]\}$
16: $align[3] \Leftarrow \max\{align_2[3], align_2[4] + align_1[3]\}$
17: $align[4] \Leftarrow align_1[4] + align_2[4]$
18: **end if**
19: **return** align

current half, *align*[2] and *align*[3] are respectively the optimal alignment including the left-most and the right-most element of the split. *align*[4] is the alignment containing all the elements of the split. Note that we are comparing only the optimum alignment with 0. For the other alignments the comparison is not necessary. This is because, we could have a split where we have a contiguous sequence of mismatches for the characters in the current sub-diagonal (whose cost will be less than zero), yet it is part of an optimal alignment (due to the matches at either end).

Theorem 3. *Algorithm 3 solves the parametric gapless alignment problem under the Jukes-Cantor model in* $O(mn + mn^{2/3} \log m)$ *time.*

Proof. **Correctness:** In the algorithm we construct the solution by dynamic programming using the recurrence 5.1 . In each cell of the dynamic programming table we "lift" the computation to provide the optimal alignment for all parameters (instead of computing the optimal value for a fixed parameter). Since no gaps are allowed, the value of any entry in the dynamic programming table depends only on the entries along the same diagonal. Hence, we can consider the diagonals independently. It is well known that by lifting the computation of the algorithm for all parameters, i.e., using upper envelopes instead of numeric values of fixed parameter values and computing the maximum and sum of these upper envelopes instead of numeric values gives the optimal algorithm with respect to all the parameters. To complete the proof, we need to argue that the divide and conquer algorithm obtains all the optimal solutions along each diagonal.

In the divide phase, we divide a (sub-)diagonal D into two equal parts, which we will call the left and the right halves, denoted by, L, R. To conquer, we compare the (parametric) optimal alignments in each half. The combine part of the algorithm is non-trivial. During the combine operation the optimal alignment of a diagonal is either the best alignment of the sub-halves (by themselves) or an alignment that straddles the two halves. An alignment can straddle the two halves in 4 ways. It can be either an alignment with a part from the left half and a part from the right half (including the common boundary elements), or a part of the left half and the whole of the right half, or the part of the right half and the whole of the left half or the

entire diagonal. Since we are storing the best alignment including the right boundary element and the left boundary element in $align[2], align[3]$ respectively, and the alignment of the entire half in $align[4]$, the optimal alignment is obtained by combining these parts appropriately. The correctness of maintaining the structure *align* is straightforward from the code.

We prove the correctness by induction using the following invariant:

> **Invariant:** If a sub-diagonal D is split into two halves L, R. At the end of the calls to the *diagonal*(.) routine the $align_1$ and $align_2$ structures hold the best alignment of the appropriate parts of L, R.

The base case of the invariant for the sub-diagonal of length 1 and 2 is straightforward. Suppose the invariant holds for the two halves L_1, R_1 of L and L_2, R_2 of R, then, the invariant also holds for the sub-diagonal D by the argument above. Finally, in step 4 of algorithm 2, we compute the best alignment for the entire diagonal by computing the max of all the alignments.

Analysis: We first consider the computation of $Z_i(\kappa, \lambda)$ for the diagonal i. Since we are computing the optimal alignments parametrically, when we split the diagonal into two, each part (of length $n/2$) induces $O(n^{2/3})$ regions of optimality (by Theorem 1). Since there are only constant number of cases (in steps 14–17 of Procedure 3), merging the the alignments take $O(n^{2/3})$. Note that the length of a diagonal is at most $n+1$. Thus, the recurrence for the total running time for each diagonal has the form $T(n) = 2T(n/2) + O(n^{2/3})$, which solves to $O(n)$. Now there are $m + n + 1$ diagonals, therefore computing all the Z_is takes $O(mn)$ time.

Let $Z_i(\kappa, \lambda)$ be the optimum score function of diagonal i. Since the maximum length of the diagonal is n, by Theorem 1 this function induces $O(n^{2/3})$ regions. The optimum score function $Z(\kappa, \lambda)$ is given by $\max_i Z_i(\kappa, \lambda)$. We have $(m+n+1)$ diagonals, with $O(n^{2/3})$ breakpoints along each diagonal. Therefore, by a process of pairing and taking upper envelopes and repeating the process (similar to the pairing and adding of the previous chapter), we can compute Z in time $O(m \cdot n^{2/3} \log m)$. □

5.2 Lower bound on the number of optimal solutions

Theorem 3 shows that the number of optimal regions in the parameter space decomposition of parametric gapless alignment problem is $O(n^{2/3})$. A natural question is, how tight are these bounds. In this section, we discuss the tightness of the bound on the number of optimal alignments. We show a lower bound of $\Omega(n^{2/3})$ by constructing two sequences of length n which induce $\Omega(n^{2/3})$ number of optimal solutions. The proof of the lower bound follows the construction of Fernández-Baca et al. (9), where they show tight bounds for parametric sequence alignment.

Let $\Gamma_0, \Gamma_1, \Gamma_2, \ldots, \Gamma_m$ be the sequence of local alignments along a diagonal so that their score functions have decreasing slopes in the parameter space. Let $score(\Gamma_i, \kappa = 2, \lambda) = 2 \cdot x_i + \lambda y_i$ where x_i and y_i are the number of matches and mismatches in Γ_i. Note that we had fixed the cost of a match at $\kappa = 2$, and the penalty for a mismatch $\lambda \in (-\infty, 2)$. Now the equation of the line defined by the alignment Γ_i in the parameter space decomposition is given by $score_i = 2 \cdot x_i + \lambda y_i$. At the breakpoint between two consecutive alignments the scores are equal, therefore $\lambda_i = 2 \cdot (x_{i+1} - x_i)/(y_{i+1} - y_i)$ for $i = 1, \ldots, m-1$ and $\lambda_0 = 0$. We need the following lemma to give the lower bound.

Lemma 4. *Given a set of fractions* $\frac{a_1}{b_1} > \frac{a_2}{b_2} > \ldots > \frac{a_m}{b_m}$ *such that* $n = a_1 + \ldots + a_m + b_1 + \ldots + b_m + a_1 + 1$. *There exists two strings of length* n *that induce* $m + 1$ *optimal solutions.*

Proof. Since our scoring scheme counts only matches and mismatches we will consider strings over the alphabet $\{0, 1\}$. Let the two strings $S = s_1 \ldots s_n$ and $T = t_1 \ldots t_n$ be defined as follows. We let T be the string of n 1s, i.e., $t_i = 1$ for all $i \in [n]$. Let $a_0 = a_1 + 1$ and $b_0 = 0$, and let

$$i(r) = \sum_{k=0}^{r} a_k \text{ and } j(r) = \sum_{k=0}^{m} b_k$$

for $r = 1, \ldots, m$ and

$$i(-1) = 0 \text{ and } i(0) = 0 \text{ and } j(0) = 0$$

We now define T as follows:

$$t_{i(r-1)+j(r)+k} = 1 \qquad \text{for } k = 1, \ldots, a_r$$

for $r = 0, \ldots, m.$ and $t_i = 0$ for all other values of i.

Since the first string S is all 1s, any sequence of matches and mismatches along the main diagonal (i.e., the diagonal from the cell $(0,0)$ to (n,n)) dominates the sequence of matches along any other diagonal. Therefore, we consider the optimal alignment only along the main diagonal in this case.

Along the main diagonal we have a_0 matches, followed by b_1 mismatches, followed by a_1 matches and b_2 matches etc. We now show that we have $m+1$ different alignments which are optimal. When $\lambda \geq 0$, as the mismatches do not have any penalty, we can include the mismatches in the alignment without decreasing the score. Thus the entire diagonal forms the alignment. Now for different values of λ, we have optimal alignments of the following form - a_0 matches; a_0 matches, b_1 mismatches, a_1 matches etc. Thus, there are $m+1$ optimal alignments. □

Theorem 5. *For every positive integer n, there exists a pair of strings of length n which induces $\Omega(n^{2/3})$ optimal solutions.*

Proof. Let $a_i = x_{i+1} - x_i$ and $b_i = y_{i+1} - y_i$ for $i \in [m-1]$. We shall obtain an bound on the number of distinct irreducible fractions $a_i/b_i, a_i, b_i > 0$ such that

$$\sum_{i=1}^{k}(a_i + b_i) \leq n.$$

The maximum number of fractions, k, is attained by considering each successive integer r and taking fractions of the form a/b such that $a+b=r$ till the set of fractions whose numerators and denominators add up to n.

Let $\phi(m)$ be the Euler totient function (3), which gives the number of integers less than m which are relatively prime to m. The number of fractions a/b such that $a+b=r$ is $\phi(r)$. Thus, the largest number of distinct r's is given by the value of s satisfying

$$\sum_{r=1}^{s+1} r\phi(r) > n \geq \sum_{r=1}^{s} r\phi(r)$$

Fernández-Baca et al. (9) show that

$$s = (\frac{\pi^2}{2})^{1/3} n^{1/3} + O(\log n)$$

Therefore the total number of pairs is

$$\sum_{r=1}^{s+1} r\phi(r) = 3(\frac{n}{2\pi})^{2/3} + O(n^{1/3}\log n). \tag{5.2}$$

We will first show the lower bound for all n of the form

$$n = \sum_{r=1}^{s} r\phi(r) \tag{5.3}$$

there exists two strings of length n whose parametric optimal alignment induces $3(\frac{n}{2\pi})^{2/3} + O(n^{1/3}\log n)$ optimal regions. For this purpose, let F_n be defined as follows

$$F_n = \bigcup_{r=1}^{s} \{a/b : a/b \text{ is irreducible and } a+b=r\}.$$

Then,

$$k = |F_n| = \sum_{\cdot=1}^{\circ} \phi(i).$$

By equation (5.2), we have $k = 3(\frac{n}{2\pi})^{2/3} + O(n^{1/3}\log n)$. Let $a_1/b_1 > a_2/b_2 > \ldots > a_k/b_k$ be the sorted sequence of elements of F_n. Note that $n = \sum_{i=1}^{k} a_i + \sum_{i=1}^{k} b_i$

By Lemma 4 we show that for any sequence of fractions, there exists pair of strings of length n which induce $k+1$ optimal solutions.

If n is not of the form in equation 5.3, we choose s such that

$$\sum_{r=1}^{s+1} r\phi(r) > n > \sum_{r=1}^{s} r\phi(r).$$

It is shown in (9) that $s = (\pi^2/3)^{1/3}n^{1/3} + O(\log n)$. Let $n' = \sum_{r=1}^{s} r\phi(r)$. By Lemma 4 there are two strings of length n' which induce $|F_{n'}| = \sum_{r=1}^{s} \phi(r) = (\pi^2/3)^{1/3}n^{1/3} + O(\log n)$. Appending $n - n'$ 1s at the end of the first sequence and $n - n'$ 0s at the end of the second sequence, the number of optimality regions induced by the two new sequences remains $3(\frac{n}{2\pi})^{2/3} + O(n^{1/3}\log n)$. □

We see from the above theorems that the number of optimal local alignments between a pair of strings of length n is $\Theta(n^{2/3})$. However, the number of optimal solutions appear to be much smaller in practice.

CHAPTER 6. DISCUSSION

We have shown optimal algorithms for the case of ungapped local alignments and phylogeny. Several open problems remain. First, there is the question of the parametric behavior of other Markov models of DNA evolution (for example, the Tamura-Nei model (8)) whose substitution matrices have more complex structures than those of *JC* and *K2P*. Also of interest are the various models of protein evolution. The number of distinct entries in the unit-time substitution matrices for these models is a limiting factor in these problems.

In this thesis, we consider the case where the phylogenetic tree has uniform edge lengths and the rates of evolution at each site is uniform. An interesting open problem is handling non-uniform edge lengths. While linearization can be applied to this problem, the parameter dependence seems complex. Another open problem is to extend our results to models with varying rates of evolution between positions or to models where there are dependencies among sites.

Finally, an important open problem is allowing gaps. When gaps are not allowed, we could use divide-and-conquer to obtain our bounds. While, at least for pairwise alignment, gaps can be handled by polytope propagation (19), it does not appear to be easy to attain the same efficiency as for gapless models.

BIBLIOGRAPHY

[1] P. Agarwal and D. States. (1996). A Bayesian evolutionary distance for parametrically aligned sequences. *Journal of Computational Biology*, 3:1–17.

[2] P. K. Agarwal and M. Sharir. (1995). *Davenport-Schinzel Sequences and their Geometric Applications.* Cambridge University Press, Cambridge–New York–Melbourne.

[3] T.M. Apostol. (1976). *Introduction to Analytical Number Theory.* Springer, New York.

[4] B. Chor, A. Khetan, and S. Snir. (2003). Maximum likelihood on four taxa phylogenetic trees: analytic solutions. In *RECOMB '03: Proceedings of the seventh annual international conference on Computational molecular biology*, pages 76–83. ACM Press.

[5] R. F. Cohen and R. Tamassia. (1995). Dynamic expression trees. *Algorithmica*, 13(3):245–265.

[6] M. Dayhoff, R. Schwartz, and B. Orcutt. (1978). A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5:345–352.

[7] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press.

[8] J. Felsenstein. (2003). *Inferring Phylogenies.* Sinauer Assoc., Sunderland, Mass.

[9] D. Fernández-Baca, T. Seppäläinen, and G. Slutzki. (2002). Bounds for parametric sequence comparison. *Discrete Applied Mathematics*, 118:181–198.

[10] D. Fernández-Baca, T. Seppäläinen, and G. Slutzki. (2004). Parametric multiple sequence alignment and phylogeny construction. *Journal of Discrete Algorithms*, 2:271–287. Special issue on Combinatorial Pattern Matching, R. Giancarlo and D. Sankoff, eds.

[11] D. Fernández-Baca and G. Slutzki. (1997). Optimal parametric search on graphs of bounded tree-width. *Journal of Algorithms*, 22:212–240.

[12] W. M. Fitch and T. F. Smith. (1983). Optimal sequence alignments. *Proc. Natl. Acad. Sci. USA*, 80:1382–1386.

[13] L.J. Guibas, J. Hershberger, D. Leven, M. Sharir, R.E. Tarjan. (1987). Linear-Time Algorithms for Visibility and Shortest Path Problems Inside Triangulated Simple Polygons. *Algorithmica*, 2: 209-233.

[14] D. Gusfield. (1997(. Algorithms on Strings Trees, and Sequences. *Cambridge University Press.*

[15] D. Gusfield, K. Balasubramanian, and D. Naor. (1994). Parametric optimization of sequence alignment. *Algorithmica*, 12:312–326.

[16] T.H. Jukes, and C.R. Cantor. (1969). Evolution of protein molecules. *Mammalian Protein Metabolism*, Vol. III, ed. M.N. Munro. Academic Press, New York.

[17] M. Kimura. (1980). A simple model for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. Journal of Molecular Evolution, 16:111-120.

[18] L. Pachter and B. Sturmfels. (2004). Parametric inference for biological sequence analysis. *Proc. Natl. Acad. Sci. USA*, 101(46):16138–16143.

[19] L. Pachter and B. Sturmfels. (2004). Tropical geometry of statistical models. *Proc. Natl. Acad. Sci. USA*, 101(46):16132–16137.