Probabilistic Models of Evolution and Language Change

Alexandre Bouchard-Cote
Michael Jordan
Daniel Klein
Thomas L. Griffiths
Yun S. Song

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2010-153
http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-153.html

December 14, 2010

Acknowledgement

(See pdf file)

Probabilistic Models of Evolution and Language Change

by

Alexandre Bouchard-Côté

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science
and the Designated Emphasis
in
Communication, Computation, and Statistics

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Michael I. Jordan, Co-chair
Professor Dan Klein, Co-chair
Professor Thomas L. Griffiths
Professor Yun S. Song

Fall 2010

Probabilistic Models of Evolution and Language Change

by

Alexandre Bouchard-Côté

Abstract

Probabilistic Models of Evolution and Language Change

by

Alexandre Bouchard-Côté

Doctor of Philosophy in Computer Science
and the Designated Emphasis
in
Communication, Computation, and Statistics

University of California, Berkeley

Professor Michael I. Jordan, Co-chair

Professor Dan Klein, Co-chair

Both linguistics and biology face scientific questions that require reconstructing phylogenies and ancestral sequences from a collection of modern descendants. In linguistics, these ancestral sequences are the words that appeared in the protolanguages from which modern languages evolved. Linguists painstakingly reconstruct these words by hand using knowledge of the relationships between languages and the plausibility of sound changes. In biology, analogous questions concern the DNA, RNA, or protein sequences of ancestral genes and genomes. By reconstructing ancestral sequences and the evolutionary paths between them, biologists can make inferences about the evolution of gene function and the nature of the environment in which they evolved.

In this work, we describe several probabilistic models designed to attack the main phylogenetic problems (tree inference, ancestral sequence reconstruction, and multiple sequence alignment). For each model, we discussing the issues of representation, inference, analysis and empirical evaluation.

Among the contributions, we propose the first computational approach to diachronic phonology scalable to large scale phylogenies. Sound changes and markedness are taken into account using a flexible feature-based unsupervised learning framework. Using this model, we attacked a 50-year-old open problem in linguistics regarding the role of functional load in language change. We also introduce three novel algorithms for inferring multiple sequence alignments, and a stochastic process allowing joint, accurate and efficient inference of phylogenetic trees and multiple sequence alignments.

Finally, many of the tools developed to do inference over these models are applicable more broadly, creating a transfer of idea from phylogenetics into machine learning as well. In particular, the variational framework used for multiple sequence alignment extends to a broad class of combinatorial inference problems.

Acknowledgements

I am humbled by the sum of help, both at a professional and personal level, that made this thesis possible. Summarizing in a few paragraphs this generosity is both a challenge and a pleasure.

I am greatly indebted to my two advisors and mentors, Michael Jordan and Dan Klein. Their enthusiasm kept me going in the lows, and their insight guided me to exciting intellectual journeys. Thank you Dan for your devotion. I will never forget your exceptional generosity with your time and energy. Thank you Mike for believing in me since the beginning, and also for introducing me to the field of statistics, my new home.

I had the opportunity to collaborate with a variety of departments at Berkeley. The input from people coming from different points of view was crucial to this cross-disciplinary research. I am grateful to Tom Griffiths, who played an important role in the development of this thesis, both as a committee member and as a collaborator. Many thanks to Yun Song, Andrew Garrett, and Ian Holmes for their warm encouragements and for sharing their expertise. I am also grateful to Jennifer Dy and Bin Yu, for the altruistic help they provided when I was preparing my job talk.

A special thank you to Percy Liang: you have been a fantastic friend and a source of inspiration. It was a great pleasure to see you and Ashley last October, and I hope there will be many such reunions in the future.

I already miss very much the NLP and SAIL research groups, where I was surrounded by exceptional individuals, with big brains and big hearts. I have the fortune to have interacted with Taylor Berg-Kirkpatrick, John Blitzer, John DeNero, Dan Gillick, Aria Haghighi, David Hall, Simon Lacoste-Julien, Percy Liang, Guillaume Obozinski, Adam Pauls, Slav Petrov, Sriram Sankararaman, and Ben Taskar. These interactions were not restricted to the lab environment: I have many fond memories of machine learning runs with John and Percy, delightful diners with Sriram, and sailing excursions with John.

Many thanks also to the occupants of the Ashby House, especially to Mark Tarses, for his generosity and his smile; and to Kurt Miller, Blaine Nelson, and Fabian Wauthier, who were always supportive despite having more than their share of the grumpy version of myself. I will always keep good memories of the infamous house parties. Blaine: thank you for the swimming expeditions, which helped keeping the balance.

I am thankful to Bob and Susan Thackeray, who were always supportive and kind to

me. In the past five years, I have spent many weekends in Victoria and Youbou, and you always made me feel welcomed and at home.

For their help during my transition to grad school, I am grateful to Prakash Panangaden and Doina Precup. They also deserve credit for getting me started in research when I was an undergrad at McGill.

Merci à mes amis de longue date, Alain Doyon, Christian Dumais, Pierre-André Gagnon, Émile Girard, Louis-David Lavoie et Francis Perron, qui m'ont toujours supporté et accueilli à bras ouverts durant mes visites à Montréal. Alain, j'espère que nos updates de math continueront longtemps, et que tu me pardonneras de parfois prendre du temps à répondre.

Je suis reconnaissant auprès de mes camarades en exil à Alma-Ouest, Murielle Doré et Francis Perron: votre gentillesse m'a aidé à tenir bon. Je suis déjà impatient de convoquer notre prochain conventum.

J'ai eu la chance de recevoir beaucoup d'encouragements depuis le tout début dans mes poursuites intellectuelles. Merci à Simone Tremblay et Rodolphe Bouchard, Lilianne Côté et Roger Fortin, Alain, Brigitte, Denis, Hugues et Marie Ginette Bouchard pour m'avoir donné la passion des livres, de la langue, de l'informatique, des mathématiques, de l'histoire, du voyage, et de m'avoir aidé dans mes projets.

Cette thèse est dédiée à ma famille. À mes parents, Pauline Côté et Serge Bouchard, pour leur amour inconditionnel et pour m'avoir tant donné. Je vous aime, et vous remercie d'être un si beau modèle. À mon frère Félix Bouchard-Côté, pour être un grand ami toujours là pour moi. Je suis fier de toi, et heureux que tu aies trouvé ta voie. À la femme de ma vie, Joana Thackeray, pour m'avoir accompagné durant ce long processus. C'est un immense plaisir que de tout partager avec toi.

Dédié à ma famille, à Pauline, Serge, Félix et à Joana

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The goal of phylogenetics is to draw inferences about the past from the diversity of the present. Although phylogenetics is best known from its application to the reconstruction of biological histories (from biodiversity), phylogenetic also has ramifications for the problem of reconstructing linguistic histories (from the world's linguistic diversity). This thesis covers both applications, hypothesizing that the two problems have sufficient similarities to justify a joint study, but also have sufficient differences to foster innovation.

The central challenge of phylogenetics is uncertainty, which is ingrained in the field: even if evolution and language change were completely understood (which is not the case), the sad reality is that part of our past is gone forever. Some changes are irreversible: species go instinct, languages die. This introduces uncertainty in nearly all reconstructions.

The best tool we have for coping with uncertainty is probability theory. As a consequence, biologists have adopted probabilistic modeling for studying evolution over the last few decades. In computational linguistics, on the other hand, this transition is only starting to take place.

1.1 Phylogenetic tasks

Phylogenetic inference can be further divided into several interrelated tasks:

Ancestral sequence reconstruction The problem of reconstructing the sequence cor-

responding to the most recent common ancestor (MRCA) of a collection of extant taxa.

Phylogenetic tree inference The problem of identifying the speciation order (tree topology) above a collection of taxa, possibly with branch lengths estimates for each branch in the tree (for example, to approximate the time between speciation events).

Cognate alignment The problem of finding homologous word forms or gene from pronunciation vocabularies or genomes.

Multiple sequence alignment The problem of identifying, for each cognate set, the homologous nucleotides in the related sequences.

Note that we have made simplifications in the descriptions of these tasks. For example evolution and language change do not strictly follow tree-shaped paths. In the next chapter, we give a more complete picture of the phenomena involved, and at the same time define the terminology used above to describe the tasks.

1.2 Contributions

The contributions of this thesis can be divided into three categories:

New models We have introduced three new model families for phylogenetic inference. See next section.

New algorithms The price of using sophisticated probabilistic models is the intense computational burden associated with them. The second contribution of this thesis is a new set of techniques that reduce this burden. Developing these techniques involves two steps: first, using probability theory, and stochastic process theory in particular, for identifying computationally convenient representations of the models; and second, developing efficient algorithms for computing conditional expectations using these representations. These advances enabled state-of-the-art multiple sequence alignment results.

A new approach to diachronic phonology The first automated system capable of large-scale reconstruction of protolanguages. This system is inspired, but significantly different from current phylogenetic models used in biology. It enabled us to attack the functional load hypothesis, a fifty year old open question in diachronic linguistics.

The first and second points made possible the third point. They were also partly motivated by the diachronic phonology application, but more classical applications to biological data also played a crucial role.

1.3 Organization of the thesis

In principle, one could define a single joint model over all the entities involved in phylogenetic analysis and use it to tackle all four tasks described in the previous section. Using a joint model has pros and cons. On one hand, there are questions that can only be answered if a consistent answer to several or all of the inference problems is available. We show an example in Section 3.4.6, where a quantity called functional load needs to be estimated, which depends on the ancestral sequence reconstructions, the alignments, and the phylogenetic tree. On the other hand, when the focus is on a single task, simpler models may perform better for statistical and computational reasons (joint models generally have more parameters and involve a more complicated combinatorial space to sum over).

We therefore present in this thesis a toolbox containing many related models. The models covered are:

Phylogenetic Exponential Family (PEF) This is the richest model in the toolbox, featuring contextual dependencies, branch-specific parameterization, and explicit representation of alignments, trees and ancestral sequences. Using this model, we achieved state-of-the-art automatic proto-language reconstruction. We also validated the model on biological sequences, measuring the accuracy of multiple protein sequences alignments.

Multiple Alignment Random Field (MARF) This is a simpler model, focused on the task of multiple sequence alignment. It is flat, in the sense that all pairwise sequence alignments are considered instead of those along a hidden phylogenetic

tree. This flat configuration enables inference using a novel variational inference framework. This yields state-of-the-art multiple sequence alignment performances.

Poisson Sequence Change Process (PSCP) This model enables efficient phylogenetic tree inference, addressesing one important limitation of both of the previous models. It is a promising trade-off between the richness of PEFs and the modeling limitations of MARFs. Using this model, we obtained state-of-the-art results in phylogenetic tree inference.

We give a more detailed account of the trade-offs between the different models in the conclusion of this thesis.

We start by giving some background in the next chapter. Then we devote one chapter for each model, except for PEF, on which we devote an extra chapter to explore in detail the problem of computing conditional expectations in this model.

Chapter 2

Background

2.1 Summarizing taxa with sequences

In this thesis, sequences over a finite alphabet will be used to summarize both languages (roughly, a maximal group of mutually intelligible dialects) and species (roughly, a maximal group of organisms capable of interbreeding and producing fertile offspring). In biology, although morphological characters such as height have traditionally dominated the field, the sequence representation has taken over and is now standard. Biological sequences can be made out of an alphabet of amino acids (in the case of protein), or nucleotides (DNA and RNA).

In computational approaches to diachronic linguistics, sequential summaries are not yet mainstream: coarser, finite-dimensional representations (described in Section 3.2.1) are still prevalent. These coarser representations contrast not only with the representations used in computational biology, but also those used in traditional (non-computational) diachronic phonology, where word forms are generally modeled by sequences of sound units called *phoneme*. In the linguistic component of this thesis, we depart from finite-dimensional representations, and take an approach inspired by traditional diachronic phonology: languages are encoded by the pronunciation of their words. The alphabet used to represent these sequences is the International Phonetic Alphabet (IPA), a standard encoding for the sounds of the world's languages.

A single sequence is not always sufficient to give a reasonable summary of a language or species (the term *taxon* will be used throughout this thesis to describe languages or

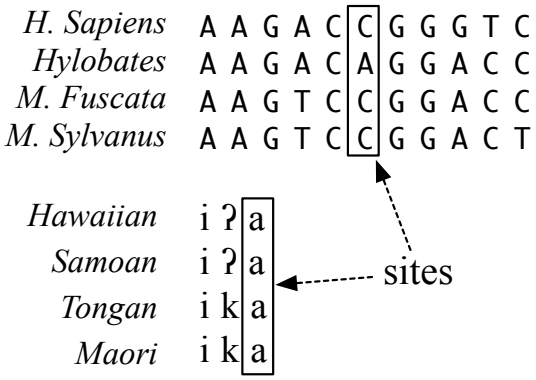


Figure 2.1. Toy examples of comparative phonological and genomic data. A site is a collection of nucleotides or phonemes all descending from the same ancestral character.

species collectively). For example, there are many words in a language's vocabulary, and there are many chromosomes in a species' genome. The insufficiency of using a single sequence is especially acute in the case of language representation, since any single word is very short, but the number of words is large. Fortunately, models assuming a single-sequence representation can be extended to collection-of-sequences representations. This is discussed in more details in Section 2.5, but for simplicity, let us assume for now that there is a single string per taxon.

2.2 Shared ancestry

Now that a representation has been fixed, let us consider some examples of comparative phonological and genomic data (see Figure 2.1).

Note that in the data shown, all the sequences have the same length. This is an atypical case since insertions and deletions (*indels*) usually change the length of both biological and phonological sequences. We will discuss indels in more details in Section 2.4, but this toy example will simplify the discussion by allowing us to easily define the notion of a *site*, a collection of nucleotides or phonemes (we will refer to generic elements of the alphabet using the term *character*) all descending from the same ancestral character. In this example, the characters in site i are just those at position i in each sequence.

All the characters that come from a common ancestor are called *homologous*. Homologous characters tokens are not necessarily of the same type, because of a kind of point mutation

called *substitution*, defined as the mutations that do not change the length of the sequence, and act on a single character of this sequence.

Looking at the data, it is clear that the sequences share many similarities. The main explanations for these similarities are:

Chance These sequences might in fact be unrelated. Since the alphabet and the sequence length is finite, there is a positive probability for this event, but it decays exponentially in the length of the sequences.

Shared ancestry The taxa observed descend from a common ancestral taxon, consequently their sequence representations are similar. In the case of basic biological structures such as ribosomes, shared ancestry explains similarities among all living organisms. In linguistics, on the other hand, it is not known for all pairs of languages if there are word forms with a shared ancestry.

Convergent evolution This happens when evolution is optimizing an heavily constrained fitness landscape, and where there are only a few viable solutions. For example, parents tend to associate the first word-like sounds emitted by babies to "mother" and "father." The easily produced bilabial stops are hence overrepresented in the phoneme distributions of these words, leading to similarities such as Chinese baba, French papa that are not believed to be caused by a shared ancestry [37].

In this work, we focus on similarities arising from shared ancestry, which often leave a distinctive trace: a phylogenetic tree.

2.3 Phylogenetic trees

Further inspection at the data in Figure 2.1 reveals that the sequences are not only related, but also related at different degrees. Notice for example that if the *H. Sapiens* sequence in Figure 2.1 shares a mutation with exactly one other sequence x, then $x = \textit{Hylobates}$ in the majority of the cases. This structure can be explained by a *phylogenetic tree*.

Phylogenetic trees are models representing the evolutionary relationships between taxa. In the most basic form, *ultrametric trees*, these models are specified by a rooted directed tree with leaves in correspondence with the set of taxa under study (see Figure 2.2.

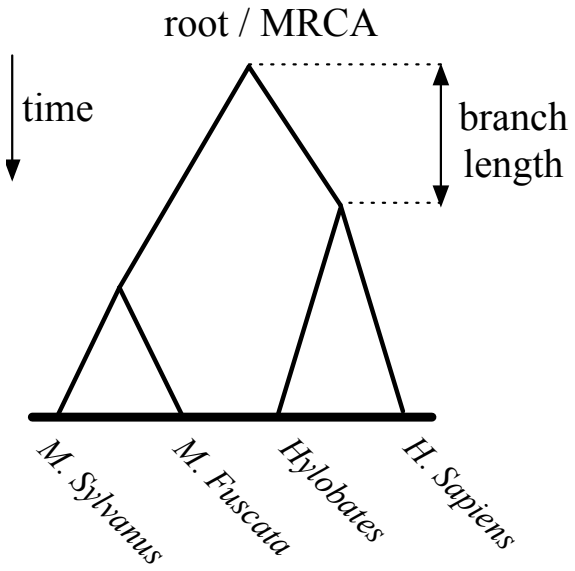


Figure 2.2. An example of an ultrametric tree.

Each internal node n in this tree represents a speciation event corresponding to the Most Recent Common Ancestor (MRCA) of all taxa under n. There is also a positive real number associated with each edge, called a *branch length*, which is proportional to the time that separates two speciation events. Note that when the leaves all correspond to modern organisms (as it is usually the case—fossil DNA is rare), the sum of branch lengths along a path from the root to any leaf is constant.

Since time elapsed between speciation events cannot be measured directly, the expected number of change per site is used as a proxy to estimate branch lengths. Ultrametric trees thus make the assumption that evolution operates at the same rate in all branches of the tree. In most case, this is not realistic, since several factors such as effective population size and environmental pressure can have drastic effects on evolutionary rates.

Non-clock trees generalize ultrametric ones, relaxing the assumption of a constant evolution rate across branches (see Figure 2.3). This means that each branch has an arbitrary positive number associated with it, and that the sum of branch lengths along a path from the root to the leaves under it is not required to be constant.

Phylogenetic trees exist both in the rooted version described above, and in the unrooted version, where directed trees are replaced by undirected ones. Going from unrooted to

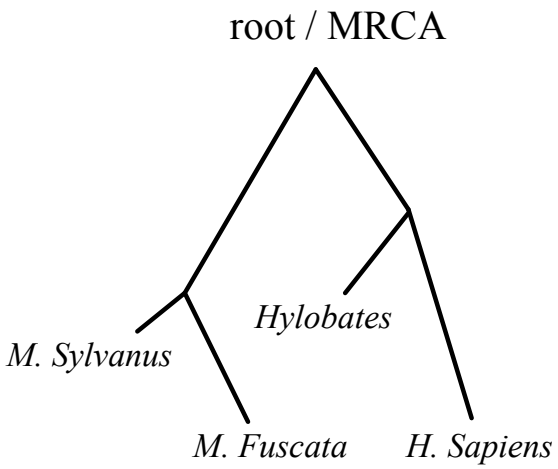


Figure 2.3. An example of a nonclock tree.

rooted in ultrametric trees is trivial (the root is the unique point of constant distance to the leaves), but can require an *outgroup* in the non-clock case (an extra taxon related to the taxa under study, but less closely than the ingroup relationships).

Even non-clock trees do not capture all the relationship that exist between taxa in evolution and language change. There are several sources of deviation:

Borrowing and lateral transfer In diachronic linguistics, the tree assumption is violated by borrowing, for example English's acquisition of French vocabulary after the Norman conquest. In biology, this type of violation is known as lateral transfer, and it occurs when a nucleotide sequence from one species is incorporated into another one by a mechanism other than clonal descent, e.g. via transposon, virus, or plasmid exchange.

Population structure and recombination Another type the tree structure deviation comes from population genetics and dialectology considerations. Consider Figure 2.4, where the red diamond and blue circles represent two alleles present in a population (e.g. two versions of a gene or two versions of the pronunciation of a word form). While in the main population, the frequency of the blue allele becomes dominant and eventually *fixed* in the population (i.e. all the individual share the same allele in the population), the first and second taxa to branch out arise from a small subpopulation (dialect) where the red allele is predominant, so the red allele

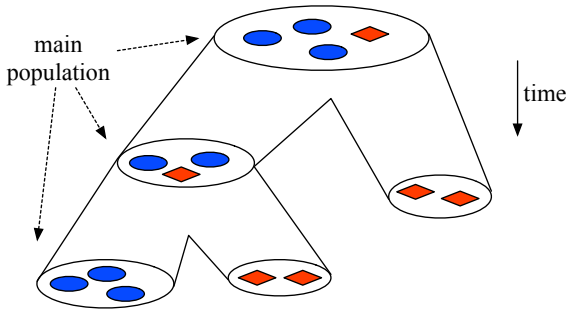


Figure 2.4. An example of the confounding effect of dialects and population structure on phylogenetic inference.

becomes fixed in these two new taxa. The outcome of the process suggests the wrong phylogenetic tree topology. In asexual inheritance processes, the population variations tend to quickly collapse, but sexual reproduction and recombination reverses this trend. Since language is generally acquired from more than one source, dialect structure is also an important aspect of diachronic linguistics.

Hybridization and creoles In the extreme case, a new taxon can be created from two parent taxa. This is known as hybridization in biology and creolization in linguistics. This truly non-tree process is not as frequent as the previous two processes.

2.4 Insertions, deletions and multiple sequence alignments

Let us now look at the effect of insertions and deletions on sequence analysis. Consider the following sequences:

a: C A T A C
b: C A G
c: A T C C

While the sequences still have apparent similarities, insertion and deletions (indel) made their lengths vary, shifting these similarities. We show an example of indel history in Figure 2.5 that yields these sequences.

A convenient way of revealing the similarities between sequences is to determine their sets

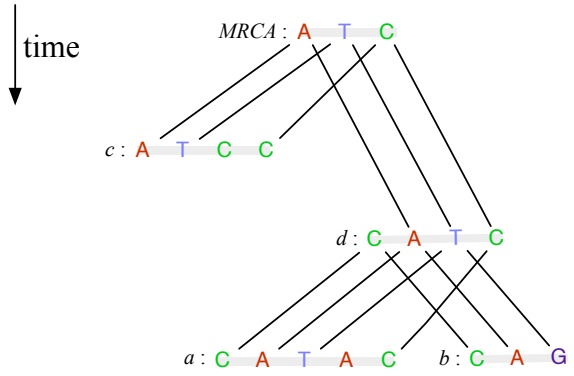


Figure 2.5. Example of an indel history.

a :	C	A	-	T	A	C
b :	C	A	-	G	-	-
c :	-	A	T	C	-	C

Figure 2.6. Example of a linearized MSA.

of homologous characters. When we restrict our attention to three types of mutations (substitution, insertions, deletions), these sets of homologous classes are called *Multiple Sequence Alignments* (MSAs). MSAs are at the core of most phylogenetic models (with a few exceptions, for example [10]).

The restriction to these three types of mutation allows us to visualize MSAs using a *linearized alignment matrix*. For example, the indel history shown in Figure 2.5 can rendered as in Figure 2.6.

In this figure, the sequences have been padded with a *gap character* '-' so that two nucleotides are in the same column iff they are homologous. Note that there can be several ways of producing this visualization, these are considered equivalent, and can be formalized as different linearization of a partial order [80].

When defining MSAs, only the characters in the sequences at the leaves of the phylogenetic

tree are usually included. When the characters of the internal sequences are included as well, we call the more informative structure a *derivation*.

Insertions, deletions and substitutions do not exhaust all types of change found in biology and linguistics. Other types include:

Duplication In linguistics, this process is called reduplication (as in 'bye-bye', for example), and it is a well-studied mechanism to derived morphological and lexical forms. In biology, gene duplication plays a major role in evolution, but operates at a larger scale than linguistics in terms of the number of characters duplicated.

Swaps In linguistics, this process is called metathesis (e.g. Old English frist > English first), in biology, it is called transposition.

In this work, we have ignored these phenomena, mainly for computational and interpretability reasons (the linearized alignment matrix is not possible in general in the presence of either of these changes). Note that theses changes are also generally not regular in linguistics, and therefore less informative.

2.5 Collections of strings

So far, we have talked about taxa being summarized by a single string. In diachronic phonology, this is problematic since individual words are short, and one would like to exploit as many words as possible to get more accurate phylogenetic inferences. In biology it might also be desirable to use information from several genes jointly.

One solution to this is to simply concatenate the genes or the words to form a single string. The difficulty is that there is no canonical ordering of neither words nor genes. To address this, one can *align* the word forms and genes, deeming two words or genes homologous iff they contain homologous characters. In linguistics, a group of word forms that are homologous is called a *cognate set*, we will use the same terminology in biology for simplicity. Both insertions and deletions can be found in the ancestral history of cognate alignments (in linguistics, these cognate indels are caused by semantic changes).

Once a cognate alignment has been found, it is then possible to give an ordering of the gene or word forms that is consistent across all taxa, and therefore treat the data as a single string, introducing a string boundary character '#' between words. Conceptually, this is a valid reduction, but in practice algorithms would be slowed down by such representation.

Language	'coral'	'branch'	'drown'	'voice'
Hawaiian	pu**n**a	ma**n**a	lemo	leo
Samoan	puŋa	maŋa	lemo	leo
Maori	puŋa	maŋa	**r**emo	**r**eo

Table 2.1. Example of sound changes in action.

2.6 Homogeneity and branch specificity

In the basic phylogenetic trees introduced earlier, mutations are more frequent on longer branches, but in reality, the rates of change vary in other ways. This is especially true in diachronic phonology, where *regular sound changes* play a key role.

To understand the concept of regular sound changes, consider the dataset shown in Table 2.1.

It is apparent from this data that the ancestral phoneme *ŋ (the star means that it is a reconstructed, unattested form) was substituted to n along the branch leading to Hawaiian not only in the history of the word form for 'coral', but also in the case of 'branch'. In fact,this substitution occurred in all ancestral words having the proto-phoneme *ŋ. On the other hand, this substitution did not affect any word forms in the branch going to Maori or Samoan. At the same time, the other sound change apparent in the data, l > r affected only word forms going to Maori, showing that a simple proportionality between branch length and amount of change is insufficient in the analysis of phonological data.

In biology, other factors make the distribution of mutations non-homogenous. For example the parts of a sequence critical to its mission (for example, active sites in proteins) are generally more conserved since most organisms with a mutation on these sites does not survive.

2.7 Change tendencies

The non-homogeneities described in the previous section makes phylogenetic inference harder (in statistical terms, by increasing the number of parameters to estimate). Fortunately, other types of regularities have the opposite effect: certain types of change are universally more frequent than others, which will be used in the next chapters to give

lower probability to a large, unrealistic portion of the search space. In linguistics, the field investigating these tendencies is called *typology*.

The most basic tendency is that some mutations are more frequent than others. It is well known for instance that transitions (DNA point mutations that change a purine nucleotide to another purine or a pyrimidine nucleotide to another pyrimidine) are more frequent than transversion (the other nucleotide point mutations). More generally, the transition matrices estimated from protein, phoneme, DNA and RNA data all significantly deviate from the uniform transition matrix. In biology, these non-uniformity can be interpreted using organic chemistry. In linguistics, some of the non-uniformities can be explained using phonetics.

Tendencies exist not only in the distribution of substitutions: in both linguistics and biology, insertions and deletions can alter large subsequences atomically. As a consequence, indel length distributions differ significantly from the geometric distribution one would expect from independent point indels.

Further regularities can be detected by looking not only at the mutations themselves, but also at the context in which they occur (meaning, the flanking characters in the sequence where the mutation occurs). For example, when analyzing exonic nucleotide data, a single nucleotide substitutions will have drastically different effects depending on whether the transcribed amino acid is changed or not (respectively missense and silent mutations). Determining the type of mutation requires looking at a triplet of nucleotides. Since the transcription mapping from nucleotide triplets to amino acid is not injective, some point mutations make the coding triplet transcribe to the same amino acid, and therefore decrease the likelihood of introducing a detrimental mutation.

Again, these contextual regularities are not limited to substitutions, but are also found in indels. For example, large insertions are less frequent in the hydrophobic core of globular proteins, since these stretches often correspond to the folded core.

Change tendencies are especially important in diachronic phonology, where the inventory of possible changes is much larger. There can be easily over one hundred phonemes involved in analyses involving even a moderate number of languages, contrasting with the 22 standard amino acids. Diachronic linguists have developed a formalism to express some of these regularities. This formalism is based on *distinctive features* defined on phonemes, which describe basic units of change, for example voicing, manner of articulation, and palatalization, among others. Changes involving only a few features in this representation are generally considered more frequent cross-linguistically.

2.8 Main approaches to phylogenetic modeling

Traditionally, phylogenetic inference has been done using parsimony-based model. In this framework, the preferred explanations for the data are those involving the minimum amount of changes. Parsimony has since been largely superseded by statistical approaches. The fundamental motivation for this transition is that changes in phylogenetics should not be considered unusual events to be minimized. Instead, they should be expected, be part of the model, and their regularities should be estimated and exploited.

Generative modeling is an important type of approach to phylogenetic inference. There is no standard definition for the concept of generative models, but in this thesis we will define them as models where joint distribution computations (e.g. sampling) are tractable (i.e. in the complexity class P). Generative models are useful in phylogenetics, because for many tasks (tree inference, ancestral sequence reconstruction), it is hard or impossible to perform held-out validation tests (because of a lack of supervised data). An important example is the class of models derived from a directed graphical model. Note that a tractable sampling algorithm for the joint does not imply a tractable sampling algorithm for the posterior. Generating data from the joint distribution is one of the solution used in practice to work around this problem. Given a rooted tree, it is natural to consider a directed graphical model with the graph topology given by the topology of the phylogenetic tree.

The standard technique to specify the transition probability for a node n in the graphical model given its parent n' is to use a *stochastic process*, i.e. a collection of random variables indexed by an uncountable set S, $\{X_t : t \in S\}$. In phylogenetics, this set S is uncountable it models time (i.e. a large number of successive generations is approximated by a continuum). The type of stochastic process involved is usually a *Continuous Time Markov Chain* (CTMC), where $S = [0, T]$, and the individual variables X_t are assumed to have a countable domain. Using the phylogenetic tree, the length of the interval T is obtained from the branch length corresponding to (n, n') in the phylogenetic tree. Then, to get $\mathbb{P}(X_T \in A | X_0)$ (marginalizing all the X_t's in between), one can use standard tools from stochastic process theory.

For example, if X_t has a finite domain, and the sequence (X_t) is assumed to be Markovian, this is done as follows. First, define the matrices $(P_T)_{i,j} = \mathbb{P}(X_T = j | X_0 = i)$. By the Markov assumption, we get that the matrices P_T must satisfy $P_{T+S} = P_T P_S$. We also have $P_0 = I$ as a boundary condition. If these equation were scalar-valued, i.e.

$f(t+s) = f(t)f(s), f(0) = 1$, this would imply that $f(t) = \exp(tq)$ for some $q \in \mathbb{R}$, using basic real analysis. In the matrix-value case, the equations also imply that $P_T = \exp(TQ)$, where this time Q is a matrix (called the *rate matrix*), and exp is the *matrix exponential*, which can be defined using the same series as the standard exponential, but generalized to matrices:

$$\exp(M) = I + M + \frac{M^2}{2!} + \frac{M^3}{3!} + \dots$$

In practice, the matrix exponential is computed by diagonalization.

When X_t has a countably infinite domain, finding expressions for the marginal is more complicated, and a closed-form solution is not guaranteed to exist in general. The countably infinite case is of great relevance to this thesis, since X_t will generally be string-value in the next chapters.

The most popular string-value CTMC is the TKF91 model [87]. Its main advantage is that its marginals $\mathbb{P}(X_T = s|X_0)$ can be expressed as a string transducer with a set of weights obtained from closed-form functions of T.

The approach presented in Chapter 3 is based on a generative model, but not a stochastic process. The one presented in Chapter 5 focuses on MSA and is not based on a generative model. Finally, Chapter 6 takes a generative, stochastic process approach.

2.9 Information lost in sequential phylogenetic summaries

As the name suggests, sequential summaries involve a loss of information. At least three sources of information loss can be identified:

Populations and dialects Two organisms belonging to the same species do not generally share the same genetic code. Similarly, two individuals speaking the same language often pronounce many words differently. As a consequence, collapsing species and languages to a single sequence eliminates the population and dialect structures. We described in Section 2.3 one of the consequences of this loss of information on phylogenetic tree inference. However the explanation of the variance in the string summaries is generally dominated by variation across species rather than variation across individuals.

Geospatial information The location and spread of the taxa can be phylogenetically informative. Very large distance generally means more time elapsed since the last contact. Note that simple geodesic distance is not sufficient though: geography and hydrography, migratory patterns, diffusion rates and other factors will also need to be considered to exploit this information effectively.

Non-sequential inheritance systems There is evidence for non-sequential inheritance both in linguistics and biology. Syntax is an example of a linguistic structure that is not easily accommodated by a sequential framework, yet it is known to change over time. In biology, structural inheritance systems such as those involved in fungal prions similarly escape a sequential encoding approach. In the two examples, the change mechanisms are currently poorly understood, so using them for phylogenetic inference seems premature.

Sequential summaries are still dominant for the purpose of phylogenetic inference, so the discussion in this thesis is centered around this type of summary. Note however that alternatives types of summary exist: morphological character (discrete, continuous and functional), cognate matrices, parameter space representations, population genetic representations such as SNP frequencies, and geographic diffusion models. Note also that different types of summary can be combined.

Chapter 3

Phylogenetic Exponential Families

As described in Section 2.8, generative models of evolution have traditionally been approached from the point of view of stochastic process theory. When simple models suffice, this approach has been very successful, but there exists many situations where simple models do not suffice. An important example, and the focus of this chapter, is diachronic phonology, where sound changes involving contextual and branch-specific dependencies are predominant.

In these more complicated situations, the stochastic process approach becomes less attractive. One reason for this is of computational nature (the differential equations from which marginal are extracted are difficult to solve), but there are also more fundamental limitations of the stochastic process approach that make them poor candidates for diachronic phonology. These failings are explained in more detail in 3.2.6.

In this chapter, we propose a new approach to phylogenetic modeling, called *Phylogenetic Exponential Families* (PEF), that efficiently handles contextual and branch-specific dependencies as well as long indels. Instead of depending on stochastic processes, the approach directly represents branch-specific marginal mutation distributions in the form of the canonical parameters of an exponential family over string transducers. These parameters are estimated from the data. By restricting the type of transducers allowed, this can be done efficiently using a stochastic EM algorithm. PEFs allow easily incorporating context, branch specific phenomena, parameter tying, and expert knowledge.

For concreteness, we first describe the model in the context of protolanguage reconstruction. After showing new algorithms for efficient learning and inference, we then present

a series of experiments on protolanguage reconstruction. These experiments do not only validate the methods, they are themselves important contributions: previous work in diachronic phonology has been limited to non-computational approaches, small datasets, or both. In contrast, we reconstruct using hundreds of modern languages, enabling us to attack new scientific questions.

We then apply the model to the biological setup in Section 3.5, gaining additional empirical validation of the method on a protein alignment task.

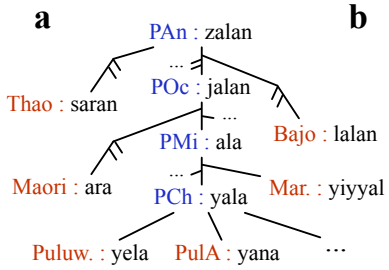
3.1 Motivation

Reconstruction of the protolanguages from which modern languages are descended is a difficult problem, and has occupied historical linguists since the late 18th century. To solve this problem, linguists have developed a labor-intensive manual procedure called the *comparative method* [31], drawing on information about the sounds and words that appear in several modern languages to hypothesize protolanguage reconstructions even when no written records are available, opening one of the few possible windows to prehistoric societies [78, 14]. Reconstructions can help understanding many aspects of our past, such as the migration patterns [68], scripts [93], and technological level [78] of early societies. Comparing reconstructions across many languages can also help reveal the nature of language change itself, identifying which aspects of languages are most likely to change over time, a long-standing question in historical linguistics [58, 41].

Several factors contribute to the complexity of protolanguage reconstruction. Consider for example the development of the French word *chambre*, /ʃambʁ/ in the International Phonetic Alphabet (IPA), translated as 'chamber'. In this case, we know that the word is descended from the Latin /kamera/ 'vault, arched chamber'. First, note that not only substitutions of sounds (acting on Latin /k/ here) are needed to explain the data, but also insertions and deletions (acting respectively on bilabial /b/ and final vowel in the example). Moreover, simple correspondences such as Latin /k/ : French /ʃ/ are often compounds of several changes, in this case going through intermediate Old French /ʧ/. In general, it is not uncommon to find half a dozen or more changes acting in the history of a single word form: see Figure 3.1(a) for an example from the Austronesian language family.

In many cases, unlike the simple example of French and Latin, direct protolanguage evi-

a



b

Language	'road/path'		'leg/foot'
Thao	saran		
Maori	ara		
Puluw.	yela		peʁe
PulA	yana		pasa
Mar.	yiyyal		
Bajo	lalan		
Kwamera		suatuk	
Ghari		sautu	

c

Language	'road/path'		'leg/foot'
Thao	1	0	0
Maori	1	0	0
Puluw.	1	0	1
PulA	1	0	1
Mar.	1	0	0
Bajo	1	0	0
Kwamera	0	1	0
Ghari	0	1	0

Figure 3.1. (a) A subtree of the Austronesian family along which the International Phonetic Alphabet representation for the word 'road/path' mutates. Modern, documented languages are shown in red (Puluw.: Puluwatese, PulA: Pulo-Anna dialect of Sonsorol, Mar.: Marshallese), while unattested, reconstructed protolanguages are shown in blue (PAn: Proto-Austronesian, POc: Proto-Oceanic, PMi: Proto-Micronesian, PCh: Proto-Chuukic). (b) The cognate set corresponding to this subtree. (c) The translation of this information into the kind of binary matrix used in previous work, where the emphasis was on inferring linguistic phylogenies rather than reconstructing languages. This representation omits the structure of the words themselves, and thus discards a significant amount of information.

dence is not available. Fortunately, owing to the world's considerable linguistic diversity, it is still possible to propose reconstructions when written records are unavailable. This is done by leveraging a large collection of extant languages descending from a common protolanguage. Words across these modern languages can be organized into *cognate sets*, containing words suspected to have a shared ancestral form (Figure 3.1(b)). The key observation that makes reconstruction from these data possible is that many diachronic phenomena can be accounted for by a limited set of regular sound changes, each applied to the entire vocabulary of a specific language at a specific stages of its history [31]. For example, other Latin words were subjected to the /k/ > /ʧ/ place of articulation change, such as Latin /kambire/ 'barter' which eventually became French /ʃãʒ/ 'change'. Still, several factors obscure these regularities. In particular, sound changes are often context sensitive: Latin /kor/ 'heart' with French reflex /kœʁ/ illustrates that the place of articulation change occurred only before the vowel /a/.

3.2 Model specification

In this section, we introduce PEFs, a model family capturing all of the phenomena discussed in the previous section. We start by a survey of related previous work and an overview of the new method we propose.

3.2.1 Overview and previous work

Several groups have recently explored how methods from computational biology can be applied to problems in historical linguistics, but this work has focused on identifying the relationships between languages (as might be expressed in a phylogeny) rather than reconstructing the languages themselves [25, 75, 20, 24, 65]. Much of this work has been based on binary cognate matrices, which discard all information about the form that words take, simply indicating whether they are cognate. To illustrate the difference between the representation of the data used by these models and ours, it is useful to compare in Figure 3.1 the form of the input data leveraged by our algorithm (b) versus the much coarser binary cognate matrix (c) used by phylogenetic approaches. The models used for phylogenetics do not have the resolution required to infer ancestral phonetic sequences, and thus cannot be used to reconstruct protolanguages.

The limited resolution of the representations used in previous computational approaches to historical linguistics has meant that almost all existing protolanguage reconstructions have been done manually. However, in order to go deeper into the past and to get more accurate reconstructions, large numbers of modern languages need to be analyzed. The Proto-Austronesian language, for instance, has over 1200 descendant languages widely dispersed throughout the islands of Southeast Asia, the Pacific Ocean, Madagascar and continental Asia [56]. All of these languages could potentially increase the quality of the reconstructions, but the combinatorial increase in the space of possibilities as each language is added makes it hard to manually analyze more than a few languages simultaneously. The few previous systems for automated reconstruction of protolanguages [70, 44] were also unable to handle this combinatorial increase, since they relied on deterministic models of sound change and exact but intractable algorithms for reconstruction.

Using a probabilistic model of sound change and an approximate algorithm for inverting this model allows us to reconstruct the lexicon and phonology of protolanguages given a large collection of cognate sets from modern languages. We make the simplifying assump-

tion that each word evolves along a tree-shaped network of languages. All internal nodes in the tree, particularly the root, are languages whose word forms are not observed, and the modern languages are at the leaves (see Figure 3.2). The output of our system is a posterior probability distribution over *derivations*, represented by a collection of samples $d_1, \ldots, d_n$. Each sampled derivation d_i contains, for each cognate set, a reconstructed transcription of the ancestral forms (see Figure 3.1(a)), as well as a list of sound changes used to go from each word form in the tree to its parent. This representation is rich enough to answer a wide range of queries that would normally be answered by carrying out the comparative method manually, such as which sound changes were most prominent along each branch of the tree.

Following previous work on multiple sequence alignment in computational biology, we model evolution of discrete sequences using a probabilistic string transducer formalism [33]. Unlike simple molecular indel models such as the TKF91 model [87], the parameterization of our model is very expressive: Mutation probabilities are context-sensitive and each branch has its own set of parameters. This branch-specific parameterization plays a central role in our system, allowing explicit modeling of sound changes. For instance, in the example presented earlier, we would like the probability of the mutation /k/ > /ʧ/ to be automatically learned to be close to one in the branch going from Latin to Old French and close to zero in the branch between Old French and French. The importance of context-sensitive modeling is illustrated in the same example (the sound change occurred only before the vowel /a/). This flexibility comes with the cost of having literally millions of parameters to set, creating challenges not found in most computational approaches to phylogenetics. Our algorithm learns these parameters in an unsupervised fashion. The method is based on established principles from machine learning and statistics—maximization of a regularized likelihood objective function using a Monte Carlo Expectation Maximization algorithm [11, 89, 38]. We tied the parameters across branches to learn cross-linguistic trends and overcome data sparsity (see 3.2.5).

3.2.2 Problem formulation

In this section, we formalize the problem we are interested in, i.e. automatic reconstruction of protolanguages given their modern descendants. We aim to reconstruct the words in these protolanguages, where each word is represented as a string of phonemes. We assume that the relationships between languages are known, and are expressed in a phylogenetic tree where modern languages are at the leaves of the tree and internal nodes correspond

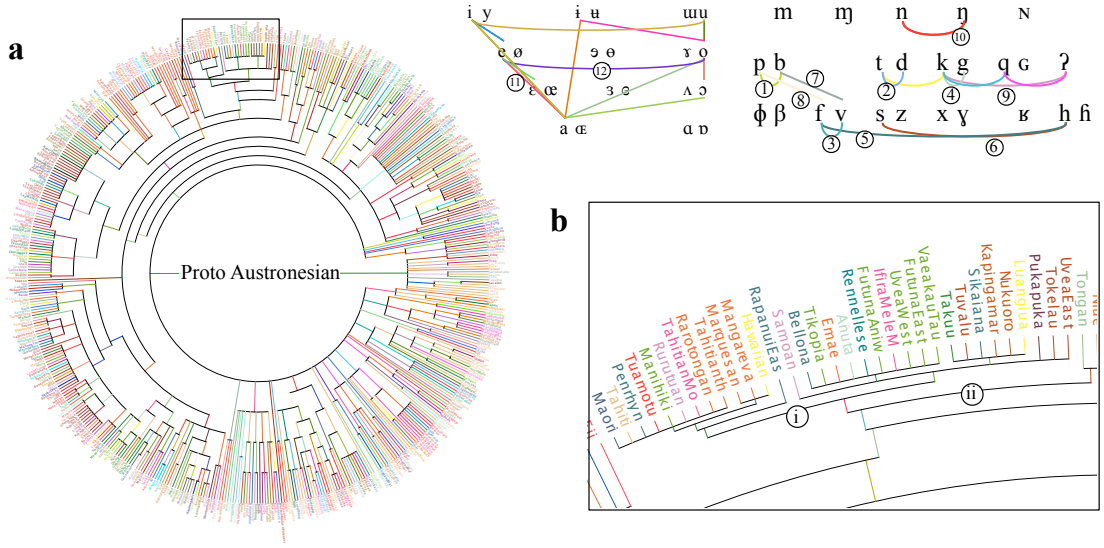


Figure 3.2. (a) The Austronesian phylogenetic tree used in our analyses. The colors encode the most prominent sound change along each branch, as inferred automatically by our system. Refer to the IPA chart on the top right for the key to the coding of these changes. Note that the standard organization of the IPA chart into columns and rows according to place, manner, height and backness is only for visualization purposes: this information was not encoded in the model in this experiment, showing that the model can recover realistic cross-linguistic sound change trends. All of the arcs in this figure correspond to sound changes frequently used by historical linguists: sonorizations /p/ > /b/ (1) and /t/ > /d/ (2); voicing changes (3,4); debuccalizations /f/ > /h/ (5) and /s/ > /h/ (6); spirantizations /b/ > /v/ (7) and /p/ > /f/ (8); changes of place of articulation (9,10); vowel changes in height (11) and backness (12) [31]. While this visualization depicts sound changes as undirected arcs, the sound changes are actually represented with directionality in our system. (b) Zooming in a portion of the Oceanic languages, where the Nuclear Polynesian family (i) and Polynesian family (ii) are visible. Several attested sound changes such as debuccalization to Maori and place of articulation change /t/ > /k/ to Hawaiian [57] are successfully localized by the system.

to protolanguages that are ancestors of those languages. We also assume that the words in modern languages have been identified as belonging to cognate sets, where all words in a cognate set are descended from a common ancestor. A given cognate set can contain words from only a subset of modern languages. The goal is to reconstruct the word in each protolanguage that corresponds to each cognate set.

3.2.3 Generative process

We start this section by introducing some notation. Let τ be a phylogenetic tree of languages, where each language is linked to the languages that descended from it. In such a tree, the modern languages, whose word forms will be observed, are the leaves of τ. The most recent common ancestor of these modern languages is the root of τ. All internal nodes of the tree (including the root) are protolanguages whose word forms are not observed. Let L denote all languages, modern and otherwise. All word forms are assumed to be strings Σ^* in the International Phonetic Alphabet (IPA).

We assume that word forms evolve along the branches of the tree τ. However, it is usually not the case that a word belonging to each cognate set exists in each modern language—words are lost or replaced over time, meaning that words that appear in the root languages may not have cognate descendants in the languages at the leaves of the tree. Formally, we assume there to be a known list of C cognate sets. For each $c \in \{1, \ldots, C\}$ let $L(c)$ denote the subset of modern languages that have a word form in the c-th cognate set. For each set $c \in \{1, \ldots, C\}$ and each language $\ell \in L(c)$, we denote the modern word form by $w_{c\ell}$. For cognate set c, only the minimal subtree $\tau(c)$ containing $L(c)$ and the root is relevant to the reconstruction inference problem for that set.

Our model of sound change is based on a generative process defined on this tree. From a high-level perspective, the generative process is quite simple. Let c be the index of the current cognate set, with topology $\tau(c)$. First, a word is generated for the root of $\tau(c)$ using an (initially unknown) root language model (i.e. a probability distribution over strings). The words that appear at other nodes of the tree are generated incrementally, using a branch-specific distribution over changes in strings to generate each word from the word in the language that is its parent in $\tau(c)$. While this distribution differs across branches of the tree, making it possible to estimate the pattern of changes involved in the transition from one language to another, it remains the same for all cognate sets, expressing changes that apply stochastically to all words. The central challenge in accurately reconstructing protolanguages is obtaining good estimates of these branch-specific distributions, a problem that we consider in the next section.

In the remainder of this section, we clarify the exact form of the conditional distributions over string changes, the distribution over strings at the root, and the parameterization of this process.

3.2.4 Distributions

The conditional distributions over pairs of evolving strings are specified using a *lexicalized stochastic string transducer* [92], as illustrated in Figure 3.3 (b-d). We call this process the *mutation Markov chain*. We describe the model in the case where each operation is conditioned on only short *context* (a single character on the parent string, a single previous character on the descendent string), but it can be easily extended to richer contexts.

Consider a language ℓ' evolving to ℓ for cognate set c. Assume we have a word form $x = w_{cl'}$. The generative process for producing $y = w_{cl}$ works as follows. First, we consider x to be composed of characters $x_1 x_2 \dots x_n$, with the first and last being a special boundary symbol $x_1 = \# \in \Sigma$ which is never deleted, mutated, or created. The process generates $y = y_1 y_2 \dots y_n$ in n chunks $y_i \in \Sigma^*, i \in \{1, \dots, n\}$, one for each x_i. The y_i's may be a single character, multiple characters, or even empty. In the example shown in Figure 3.3, all three of these cases occur.

To generate y_i, we define a *mutation Markov chain* that incrementally adds zero or more characters to an initially empty y_i. First, we decide whether the current phoneme in the top word $t = x_i$ will be deleted, in which case $y_i = \epsilon$ as in the example of /s/ being deleted shown in Figure 3.3. If t is not deleted, we chose a single substitution character in the bottom word. This is the case both when /a/ is unchanged and when /ŋ/ substitutes to /n/. We write $\mathscr{S} = \Sigma \cup \{\zeta\}$ for this set of outcomes, where ζ is the special outcome indicating deletion. Importantly, the probabilities of this multinomial can depend on both the previous character generated so far (i.e. the rightmost character p of y_{i-1}) and the current character in the previous generation string (t). This multinomial decision acts as the initial distribution of the mutation Markov chain.

We consider insertions only if a deletion was not selected in the first step. Here, we draw from a multinomial over $\mathscr{S}$, where this time the special outcome ζ corresponds to stopping insertions, and the other elements of $\mathscr{S}$ correspond to symbols that are appended to y_i. In this case, the conditioning environment is $t = x_i$ and the current rightmost symbol p in y_i. Insertions continue until ζ is selected. In the example shown in Figure 3.3, we follow the substitution of /ŋ/ to /n/ with an insertion of /g/, followed by a decision to stop that y_i. We will use $\theta_{S,t,p,\ell}$ and $\theta_{I,t,p,\ell}$ to denote the probabilities over the substitution and insertion decisions in the current branch $\ell' \to \ell$.

A similar process generates the word at the root ℓ of a tree, treating this word as a single string y_1 generated from a dummy ancestor $t = x_1$. In this case, only the insertion prob-

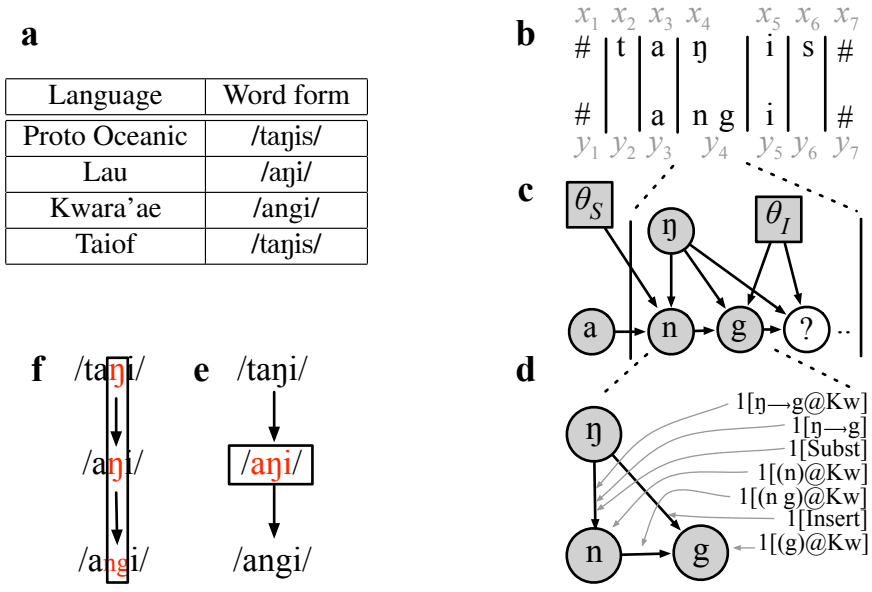


Figure 3.3. (a) A cognate set from the Austronesian dataset. All word forms mean *to cry*. (b-d) The mutation model used in this chapter. (b) The mutation of POc /taŋis/ to Kw. /angi/. (c) Graphical model depicting the dependencies among variables in one step of the mutation Markov chain. (d) Active features for one step in this process. (e-f) Comparison of two inference procedures on trees: Single sequence resampling (e) draws one sequence at a time, conditioned on its parent and children, while ancestry resampling (f) draws an aligned slice from all words simultaneously. In large trees, the latter is more efficient than the former.

abilities matter, and we separately parameterize these probabilities with $\theta_{R,t,p,\ell}$. There is no actual dependence on t at the root, but this formulation allows us to unify the parameterization, with each $\theta_{\omega,t,p,\ell} \in \mathbb{R}^{|\Sigma|+1}$ where $\omega \in \{R, S, I\}$. The probability distributions represented by θ are thus the full specification of our probabilistic generative model.

3.2.5 Parameterization

Instead of directly estimating the transition probabilities of the mutation Markov chain (which could be done, in principle, by taking them to be the parameters of a collection of multinomial distributions) we reparameterize them using a curved exponential family. This model specifies a distribution over transition probabilities via natural parameters defined on a set of features that describe properties of the sound changes involved. These features provide a more coherent representation of the transition probabilities, capturing regularities in sound changes that reflect the underlying linguistic structure.

We used the following feature templates:

OPERATION identifies whether an operation in the mutation Markov chain is an insertion, a deletion, a substitution, a self-substitution (i.e. of the form $x \to y, x = y$), or the end of an insertion event. Examples in Figure 3.3 (d): $\mathbf{1}$[Subst] and $\mathbf{1}$[Insert].

MARKEDNESS consists of language-specific n-gram indicator functions for all symbols in Σ. Only unigram and bigram features are used for computational reasons. Examples in Figure 3.3 (d): the bigram indicator $\mathbf{1}$[(n g)@Kw] (Kw stands for Kwara'ae, a language of the Solomon Islands), the unigram indicators $\mathbf{1}$[(n)@Kw] and $\mathbf{1}$[(g)@Kw].

FAITHFULNESS consists of indicators for mutation events of the form $\mathbf{1}[x \to y]$, where $x \in \Sigma$, $y \in \mathscr{S}$. Examples: $\mathbf{1}$[ŋ → n], $\mathbf{1}$[ŋ → n@Kw].

Feature templates similar to these can be found for instance in [16] and [8], in the context of string-to-string transduction models used in computational linguistics. This approach to specifying the transition probabilities produces an interesting connection to stochastic optimality theory [23, 97], where a logistic regression model mediates markedness and faithfulness of the production of an output form from an underlying input form.

Data sparsity is a significant challenge in protolanguage reconstruction. While the experiments we present here use an order of magnitude more languages than previous computational approaches, the increase in observed data also brings with it additional unknowns

in the form of intermediate protolanguages. Since there is one set of parameters for each language, adding more data is not sufficient for increasing the quality of the reconstruction; it is important to share parameters across different branches in the tree in order to benefit from having observations from more languages. We used the following technique to address this problem: we augment the sufficient statistics to include the current language (or language at the bottom of the current branch) and use a single, global natural parameter vector instead of a set of branch-specific natural parameters. Generalization across branches is then achieved by using features that *ignore* ℓ, while branch-specific features depend on ℓ. For instance, in Figure 3.3 (d), $\mathbf{1}[\text{ŋ} \rightarrow \text{n}]$ is an example of a universal (global) feature shared across all branches while $\mathbf{1}[\text{ŋ} \rightarrow \text{n@Kw}]$ is branch-specific. Similarly, all of the features in OPERATION, MARKEDNESS and FAITHFULNESS have universal and branch-specific versions.

Using these features and parameter sharing, the Phylogenetic Exponential Family (PEF) defines the transition probabilities of the mutation process and root language model as follows:

$$\theta_{\omega,t,p,\ell} = \theta_{\omega,t,p,\ell}(\xi;\lambda) = \frac{\exp\{\langle \lambda, f(\omega,t,p,\ell,\xi)\rangle\}}{Z(\omega,t,p,\ell,\lambda)} \times \mu(\omega,t,\xi), \tag{3.1}$$

where $\xi \in \mathscr{S}$, $f : \{S,I,R\} \times \Sigma \times \Sigma \times L \times \mathscr{S} \rightarrow \mathbb{R}^k$ is the sufficient statistics or feature function, $\langle \cdot,\cdot \rangle$ denotes inner product and $\lambda \in \mathbb{R}^k$ is the natural parameter vector. Here, k is the dimensionality of the feature space of the PEF. Z and μ are the normalization function and base measure respectively:

$$Z(\omega,t,p,\ell,\lambda) = \sum_{\xi' \in \mathscr{S}} \exp\{\langle \lambda, f(\omega,t,p,\ell,\xi')\rangle\}$$

$$\mu(\omega,t,\xi) = \begin{cases} 0 \text{ if } \omega = S, t = \#, \xi \neq \# \\ 0 \text{ if } \omega = R, \xi = \zeta \\ 0 \text{ if } \omega \neq R, \xi = \# \\ 1 \text{ o.w.} \end{cases}$$

Here, μ is used to handle boundary conditions.

3.2.6 Inadequacy of CTMC marginals for diachronic phonology

Before moving on to the experiments, we explain in this section the detailed motivations for using PEF over the stochastic process approach of Section 2.8. The motivations can be categorized as follows:

Computational: While finding the marginals of a finite state space Continuous Time Markov Chain (CTMC) is computationally easy (using the matrix exponential), finding the marginals of a countably infinite state space CTMC (such as a CTMC over strings) is more complicated. These marginals take the form of weighted transducers, but computing the weights of these transducers is difficult except in the simplest cases. These weights are only expressed implicitly as the solution of a Partial Differential Equation (PDE). Numerical methods have been used to solve them, but with mixed empirical success [60].

Model limitations: Beyond these computational issues, models based on CTMCs are inadequate to represent sound changes. This is because related sound changes often occur in sequence within a short time. For example, in a single branch of a phylogenetic tree, both the change $x > y$ and $y > z$ might occur, but the change $x > z$ may or may not occur, depending on whether the sequence at hand is a *push chain* or a *drag chain* (i.e. depending on the ordering of the two changes). Only the latter can be modeled as the marginal of a (stationary) CTMC.[1] We show in Chapter 6 that non Markovian stochastic processes can represent sound changes, but PEFs have the advantage of simplicity over these more complicated models.

3.3 Computational aspects

The generative model introduced in the previous section sets us up with two problems to solve: estimating the values of the parameters characterizing the distribution on sound changes on each branch of the tree, and inferring the optimal values of the strings representing words in the unobserved protolanguages. Section 3.3.1 introduces the full objective function that we need to optimize in order to estimate these quantities. Section 4 describes the Monte Carlo Expectation-Maximization algorithm we used for solving the learning problem. We present the algorithm for inferring ancestral word forms in Section 3.3.4.

3.3.1 Full objective function

The generative model specified in Section 3.2 defines an objective function that we can optimize in order to find good protolanguage reconstructions. This objective function

[1] Note also that push chains, the only variant that can be modeled by CTMC, are controversial [31].

takes the form of a regularized log-likelihood, combining the probability of the observed languages with additional constraints intended to deal with data sparsity. This objective function can be written concisely if we let $\mathbb{P}_\lambda(\cdot), \mathbb{P}_\lambda(\cdot|\cdot)$ denote the root and branch probability models described in Section 3.2.4 (with transition probabilities given by the PEF), $I(c)$, the set of internal (non-leaf) nodes in $\tau(c)$, $\mathrm{pa}(\ell)$, the parent of language ℓ, $\mathrm{r}(c)$, the root of $\tau(c)$ and $W(c) = (\Sigma^*)^{|I(c)|}$. The full objective function is then

$$\mathrm{Li}(\lambda) = \sum_{c=1}^{C} \log \sum_{\vec{w} \in W(c)} \mathbb{P}_\lambda(w_{c,\mathrm{r}(c)}) \prod_{\ell \in I(c)} \mathbb{P}_\lambda(w_{c,\ell} | w_{c,\mathrm{pa}(\ell)}) - \frac{||\lambda||_2^2}{2\sigma^2} \tag{3.2}$$

where the second term is a standard L^2 regularization penalty intended to reduce overfitting due to data sparsity (we used $\sigma^2 = 1$) [28]. The goal of learning is to find the value of λ, the natural parameters of the PEF for the transition probabilities, that maximizes this function.

3.3.2 A Monte Carlo Expectation-Maximization algorithm

Optimization of the objective function given in Equation 3.2 is done using a Monte Carlo variant of the Expectation-Maximization (EM) algorithm [12]. This algorithm breaks down into two steps, an E step in which the objective function is approximated and an M step in which this approximate objective function is optimized. The M step is convex and computed using L-BFGS [52] but the E step is intractable. See Chapter 4 for a detailed discussion on the E step. The M step is described in the next section.

3.3.3 M step: Convex optimization of the approximate objective

In the M step, we update the natural parameters λ of the PEF defined in Equation 3.1. Let $C = (\omega, t, p, \ell)$ denote local transducer contexts from the space $\mathbf{C} = \{S, I, R\} \times \Sigma \times \Sigma \times L$ of all such contexts. Let $N(C, \xi)$ be the expected number of times the transition ξ was used in context C in the preceding E-step. Given these sufficient statistics, the estimate of λ is given by optimizing the expected complete (regularized) log-likelihood $\mathcal{O}(\lambda)$ derived from the original objective function given in Equation 3.1,

$$\mathcal{O}(\lambda) = \sum_{C \in \mathbf{C}} \sum_{\xi \in \mathscr{S}} N(C, \xi) \Big[\langle \lambda, f(C, \xi) \rangle - \log \sum_{\xi'} \exp\{\langle \lambda, f(C, \xi') \rangle\} \Big] - \frac{||\lambda||^2}{2\sigma^2}.$$

We use L-BFGS [52] to optimize this convex objective function. L-BFGS requires the partial derivatives

$$\begin{aligned}\frac{\partial \mathcal{O}(\lambda)}{\partial \lambda_j} &= \sum_{C \in \mathbf{C}} \sum_{\xi \in \mathscr{S}} N(C, \xi) \Big[f_j(C, \xi) - \sum_{\xi'} \theta_C(\xi'; \lambda) f_j(C, \xi') \Big] - \frac{\lambda_j}{\sigma^2} \\ &= \hat{F}_j - \sum_{C \in \mathbf{C}} \sum_{\xi \in \mathscr{S}} N(C, \cdot) \theta_C(\xi'; \lambda) f_j(C, \xi') - \frac{\lambda_j}{\sigma^2},\end{aligned}$$

where $\hat{F}_j = \sum_{C \in \mathbf{C}} \sum_{\xi \in \mathscr{S}} N(C, \xi) f_j(C, \xi)$ is the empirical feature vector and $N(C, \cdot) = \sum_{\xi} N(C, \xi)$ is the number of times context C was used. $\hat{F}_j$ and $N(C, \cdot)$ do not depend on λ and thus can be precomputed at the beginning of the M-step, thereby speeding up each L-BFGS iteration.

3.3.4 Ancestral word form reconstruction

In the E step described in the preceding section, a posterior distribution π over ancestral sequences given observed forms is approximated by a collection of samples $X_1, X_2, \ldots X_S$. In this section, we describe how this distribution is summarized to produce a single output string for each cognate set.

This algorithm is based on a fundamental Bayesian decision theoretic concept: Bayes estimators. Given a loss function over strings $\text{Loss} : \Sigma^* \times \Sigma^* \to [0, \infty)$, an estimator is a Bayes estimator if it belongs to the set:

$$\operatorname*{argmin}_{x \in \Sigma^*} \mathbb{E}^{\pi} \text{Loss}(x, X) = \operatorname*{argmin}_{x \in \Sigma^*} \sum_{y \in \Sigma^*} \text{Loss}(x, y) \pi(y).$$

Bayes estimators are not only optimal within the Bayesian decision framework, but also satisfy frequentist optimality criteria such as minimaxity and admissibility [77]. In our case, the loss we used is the Levenshtein [47] distance, denoted $\text{Loss}(x, y) = \text{Lev}(x, y)$ (we discuss this choice in more detail in Section 3.4.2).

Since we do not have access to π, but rather to an approximation based on samples, the objective function we use for reconstruction rewrites as follows:

$$\begin{aligned}\operatorname*{argmin}_{x \in \Sigma^*} \sum_{y \in \Sigma^*} \text{Lev}(x, y) \pi(y) &\approx \operatorname*{argmin}_{x \in \Sigma^*} \frac{1}{S} \sum_{s=1}^{S} \text{Lev}(x, X_s) \\ &= \operatorname*{argmin}_{x \in \Sigma^*} \sum_{s=1}^{S} \text{Lev}(x, X_s).\end{aligned}$$

The raw samples contain both derivations and strings for all protolanguages, whereas we are only interested in reconstructing words in a single protolanguage. This is addressed by marginalization, which is done in sampling representations by simply discarding the irrelevant information. Hence, the random variables X_s in the above equation can be viewed as being string-valued random variables.

Note that the optimum is not changed if we restrict the minimization to be taken on $x \in \Sigma^*$ such that $m \leq |x| \leq M$ where $m = \min_s |X_s|, M = \max_s |X_s|$. However, even with this simplification, optimization is intractable. As an approximation, we considered only strings built by at most k contiguous substrings taken from the word forms in $X_1, X_2, \ldots, X_S$. If $k = 1$, then it is equivalent to taking the min over $\{X_s : 1 \leq s \leq S\}$. At the other end of the spectrum, if $k = S$, it is exact. This scheme is exponential in k, but since words are relatively short, we found that $k = 2$ often finds the same solution as higher values of k.

3.4 Experiments in diachronic phonology

3.4.1 Overview

To test our system, we applied it to a large scale database containing information about cognate sets for 637 Austronesian languages (135,845 lexical items) [26]. We show in Figure 3.2 that the system can learn a large variety of realistic sound changes across the Austronesian family. The Austronesian database can also be used to quantitatively evaluate the performance of our system. Compared to phylogeny estimation systems, which can generally be evaluated only using generated data, the performance of sequence reconstruction tools can be quantitatively assessed on naturally occurring data by holding information about some modern languages in reserve and attempting to reconstruct the words in those languages. The Levenshtein distance [47] between the held-out forms and the reconstructed forms then provides a measure of the number of errors in these reconstructions. We used this strategy to establish the superiority of our method over previous reconstruction algorithms [70, 44], to show that using more languages helped reconstruction (see Figure 3(a)), and to demonstrate the stability of our learning procedure (see Section 3.4.3).

To demonstrate the utility of this kind of large-scale reconstruction of protolanguages, we used the output of our system to investigate an open question in historical linguistics. This

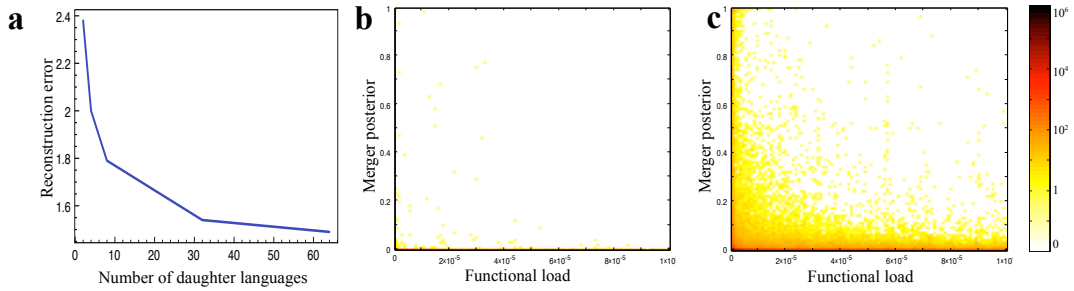


Figure 3.4. Consequences of large-scale reconstruction of protolanguages. (a) Reconstruction error is a quantitative way of measuring the increase in resolution obtained by adding more languages. Here we show the averaged protolanguage reconstruction error (as measured by the Levenshtein distance [47] to the manual reconstruction of Proto-Oceanic [1]) as a function of the number of daughter languages used by the model. Languages are added in decreasing order of cognate overlap. All improvements are statistically significant except for going from 32 to 64 languages. (b-c) Increasing the number of languages we can reconstruct also allows us to answer questions in historical linguistics, such as the effect of functional load on the probability of merging two sounds. The plots shown are heat maps where the color encodes the log of the number of sound changes that fall into a given 2-dimensional bin. Each sound change $x > y$ is encoded as pair of numbers in the unit interval, (l, m), where l is the estimated functional load of the pair and m is the posterior fraction of the phoneme x that undergo a change to y (so that $m = 1$ means unconditional merger, $m = 0$ means no merger occur for that pair x, y, and numbers in between show conditional mergers). (b) Shows the data obtained from analyzing just four languages, as in previous linguistic analyses [41]. (c) Shows the results of using 637 languages. In both cases the sound changes are identified by our automated system.

is the question of whether the amount of information provided by a sound is related to the probability that sound will change over time. The hypothesis that these two variables are related was put forward by André Martinet in 1955 [58], based purely on theoretical considerations. Later, Robert King formalized Martinet's conjecture and evaluated its empirical support based on four languages [41]. He concluded that the "hypothesis [...] seems to be not much use [...]". This conclusion was criticized by several authors [32, 83] on the basis of the small number of languages and sound changes considered. However, no positive counter-evidence was provided by his critics. To produce such evidence, we collected sound change statistics from the hundreds of languages in the Austronesian database.

The main statistic that Martinet identified as potentially affecting sound change is called

functional load [58]. This statistic formalizes the amount of information lost when a language undergoes a certain sound change. Intuitively, the functional load of a pair of phonemes reflects the proportion of words that are discriminated by those phonemes (see Section 3.4.6 for a formal definition). If two phonemes only appear in words that are differentiated from one another by at least one other sound, then one can argue that no information is lost if those phonemes merge together, because no new ambiguous forms can be created by the merger. The functional load of the phonemes increases as the number of ambiguous words that would be created by merging them increases. Figure 3.4 shows the relationship between functional load and the probability of a merger in our reconstruction of the changes in the Austronesian database. To convey the amount of noise one could expect from a study with the number of language that King previously used, we first show in Figure 3.4(a) the heat map visualization for four languages. Next, we show the same plot for 637 languages in (b). Only in this latter setup is structure clearly visible: most of the points with high merger posterior probability can be seen to have comparatively low functional load.

This demonstration illustrates the strength of our system: automatic, *en masse* reconstruction of accurate protolanguage word forms and sound change histories. The large-scale analysis of the properties of ancient languages that this system produces goes far beyond the capabilities of any previous automated system, and would require significant amounts of manual effort by linguists. Furthermore, the system is in no way restricted to applications like assessing the effects of functional load: It can be used as a tool to investigate a wide range of questions about the structure and dynamics of languages, and about the lives of our ancestors.

3.4.2 Evaluation metric

Evaluation of all methods was done by computing the Levenshtein distance [47] (uniform-cost edit distance) between the reconstruction produced by each method and the reconstruction produced by linguists. The Levenshtein distance is the minimum number of substitutions, insertions, or deletions of a phoneme required to transform one word to another. While the Levenshtein distance misses important aspects of phonology (all phoneme substitutions are not equal, for instance), it is parameter-free and still correlates to a large extent with linguistic quality of reconstruction. It is also superior to held-out log-likelihood, which fails to penalize errors in the modeling assumptions, and to measuring the percentage of perfect reconstructions, which ignores the degree of correctness of

each reconstructed word. We averaged this distance across reconstructed words to report a single number for each method. We show in Table 3.2 the average word length in each corpus. The statistical significance of all performance differences are assessed using a paired t-test with significance level of 0.05.

3.4.3 Evaluating system performance

We used the Austronesian Basic Vocabulary Database [26] as the basis for a series of experiments used to evaluate the performance of our system and the factors relevant to its success. The database includes partial cognacy judgments and IPA transcriptions, as well as a few reconstructed protolanguages. A reconstruction of Proto-Oceanic (POc) originally developed by [1] using the comparative method was the basis for evaluation.

We used the cognate information provided in the database, automatically constructing a global tree and set of subtrees from the cognate set indicator matrix $M(\ell, c) = \mathbf{1}[\ell \in L(c)], c \in \{1, \ldots, C\}, \ell \in L$. For constructing the global tree, we used bootstrapped neighbor joining [22] with a metric based on *cognate overlap*, $d_c(\ell_1, \ell_2) = \sum_{c=1}^{C} M(\ell_1, c)M(\ell_2, c)$. We bootstrapped 1000 samples and formed an accurate (90%) consensus tree. The tree obtained is not binary, but the AR inference algorithm scales linearly in the branching factor of the tree, meaning that approximate inference remained possible (in contrast, SSR scales exponentially [55]).

The first claim we verified experimentally is that having more observed languages aids reconstruction of protolanguages. To test this hypothesis we added observed modern languages in increasing order of distance d_c to the target reconstruction of POc so that the languages that are most useful for POc reconstruction are added first. This prevents the effects of adding a close language after several distant ones being confused with an improvement produced by increasing the number of languages.

The results are reported in Figure 3.5 (a). They confirm that large-scale inference is desirable for automatic protolanguage reconstruction: reconstruction improved statistically significantly with each increase except from 32 to 64 languages, where the average edit distance improvement was 0.05.

We then conducted a number of experiments intended to assess the robustness of the system, and to identify the contribution made by different factors it incorporates. First, we ran the system with 20 different random seeds to assess the stability of the solutions

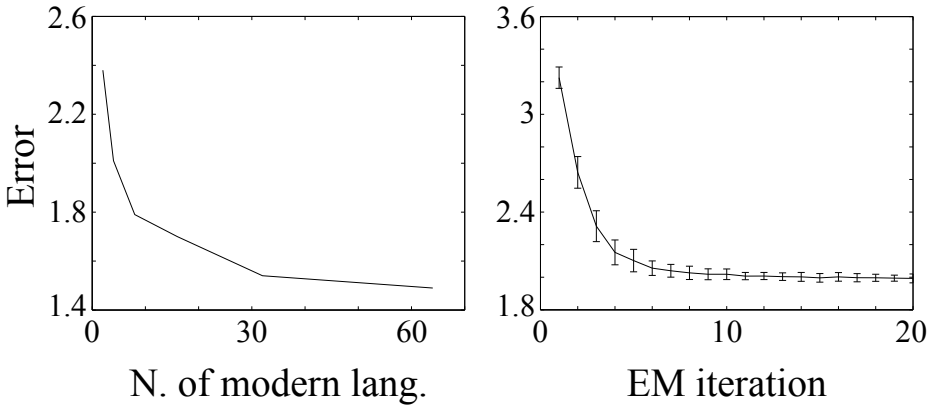


Figure 3.5. Left: Mean distance to the target reconstruction of POc as a function of the number of modern languages used by the inference procedure. Right: Mean distance and confidence intervals as a function of the EM iteration, averaged over 20 random seeds and ran on four languages.

found. In each case, learning was stable and accuracy improved during training. See Figure 3.5 (b).

Next, we found that all of the following ablations significantly hurt reconstruction: using a flat tree (in which all languages are equidistant from the reconstructed root and from each other) instead of the consensus tree, dropping the markedness features, dropping the faithfulness features, and disabling sharing across branches. The results of these experiments are shown in Table 3.1.

For comparison, we also included in the same table the performance of a semi-supervised system trained by K-fold validation. The system was run $K = 5$ times, with $1 - K^{-1}$ of the POc words given to the system as observations in the graphical model for each run. It is semi-supervised in the sense that target reconstructions for many internal nodes are not available in the dataset (for example the common ancestor of Kwara'ae (Kw.) and Lau in Figure 3.6 (b)), so they are still not filled.[2]

Figure 3.6 (b) shows the results of a concrete run over 32 languages, zooming in on a pair of the Solomonic languages and the cognate set from Figure 3.3 (a). In the example shown, the reconstruction is as good as the ORACLE (described in Section 3.4.4), though

[2]We also tried a fully supervised system where a flat topology is used so that all of these latent internal nodes are avoided; but it did not perform as well—this is consistent with the -Topology experiment of Table 3.1.

Condition	Edit dist.
Unsupervised full system	1.87
-FAITHFULNESS	2.02
-MARKEDNESS	2.18
-Sharing	1.99
-Topology	2.06
Semi-supervised system	1.75

Table 3.1. Effects of ablation of various aspects of our unsupervised system on mean edit distance to POc. -Sharing corresponds to the restriction to the subset of the features in OPERATION, FAITHFULNESS and MARKEDNESS that are branch-specific, -Topology corresponds to using a flat topology where the only edges in the tree connect modern languages to POc. The semi-supervised system is described in the text. All differences (compared to the unsupervised full system) are statistically significant.

off by one character (the final /s/ is not present in any of the 32 inputs and therefore is not reconstructed). In (a), diagrams show, for both the global and the local (Kwara'ae) features, the expectations of each substitution superimposed on an IPA sound chart, as well as a list of the top changes. Darker lines indicate higher counts. This run did not use natural class constraints, but it can be seen that linguistically plausible substitutions are learned. The global features prefer a range of voicing changes, manner changes, adjacent vowel motion, and so on, including mutations like /s/ to /h/ which are common but poorly represented in a naive attribute-based natural class scheme. On the other hand, the features local to the language Kwara'ae pick out the subset of these changes which are active in that branch, such as /s/ > /t/ fortition.

3.4.4 Comparison against other methods

The first competing method, PRAGUE was introduced in [70]. In this method, the word forms in a given protolanguage are reconstructed using a Viterbi multi-alignment between a small number of its descendant languages. The alignment is computed using hand-set parameters. Deterministic rules characterizing changes between pairs of observed languages are extracted from the alignment when their frequency is higher than a threshold, and a proto-phoneme inventory is built using linguistically motivated rules and parsimony. A reconstruction of each observed word is first proposed independently for each language. If at least two reconstructions agree, a majority vote is taken, otherwise no reconstruction is proposed. This approach has several limitations. First, it is not tractable for larger

trees, since the time complexity of their multi-alignment algorithm grows exponentially in the number of languages. Second, deterministic rules, while elegant in theory, are not robust to noise: even in experiments with only four daughter languages, a large fraction of the words could not be reconstructed.

Since PRAGUE does not scale to large datasets, we built a second, more tractable baseline. This new baseline system, CENTROID, computes the centroid of the observed word forms in Levenshtein distance. Let $\mathrm{Lev}(x, y)$ denote the Levenshtein distance between word forms x and y. Ideally, we would like the baseline system to return:

$$\operatorname*{argmin}_{x \in \Sigma^*} \sum_{y \in O} \mathrm{Lev}(x, y),$$

where $O = \{y_1, \ldots, y_{|O|}\}$ is the set of observed word forms. This objective function is motivated by Bayesian decision theory [77], and shares similarity to the more sophisticated Bayes estimator described in Section 3.3.4. However it replaces the samples obtained by MCMC sampling by the set of observed words. Similarly to the algorithm of Section 3.3.4, we also restrict the minimization to be taken on $x \in \Sigma(O)^*$ such that $m \leq |x| \leq M$ and to strings built by at most k contiguous substrings taken from the word forms in O, where $m = \min_i |y_i|, M = \max_i |y_i|$ and $\Sigma(O)$ is the set of characters occurring in O. Again, we found that $k = 2$ often finds the same solution as higher values of k. The difference was in all the cases not statistically significant, so we report the approximation $k = 2$ in what follows.

We also compared against an oracle, denoted ORACLE, which returns

$$\operatorname*{argmin}_{y \in O} \mathrm{Lev}(y, x^*),$$

where x^* is the target reconstruction. We will denote it by ORACLE. This is superior to picking a single closest language to be used for all word forms, but it is possible for systems to perform better than the oracle since it has to return one of the observed word forms. Of course, this scheme is only available to assess system performance on held-out, it cannot make new predictions.

We performed the comparison against [70] on the same dataset and experimental conditions as used in the original paper (see Table 3.2). The Proto-Malayo-Javanic (PMJ) dataset was compiled by [69], who also reconstructed the corresponding protolanguage. Since PRAGUE is not guaranteed to return a reconstruction for each cognate set, only 55 word forms could be directly compared to our system. We restricted comparison to this

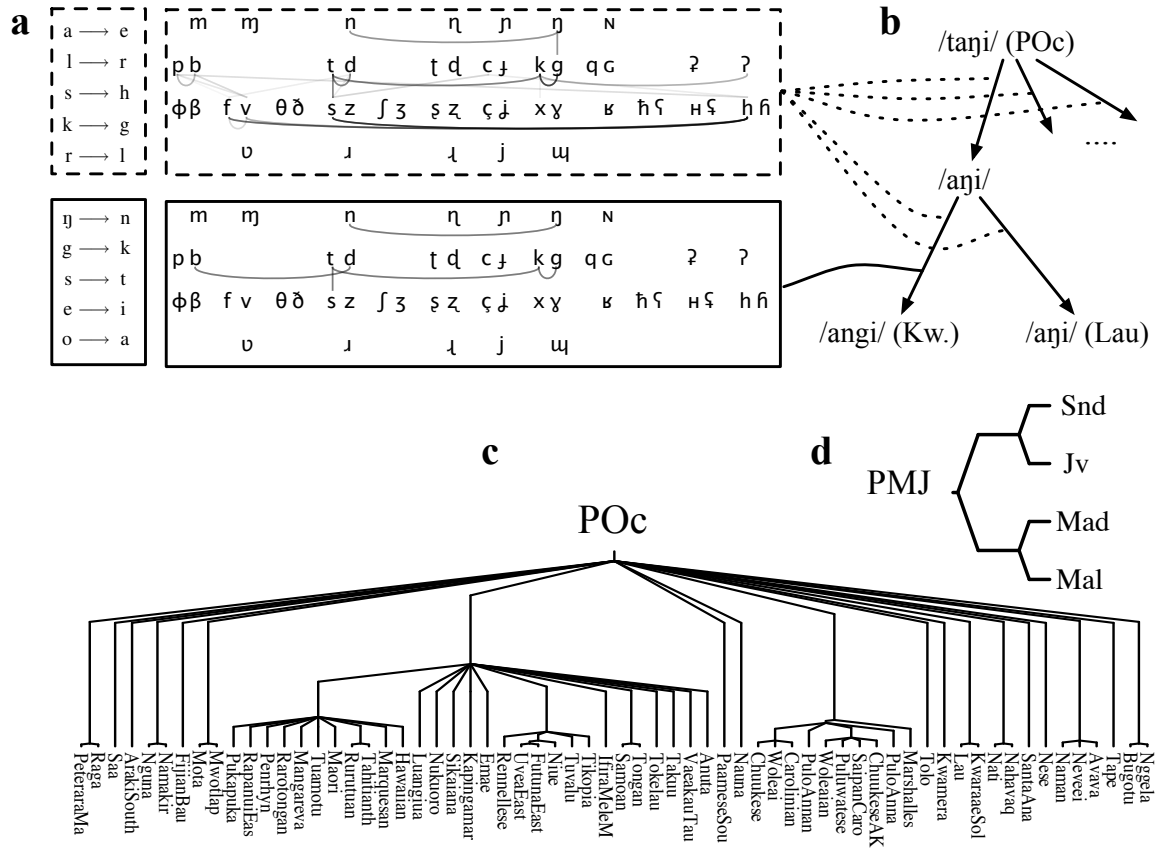


Figure 3.6. (a) A visualization of two learned faithfulness parameters: on the top, from the universal features, on the bottom, for one particular branch. Each pair of phonemes have a link with grayscale value proportional to the expectation of a transition between them. The five strongest links are also included at the right. (b) A sample taken from our POc experiments (see text). (c-d) Phylogenetic trees for two language families: Malayo-Javanic and Oceanic.

Comparison	PRAGUE	CENTROID	FL-SMALL	FL-BIG
Protolanguage	PMJ	POc	PAn	PAn
Heldout (prop.)	79 (1.0)	243 (1.0)	162 (1.0)	261 (1.0)
Modern languages	4	70	4	541
Cognate sets	179	1321	241	6323
Observed words	470	10783	570	54347
Mean word length	5.0	4.5	4.4	4.8

Table 3.2. Experimental setup: number of held-out proto-word from (absolute and relative), of modern languages, cognate sets and total observed words.

subset of the data. This favors PRAGUE since the system only proposes a reconstruction when it is certain. Still, our system outperformed PRAGUE, with an average distance of 1.60 compared to 2.02 for PRAGUE. The difference is marginally significant, $p = 0.06$, partly due to the small number of word forms involved.

To get a more extensive comparison, we considered the hybrid system that returns PRAGUE's reconstruction when possible and otherwise back off to the Sundanese (Snd.) modern form, then Madurese (Mad.), Malay (Mal.) and finally Javanic (Jv.) (the optimal back-off order). In this case, we obtained an edit distance of 1.86 against 2.33, a statistically significant difference.

Since PRAGUE does not scale well to large datasets, we also compared against ORACLE and CENTROID in a large-scale setting. Specifically, we compare to the experimental setup on 64 modern languages used to reconstruct POc described before. Encouragingly, while the system's average distance (1.49) does not attain that of the ORACLE (1.13), we significantly outperform the CENTROID baseline (1.79).

3.4.5 Incorporating prior linguistic knowledge

The model also supports the addition of prior linguistic knowledge. This takes the form of feature templates with more internal structure. We performed experiments with an additional feature template:

STRUCT-FAITHFULNESS is a structured version of FAITHFULNESS, replacing x and y with their natural classes $\text{NC}_\beta(x)$ and $\text{NC}_\beta(y)$ where β indexes types of classes, ranging over {manner, place, phonation, isOral, isCentral, height, backness, roundedness}. This feature set is reminiscent of the featurized representation of [43].

We compared the performance of the system with and without STRUCT-FAITHFULNESS to check if the algorithm can recover the structure of natural classes in an unsupervised fashion. We found that with 2 or 4 observed languages, FAITHFULNESS underperformed STRUCT-FAITHFULNESS, but for larger trees, the difference was not significant. FAITHFULNESS even slightly outperformed its structured cousin with 16 observed languages.

3.4.6 Functional load experiments

In this section, we describe how we used our system to investigate an open question in historical linguistic related to a statistical property of sound changes called *functional load*. Functional load is a concept introduced by Martinet [58]. Its purpose is to formalize the amount of information lost when a language undergoes a certain sound change. The functional load of a pair of phonemes interpolates between the concepts of complementary and contrastive distributions. Two phonemes are in *complementary distribution* if the environment in they occur is disjoint. They are in *contrastive distribution* if both are found in the same environment with a change in meaning. In one extreme, if two phonemes in complementary distribution are merged, then one can argue that no information is lost, because no new homophones are created by the merger.

The role of functional load in language change has been debated for a long time. Martinet conjectured that functional load should be considered as a factor affecting language change. His argument was based on the premise that communication is the main function of language, and therefore some measure of information should be considered as a potential factor determining the fitness of languages undergoing change. Martinet, however, did not perform an empirical evaluation—his argument was based purely on theoretical considerations.

Later, King [41] formalized Martinet's conjecture and evaluated its empirical support based on four languages. He concluded that "the functional load hypothesis [. . .] seems to be not much use [...]". This conclusion was criticized by Hockett [32] and Surendran et al. [83] on the basis of the small number of languages and sound changes considered. Indeed, within the four languages studied by King, only 16 mergers are found. However, no positive counter-evidence was provided by his critics.

We revisited the question of whether Martinet's conjecture is valid, using experiments performed on a considerably larger set of languages. The results support the view that

King's study was based on too few languages to be conclusive. Importantly, it provides statistical counter-evidence supporting the hypothesis formulated by Martinet.

In his 1967 article, King defined the following conjecture:

> WEAK POINT HYPOTHESIS: "If all else is equal, sound change is more likely to start within oppositions bearing low functional load than within oppositions bearing high functional loads."

To measure functional load quantitatively, we used the same estimator as the one used by King. King's definition is based on associating a *context vector* $\boldsymbol{c}_{x,\ell}$ to each phoneme and language. For a given language with $N(\ell)$ phoneme tokens, these context vectors are defined as follows: first, fix an enumeration order of all the contexts found in the corpus, where the context is defined as a pair of phonemes, one at the left and one at the right of a position. Element i in this enumeration will correspond to component i of the context vectors. Then, the value of component i in context vector $\boldsymbol{c}_{x,\ell}$ is set to be the number of time phoneme x occurred in context i and language l. Finally, King's definition of functional load $\mathrm{FL}_\ell(x, y)$ is the dot product of the two induced context vectors:

$$\mathrm{FL}_\ell(x, y) = \frac{1}{N(\ell)^2}\langle \boldsymbol{c}_{x,\ell}, \boldsymbol{c}_{y,\ell}\rangle = \frac{1}{N(\ell)^2}\sum_i \boldsymbol{c}_{x,\ell}(i) \times \boldsymbol{c}_{y,\ell}(i),$$

where the denominator is simply a normalization that insures $\mathrm{FL}_\ell(x, y) \leq 1$. Note that if x and y are in complementary distribution in language ℓ, then the two vectors $\boldsymbol{c}_{x,\ell}$ and $\boldsymbol{c}_{y,\ell}$ are orthogonal. The functional load is indeed zero in this case.

Earlier in this chapter, we have shown heat maps where the color encodes the log of the number of sound changes that fall into a given 2-dimensional bin. Each sound change $x > y$ is encoded as pair of numbers in the unit interval, $(\hat{l}, \hat{m})$, where $\hat{l}$ is an estimate of the functional load of the pair and $\hat{m}$ is the posterior fraction of the instances of the phoneme x that undergo a change to y. We now describe how $\hat{l}, \hat{m}$ were estimated. The posterior fraction $\hat{m}$ for the merger $x \to y$ between languages $\mathrm{pa}(\ell) \to \ell$ is easily computed from the same expected sufficient statistics used in Section 3.3.3 for parameter estimation:

$$\hat{m}_\ell(x \to y) = \frac{\sum_{p\in\Sigma} N(S, x, p, \ell, y)}{\sum_{p'\in\Sigma}\sum_{y'\in\Sigma} N(S, x, p', \ell, y')}.$$

The estimate of the functional load requires additional statistics, i.e. the expected context vectors $\hat{\boldsymbol{c}}_{x,\ell}$ and expected phoneme token counts $\hat{N}(\ell)$, but these can be readily extracted

from the output of the MCMC sampler. The estimate is then:

$$\hat{l}_\ell(x, y) = \frac{1}{\hat{N}(\ell)^2} \langle \hat{\boldsymbol{c}}_{x,\ell}, \hat{\boldsymbol{c}}_{y,\ell} \rangle.$$

Finally, the set of points used to construct the heat map is:

$$\left\{ \left(\hat{l}_{\mathrm{pa}(\ell)}(x, y), \hat{m}_\ell(x \to y) \right) : \ell \in L - \{\mathrm{root}\}, x \in \Sigma, y \in \Sigma, x \neq y \right\}.$$

3.5 Experiments in computational biology

One advantage of using the exponential families described in this chapter is that the method can be easily ported from the linguistic domain to biology. The only modification needed is to change the set of features.

Biological datasets give us a valuable opportunity to evaluate aspects of the models not tested by the linguistic data alone. In particular, for short sequences over a large alphabet, the main difficulty is ancestral reconstruction, and aligning the characters of the words is relatively easy. On the other hand, sequence alignment of biological sequences is much more challenging because of the large lengths and smaller alphabet sizes. There exists well established benchmarks for the task of Multiple Sequence Alignment (MSA), which we use in Section 3.5.2 to establish a new validation for the PEF model.

3.5.1 Features

All of the experiments in this section use the three features defined in Section 3.2.5, i.e. OPERATION, MARKEDNESS, and FAITHFULNESS, as well as the following two extra features based on proteomic expert knowledge:

HYDROPHOBIC-MODELING Large indels are less frequent in the hydrophobic core of globular proteins, since these stretches often correspond to the folded core. To detect hydrophobic cores, we used the heuristic described in [17]. The features themselves are indicator functions of insertion or deletion conjoined with whether the current location is being part of a hydrophobic stretch or not. A context of size five is needed to accommodate these features, but as described in Section 4.2, running time performance is not negatively impacted by these large contexts with AR, the inference algorithm described in the next

	Sum of Pairs score (SP)			
BAliBASE protein group	PEF	Handel [33]	Handel-EM [34]	Clustal [29]
test1/ref1	0.85	0.77	0.83	0.88
test1/ref2	0.77	0.69	0.71	0.79
test1/ref3	0.74	0.67	0.76	0.79

Table 3.3. Average SP scores for PEF and other MSA systems.

chapter. This contrast to previous approximate inference approaches (for example, SSR, also reviewed in the next chapter).

AFFINE-GAPS The length distribution of consecutive indels has a much fatter tail than the geometric tail predicted by simple models such as TKF91 (see Section 2.7). For example, a virus can insert long stretches in a genome in a single atomic event. This has been modeled using affine gap penalties, where "opening" a gap is more expensive than extending an already opened gap. This motivates having two sets of features, one for point indels, and one for indels of length more than one.

3.5.2 Multiple Sequence Alignment Results

We performed experiments on the task of protein multiple sequence alignment, for which the BAliBASE [86] dataset provides a standard benchmark. BAliBASE contains annotations created by biologists using secondary structure as alignment cues.

We report the *Sum of Pairs* (SP) scores, a standard evaluation score for this task. Sum of Pairs measures edge recall, i.e. the fraction of the annotated alignments links that are identified by the algorithm. Only a subset of the data, the "core blocks" is annotated and used to compute this score. See [15], for instance, for the details.

In all three BAliBASE subdirectories considered, the PEF's SP performance was higher than Handel's [33], a probabilistic system base on the TKF91 stochastic process. We also compared PEF to a subsequent version of Handel described in [34], where amino acid states are split to model rate heterogeneity, a feature that we did not implemented in our system—we foresee no conceptual difficulty in doing so, but we leave this for future work. Even without state splitting, ours system still outperformed [34] in two of the three subdirectories considered.

All probabilistic systems underperformed Clustal [29], a popular heuristic. One possibility for this gap is the numerous hidden variables introduced in generative approaches, resulting in a situation where exact inference in an inferior model may be preferable to approximate inference in a better model. Note however that Clustal's parameters were extensively tuned on BAliBASE, creating an unfair advantage over our unsupervised system.

One should keep in mind the main goal motivating PEF: performing ancestral sequence reconstructions. Since Clustal lacks derivation hidden variables, it cannot perform this task jointly with MSA. Conversely, we show in Chapter 5 that if one only cares about MSA performance, PEFs can be modified to achieve state-of-the art MSA performance.

Chapter 4

Inference in Phylogenetic Exponential Families

In the preceding chapter, we have delayed the discussion of an important point: how to compute or approximate the E step of the learning algorithm presented in Section . Recall that in the E step, the problem is to compute the posterior over the Phylogenetic Exponential Family (PEF) model given a collection of observed strings. This will be the topic of this chapter.

As it is often the case, the distribution associated with the posterior expectation is known only up to a normalization constant Z, which is hard to compute. Throughout this chapter, we assume that the parameters of the PEF have been fixed (in practice, these parameters come from the previous M step). In this setup, computing Z is equivalent to summing over all the accepting paths of a weighted automaton (we will review this fact in the next section). More generally, with fixed parameters, all the inference problems in PEF models (computing moments, sampling, estimating data likelihood) can be cast in the language of weighted transducers and automata [63].

The chapter is divided into two sections, covering exact and approximate inference, respectively. The contributions in exact inference are mostly theoretical: we show a new transducer formulation based on elementary operations on indexed matrices (tensor product and GL_n field operations), and use it to obtain drastically simplified transducer algorithms. As in previous exact transducer inference approaches, these algorithm have exponential running time in the number of taxa at the leaves, but they form a better foundation for the second section on approximate inference.

4.1 Exact inference

We start this section by stating formally what are the quantities we seek to compute. Given a set of parameters θ, a PEF model can be cast into a tree-shaped graphical model where nodes are string-valued random variables X representing a fragment of DNA, RNA or protein of a species. Edges denote evolution from one species to another, with conditional probabilities derived from the stochastic model described in Section 3.2.4. Usually, only the terminal nodes Y are observed, while the internal nodes are hidden. The interpretation is that the sequence at the root is the common ancestor of those at the terminal nodes, and it subsequently evolved in a branching process following the topology of the tree.

Usually, distributions in discrete graphical models are specified by exhaustive tables of real numbers. In this case, such representation is impossible since the state space, the set of string, is infinite. Instead, the distribution at the root of the graphical model is specified by a probabilistic automaton, and the conditional distributions for the edges are specified using string transducers.

The distribution of interest is $X|Y$, and more precisely, the goal is to compute $\mathbb{E}[T(X)|Y]$, where T is the sufficient statistic of PEF.

The initial step toward this goal are the same as in finite state-space directed graphical model: we first convert the directed graphical model into an equivalent factor graph, where indicator functions are added to the leaves (See Figure 4.1). In the factor graph, factors connected to one variable are weighted automata, and factors connected to two variables, weighted transducers. We explain this construction in more detail in the next section, reviewing the concepts of weighted automata and transducers at the same time.

As shown in [16], casting the problem into a factor graph makes it possible to generalize the sum-product algorithms to string-valued random variables. Sum-product allows us to get all the expected sufficient statistics with as few probabilistic computations as possible. The sum-product algorithm itself works in the same way, the difference comes in the way the intermediate factors are represented and in the way the primitive probabilistic operations are implemented. We introduce these operations in Section 4.1.2, and then present our contributions in Sections 4.1.4 and 4.1.5, which is a series of simpler algorithms for implementing these primitive probabilistic operations, and an analysis of these algorithms.

4.1.1 Transducers and factor graphs

Weighted automata and transducers are simply automata and transducer where each arc in the transition diagram (a complete graph over a set of states Q) is associated with an emission-specific non-negative weight. Here we use the mealy model (i.e. where emissions are defined on edges rather than states) and assume without loss of generality that there is a single start and stop state. In automata, *emissions* are elements of $\hat{\Sigma} = \Sigma \cup \{\epsilon\}$, and in transducers, elements of $\hat{\Sigma}^2$ (pairs of symbols). The weight function is therefore a map $w : \hat{\Sigma} \times Q^2 \to [0, \infty)$ for automata, and $w : \hat{\Sigma}^2 \times Q^2 \to [0, \infty)$ for transducers. A *valid path* is a sequence of states $q_i \in Q$ and emissions $\hat{\alpha}_i \in \hat{\Sigma}$ starting at start and ending at stop: $p = q_0, \hat{\alpha}_1, q_1, \hat{\alpha}_2, q_2, \ldots, \hat{\alpha}_n, q_n$. The weight of a path, $w(p)$ is the product of the arc weights. The *normalization* of a transducer or automata is the sum of all the valid paths' weights.

In an automaton, a path emits a string $s \in \Sigma^*$ if s is equal to the concatenation of the non-epsilon emissions produced along the path $\hat{s} = \hat{\alpha}_1 \circ \cdots \circ \hat{\alpha}_n \in \hat{\Sigma}^*$, in which case we write $s \equiv \hat{s}$. Let also the weight assigned to a string $s \in \Sigma^*$, $w(s)$, be defined as the sum of the weights of all the paths emitting s.

Similarly, in transducers, a path emits a pair of strings s, s' if s is the concatenation of the each of the first components of the emissions (removing the epsilons), and s' of each of the second components. The weight assigned to a pair of strings $w(s, s')$ is then the sum of the weights of all the paths emitting s, s'.

To exemplify the concepts of weighted transducers and automata, we show the state diagrams for the building blocks of the factor graph behind PEF inference. To simplify the notation, we will assume in this section that there are no markedness features: $\theta_{\omega,t,p,\ell} = \theta_{\omega,t,p',\ell}$ for all p, p'. Relaxing this assumption simply involves increasing the size of the state space of the weighted transducers and automata to encode the previously generated symbol. This is conceptually and computationally easy, but the automata are easier to visualize in the restricted setting.

There are three types of factors to go over (see Figure 4.1: there is an automaton at the root modeling the initial string distribution; there are transducers capturing string mutation probabilities between pairs of species connected by an edge in the original phylogenetic tree; and finally, there is one automaton attached to each leaf, which is an indicator functions on the observed strings.

The root factor simply generates strings of geometric distributed length with specified n-

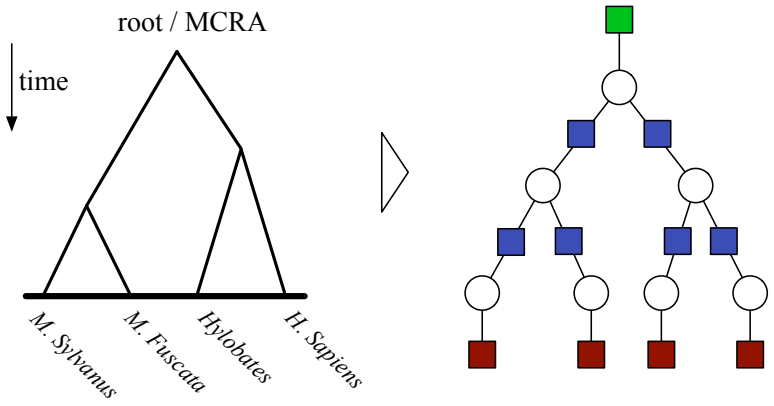


Figure 4.1. Factor graph construction (right) derived from a phylogenetic tree (left): in green, the automaton at the root that models the initial string distribution; in blue, the transducers capturing string mutation probabilities; and in red, the automata at the leaves, which are indicator functions on the observed strings.

gram expectations. We show a simple example in Figure 4.2 where the length distribution has mean $1/(1-q-p)$, and the proportion of 'a' symbols versus 'b' symbols is p/q. Note that to obtain a probabilistic automaton from a weighted automaton, one simply divides the weights by the normalization of the automaton.

We now turn to the problem of specifying the weighted transducers encoding the binary factors. The weights of these binary transducers should precisely encode the probabilities assigned to pairs of strings by the mutation Markov chain of Section 3.2.4. In order to do this properly, it is important to avoid overcounting transducer paths. Historically, weighted transducers have been motivated by the problem of finding maximum weight path, in which case overcounting is not a problem and can be safely ignored. However in our probabilistic framework, where summing over paths is needed to normalize the posterior distribution, avoiding overcounting is important.

We start by showing in Figure 4.4 a transducer that would be an appropriate implementation of the mutation Markov chain in the case of maximization (i.e. the path of maximum weight coincides with the MLE of the mutation Markov chain), but that is not correct for computing summations. The problem is that any path containing, for example, one consecutive insertion and deletion will be counted at least twice: one where the insertion is performed first, and one where the deletion is performed first. This violates the de-

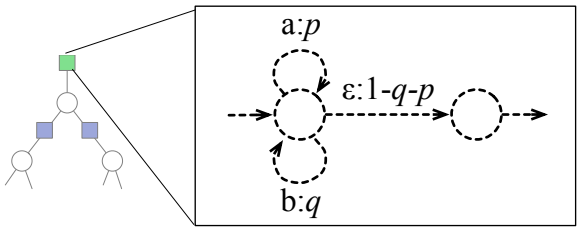


Figure 4.2. An initial (prior) string distribution automaton. We are using the standard graphical automata representation, where states are dashed circles (to differentiate them from nodes in graphical models), and arcs are labelled by emissions and weights. Zero weight arcs are omitted. The start and stop state are indicated by inward and outward arrows. We show here a simple example where the length distribution has mean $1/(1-q-p)$, and the proportion of 'a' symbols versus 'b' symbols is p/q.

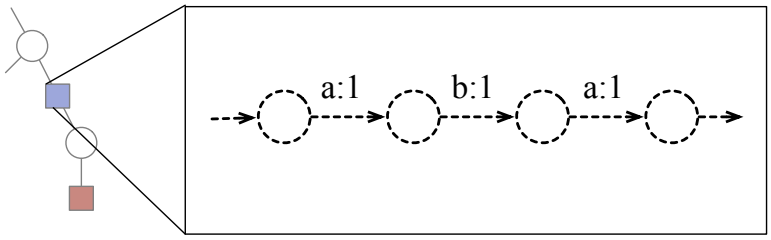


Figure 4.3. An automaton encoding a string indicator function. In this example, only the string 'aba' is accepted (i.e. the string 'aba' is the only string emitted with positive weight).

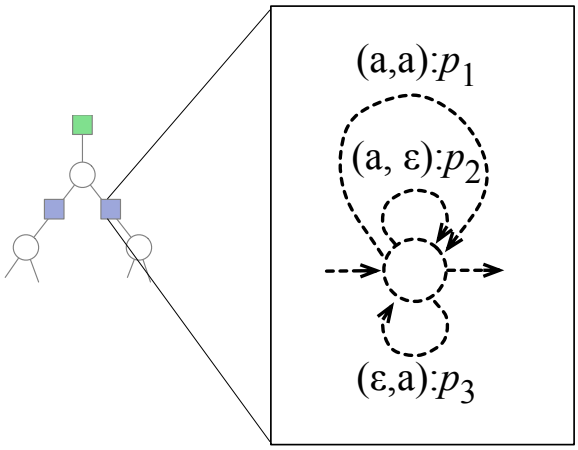


Figure 4.4. An example of a mutation transducer that gives positive weights to the same pairs of strings as the mutation Markov chain of Section 3.2.4. We show the example for the case where there is a single character in the alphabet. Note that this transducer would be an appropriate implementation in the case of maximization (i.e. the path of maximum weight coincides with the MLE of Section 3.2.4), but it is not correct for computing summations.

scription of the mutation Markov chain, where insertions need to be performed first by construction.

Fortunately, a simple modification of the transducer achieves this, shown in Figure 4.5.

Now that we have described the factor graph, we turn to the high-level description of the exact PEF inference algorithm: the sum-product algorithm generalized to string-valued factor graphs.

4.1.2 Sum-product on string-valued factor graphs

As it is often done in standard treatments of finite-state space graphical model, we start by describing the elimination algorithm. This section covers elementary material, but is necessary to setup the precise terminology needed for the following sections.

Going from the elimination algorithm to sum-product is easy in trees: it only involves executing the elimination algorithm twice while keeping track of the eliminated factors.

We therefore focus without loss of generality on the elimination algorithm on string-valued

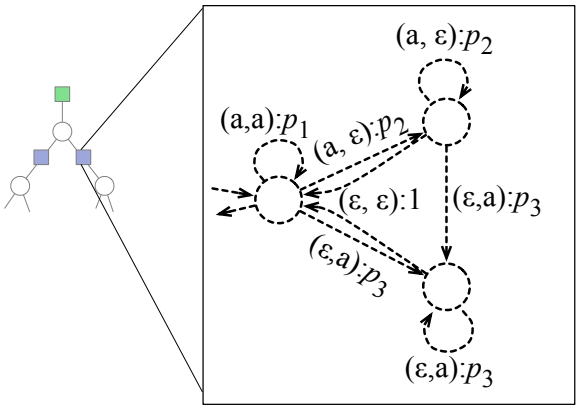


Figure 4.5. A correct mutation transducer. Note that there is a bijection between derivations in the mutation Markov chain and accepting paths in this transducer.

factor graphs.[1]. The main goal is to identify what fundamental probabilistic operations are needed.

Given a set of query nodes, the elimination algorithm proceeds iteratively as follows (see Figure 4.6 for an example): At each step, there is a factor graph containing all the query nodes plus a set S of other nodes to marginalize. As long as S is non-empty, the algorithm picks one leaf factor or node and eliminate it. We call the operation corresponding to a factor elimination *pointwise product*, and to variable elimination, *marginalization* (shown in Figure 4.7). We define two kinds of pointwise products: those where the variable connected to the eliminated factor is also connected to a binary factor (first kind, shown in Figure 4.8), and the others (second kind, shown in Figure 4.9).

The invariant of the elimination algorithm is the joint marginal distribution of the query nodes. This constrains the properties of the operations. If we let $w^{(i)}$ denote the weight function of automaton or transducer i, as shown in Figures 4.7, 4.8, and 4.9, then the constraints can be easily shown to be:

$$\text{marginalization:} \quad w^{(\mathrm{m})}(s) = \sum_{s' \in \Sigma^*} w^{(1)}(s, s') \quad \forall s \in \Sigma^*$$

$$\text{pointwise product (first kind):} \quad w^{(\mathrm{p})}(s) = w^{(1)}(s) \cdot w^{(2)}(s) \quad \forall s \in \Sigma^*$$

$$\text{pointwise product (second kind):} \quad w^{(\mathrm{p})}(s) = w^{(1)}(s, s') \cdot w^{(2)}(s) \quad \forall s, s' \in \Sigma^*$$

[1]More precisely on the elimination algorithm in the case where the query nodes are not separated by a non-query node in the original graphical model.

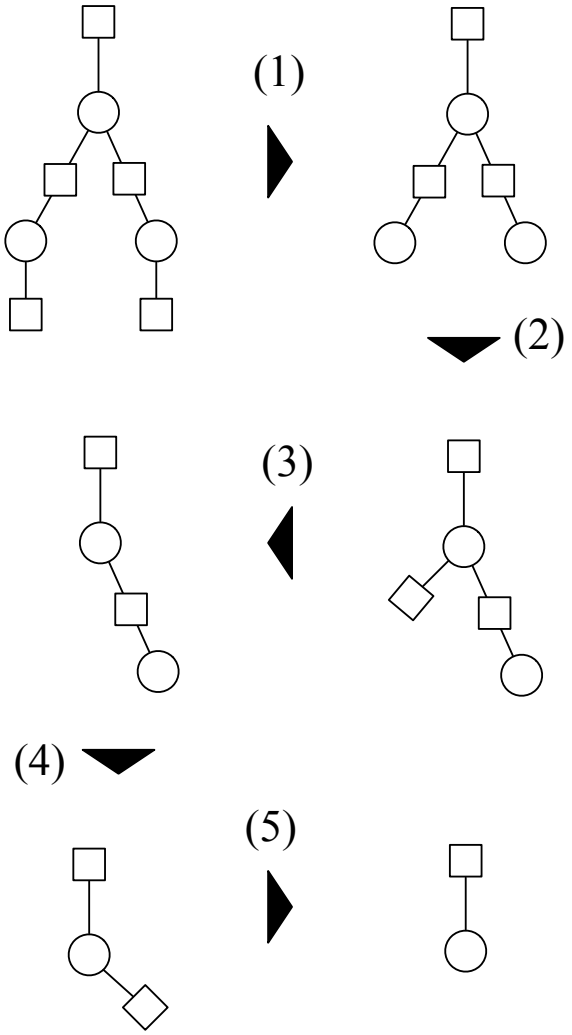


Figure 4.6. (1) and (3) are pointwise operation of the first kind; (2) and (4) are marginalization operations; and (5) is a pointwise product of the second kind.

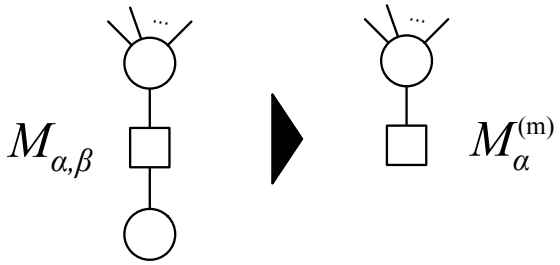


Figure 4.7. Marginalization operation

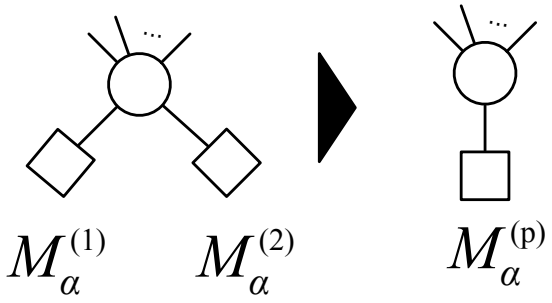


Figure 4.8. Pointwise product of the first kind

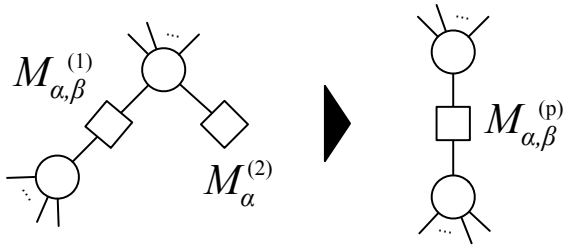


Figure 4.9. Pointwise product of the second kind

We will show in Section 4.1.4 how we can implement the operations to satisfy these properties, but before doing this, we first describe a convenient way of representing transducers in the next section.

4.1.3 Matrix representation of transducers and automata

The representation we use in this section originates from [81], where it was introduced in the context of formal power series, and therefore, of automata. Here we will need the extension to transducer, which we describe after reviewing the basic representation theory definitions.

We can assume without loss of generality that its states are labelled by the integers $1, \ldots, K$, and that state 1 is the unique start state, and K, the unique stop state. As in [81] we view automata as a character-indexed collections of $K \times K$ matrices,

$$\left(M_{\hat{\alpha}} \in \mathbb{M}_K \left([0, 1] \right) \right)_{\hat{\alpha} \in \hat{\Sigma}}$$

There is one matrix in this collection for each type of emission, i.e. for each character in the alphabet, and entry $M_\alpha(k, k')$ encodes the weight of transitioning from states k to k' while emitting α.

While epsilons are convenient when defining new automata, some of the algorithms in the next section assume that there are no epsilon transitions of positive weights. Fortunately, converting an automata defined with the matrices as above into a new automaton with epsilon-free transition matrices M'_α is easy:[2]

$$M'_\alpha = \begin{cases} 0 & \text{if } \alpha = \epsilon \\ M_\epsilon^* M_\alpha & \text{o.w.} \end{cases}$$

whenever $M^* = (1 - M)^{-1}$ exists. The proof is in Appendix A.1, where we also show that there is a simple closed form for the normalizer of any automata as well:

$$Z = \phi \left(\left(\sum_{\alpha \in \hat{\Sigma}} M_{\hat{\alpha}} \right)^* \right),$$

where $\phi(M) = M(1, K)$.

[2] We assume without loss of generality that the last emission of positive weight path is never an epsilon—this can be done by adding a boundary symbol at the end of each string.

Transducers can be viewed as a collection of matrices as well, but this time the index runs over pairs of symbols $(\hat{\alpha}, \hat{\beta}) \in \hat{\Sigma}$:

$$\left(M_{\hat{\alpha},\hat{\beta}} \in \mathbb{M}_K\left([0,1]\right) \right)_{\hat{\alpha},\hat{\beta}\in\hat{\Sigma}}$$

Note that in the context of transducers, epsilons need to be kept in the basic representation, otherwise transducers could not give positive weights to pairs of strings where the input has a different length than the output.

4.1.4 Implementation of the probabilistic operations

We now describe how the operations of marginalization and pointwise product can be succinctly expressed by the matrix formulation. Some of the results in this section are known (again, in the language of formal series). In particular a version of our result on pointwise products of the first kind appears in [51], but this previous work lacks the important epsilon-free condition, as well as a treatment of transducers needed to perform pointwise products of the second kind.

We start by the easier operation, marginalization. Referring to the notation of Figure 4.7, we show in Appendix A.2 that:

$$M^{(\mathrm{m})}_{\hat{\alpha}} = \sum_{\hat{\beta}\in\hat{\Sigma}} M_{\hat{\alpha},\hat{\beta}}.$$

The pointwise multiplication operations are simple as well, but there is a subtlety involving the epsilon transitions. Consider the following candidate for the first type of pointwise product (shown in Figure 4.8):

$$M^{(\mathrm{p})}_{\hat{\alpha}} = \{M^{(1)}_{\hat{\alpha}} | M^{(2)}_{\hat{\alpha}}\}, \tag{4.1}$$

where $\{A|B|C|\dots\}$ denotes the tensor product $A \otimes B \otimes C \otimes \cdots$. Note that we avoided the standard tensor product notation $\otimes$ because of a notation conflict with the notation of the automaton and transducer literature, in which $\otimes$ denotes multiplication in an abstract semi-ring (the generalization of normal multiplication, $\cdot$ used in this thesis). The operator $\otimes$ is also often overloaded in the literature to mean the product or concatenation of automata or transducers, which is not the same as the *pointwise* product as defined here.

Note that Equation (4.1) is incorrect if one of the transducers has non-zero epsilon transitions. To see why, note that for a given string, say ab $\in \Sigma$, computing ϕ applied to the the right hand side of Equation (4.1) yields terms such as

$$w^{(1)}(q_1, \mathrm{a}, q_2, \epsilon, q_3, \mathrm{b}, q_4) \cdot w'^{(2)}(q'_1, \mathrm{a}, q'_2, \epsilon, q'_3, \mathrm{b}, q'_4),$$

with $q_i \in Q, q'_i \in Q'$ but no terms such as

$$w^{(1)}(q_1, \mathrm{a}, q_2, \mathrm{b}, q_3) \cdot w'^{(2)}(q'_1, \mathrm{a}, q'_2, \epsilon, q'_3, \mathrm{b}, q'_4).$$

This is a problem since both types of terms are found when applying ϕ to the left hand side of Equation (4.1). We show in Appendix A.2 that this problem disappears when both automata are epsilon-free.

We now turn to pointwise products of the second kind, where an automaton is pointwise multiplied with a transducer. The difference is that all epsilons cannot be removed from a transducer: without emissions of the form $(\alpha, \epsilon), (\epsilon, \alpha)$, transducers would not have the capacity to model insertions and deletions. Fortunately, there is a work around: as long as the automaton $M^{(2)}$ is epsilon free, we show in Appendix A.2 that pointwise multiplications of the second kind can be implemented as follows:

$$M^{(\mathrm{p})}_{\hat{\alpha},\hat{\beta}} = \begin{cases} \left\{ M^{(1)}_{\hat{\alpha},\hat{\beta}} | M^{(2)}_{\hat{\alpha}} \right\} & \text{if } \hat{\alpha} \neq \epsilon \\ \left\{ M^{(1)}_{\hat{\alpha},\hat{\beta}} | I \right\} & \text{o.w} \end{cases}$$

Now that we have a simple expression for all of the operations required by the elimination (or sum-product) algorithm on string-valued graphical models, we turn to the problem of analyzing the time complexity of exact inference in string-valued graphical models.

4.1.5 Efficiency analysis

In this section, we analyze the computational complexity of the formulae obtained in the previous section. The main result is that for a fixed number of taxa, the exact E step of the algorithm of Section 4 is polynomial in the length of the sequences involved, however the cost also grows exponentially in the number of taxa. We also get a new proof for the running time of exact inference in the TKF91 model, as well as an improved running time for general string-valued, tree-shaped graphical models.

Let us look at the matrix implementation of the three operations defined in the previous sections. Clearly, the most expensive operations are to compute epsilon closures (which

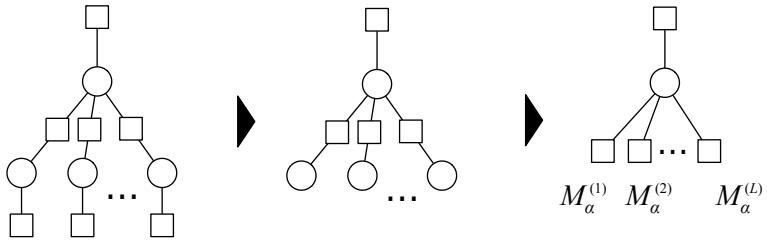


Figure 4.10. The first two steps of the elimination algorithm on the star tree.

involve taking the inverse of a matrix) and tensor products. We will denote the complexity degree of matrix multiplication by d. For tensor products, if A is a $n \times n$ matrix and B is $m \times m$, then $\{A|B\}$ is a $(nm) \times (nm)$ matrix, and so it can take up to m^2n^2 operation to form the tensor product of dense matrices. Fortunately, sparsity patterns can be exploited in both cases as we will see shortly. More precisely: we will use the following two results: First, if A has k nonzero entries, and B, l nonzero entries, then forming the tensor product takes time kl. Second, if A is also assumed to be triangular, then solving $A\mathbf{x} = \mathbf{b}$ takes time $O(k)$. We will assume throughout this section that matrices are stored in a sparse representation.

Note that in the elimination algorithm, the automata/transducer's matrices produced by a pointwise multiplication are fed into other pointwise products, augmenting the size of the automata/transducer matrix representations. Fortunately, part of this growth can be efficiently managed using sparsity and properties of tensor products. We therefore seek an expression for the total running time, i.e. taking into account the growth of the size of the intermediate factor matrix representations. To simplify the notation, we cover here the case of a star-shaped tree—removing this assumption only affect the running time up to a constant that does not depend on the sequence lengths or the number of taxa. To make the problem formulation concrete and concise, assume that we seek the normalization Z of the star-shaped, string-valued factor graph with L leaves. In the context of exact inference, the normalization problem is representative of the other tasks such as taking expectations or extracting samples.

Initially, the all the factors are of constant size, except for the factors at the leaves, which are of linear size (by that, we means that the matrices are $O(N \times N)$, where N is the geometric length of the observed sequences). In Figure 4.10, we show the result of the first two steps of the elimination algorithm. The operations involved in this first step are pointwise products of the second kind, (and therefore involve tensor products) however

the size of the initial pairwise factors is constant (see Figure 4.5), so the intermediate factors produced, $M^{(1)}, \ldots, M^{(L)}$ are of linear size. The second step is a marginalization, which does not increase the size of the matrices. Omitting the factor at the root (which is also of constant size, and therefore will not change the asymptotic running time), we get:

$$Z = \phi \left(\sum_{\hat{\alpha} \in \hat{\Sigma}} \left\{ \left(M_{\epsilon}^{(1)}\right)^{*} M_{\hat{\alpha}}^{(1)} \middle| \ldots \middle| \left(M_{\epsilon}^{(L)}\right)^{*} M_{\hat{\alpha}}^{(L)} \right\} \right)^{*} \tag{4.2}$$

Computing the right-hand-side naively would involve inverting a N^L by N^L matrix, and worse, even if the matrices $M_{\hat{\alpha}}^{(l)}$ were sparse, $\left(M_{\epsilon}^{(l)}\right)^{*}$ would not be guaranteed to be sparse as well, so the full cost of inversion would have to be paid. This would lead to a slow running time of N^{Ld}. In the rest of this section, we show how the running time can be decreased to N^L using properties of tensor products and weak assumptions on the factors. Moreover, we will show that these assumptions are always satisfied in PEFs and in many other graphical models of interest.

We will need the following basic properties:

Lemma 1. *Let $A^{(l)}$ be $m \times k$ matrices, and $B^{(l)}$ be $k \times n$ matrices. We have:*

$$\left\{ A^{(1)} B^{(1)} \middle| \ldots \middle| A^{(L)} B^{(L)} \right\} = \left\{ A^{(1)} \middle| \ldots \middle| A^{(L)} \right\} \left\{ B^{(1)} \middle| \ldots \middle| B^{(L)} \right\}$$

Lemma 2. *If $A^{(l)}$ are invertible, then:*

$$\left\{ \left(A^{(1)}\right)^{-1} \middle| \ldots \middle| \left(A^{(L)}\right)^{-1} \right\} = \left\{ A^{(1)} \middle| \ldots \middle| A^{(L)} \right\}^{-1}$$

Lemma 3. *If A is a $m \times k$ invertible matrix, and B is a $k \times n$ matrix such that $A - B$ is invertible, then*

$$\left(A^{-1} B\right)^{*} = A (A - B)^{-1}$$

Applying these to Equation (4.2), and letting $\bar{M} = I - M$, we obtain:

$$\begin{aligned}
Z &= \phi\left(\sum_{\hat{\alpha}\in\hat{\Sigma}}\left\{\left(M_{\epsilon}^{(1)}\right)^{*} M_{\hat{\alpha}}^{(1)}\middle|\ldots\middle|\left(M_{\epsilon}^{(L)}\right)^{*} M_{\hat{\alpha}}^{(L)}\right\}\right)^{*} \\
&= \phi\left(\left\{\left(M_{\epsilon}^{(1)}\right)^{*}\middle|\ldots\middle|\left(M_{\epsilon}^{(L)}\right)^{*}\right\}\sum_{\hat{\alpha}\in\Sigma}\left\{M_{\hat{\alpha}}^{(1)}\middle|\ldots\middle|M_{\hat{\alpha}}^{(L)}\right\}\right)^{*} \\
&= \phi\left(\left\{\bar{M}_{\epsilon}^{(1)}\middle|\ldots\middle|\bar{M}_{\epsilon}^{(L)}\right\}\left(\left\{\bar{M}_{\epsilon}^{(1)}\middle|\ldots\middle|\bar{M}_{\epsilon}^{(L)}\right\}-\sum_{\hat{\alpha}\in\Sigma}\left\{M_{\hat{\alpha}}^{(1)}\middle|\ldots\middle|M_{\hat{\alpha}}^{(L)}\right\}\right)^{-1}\right) \\
&= \phi\left(A(A-B)^{-1}\right),
\end{aligned}$$

where $A = \left\{\bar{M}_{\epsilon}^{(1)}\middle|\ldots\middle|\bar{M}_{\epsilon}^{(L)}\right\}$, and $B = \sum_{\hat{\alpha}\in\Sigma}\left\{M_{\hat{\alpha}}^{(1)}\middle|\ldots\middle|M_{\hat{\alpha}}^{(L)}\right\}$.

Naively, this still has running time N^{Ld}, but the advantage of this equation is that it allows us to exploit sparsity patterns. If we can show that the relevant automata and transducer's matrices are upper triangular sparse matrices (i.e. N by N matrices with $O(N)$ non zero entries), then it is easy to show that A and B would then be upper triangular as well, with only $O(N^L)$ non-zero components. Furthermore, if we let $C = (A-B)^{-1}$, we can see that only its last column $\mathbf{x}$ is actually needed to compute Z. At the same time, we can write $(A-B)\mathbf{x} = [0, 0, \ldots, 0, 1]$, which can be solved using the back substitution algorithm, getting an overall running time of $O(N^L)$.

We will call a transducer or automaton *triangular* if its states can be ordered such that all its matrices are upper triangular. The question now is whether the matrices $M_{\hat{\alpha}}^{(l)}$ are triangular. Observe first that among the original factors, the indicator factors at the leaves are triangular, but not the other ones (both the root automaton and the mutation transducer have cycles of positive weights, implying that they do not have a triangular matrix representation). Fortunately, the triangular property is not only preserve by the factor graph operations, they are also *contagious*:

Lemma 4. *If a transducer is triangular, then marginalizing it will create a triangular automaton. Moreover, the outcome of a pointwise product (either of the first or second kind) is guaranteed to be triangular whenever at least one of its input automaton or transducer is triangular.*

While this lemma is trivial to prove, it has important consequences on the cost of exact inference: the running time of $O(N^L)$ holds not only for PEF, but also more generally

whenever there are triangular automata at the leaves of the graphical model. We get as a corollary a more general proof that exact inference in star-shaped TKF91 phylogenetic tree can be done in time $O(N^L)$.

4.1.6 Summary

In this section, we have introduced a new framework for approaching exact inference in PEFs, and more generally, in string-valued graphical models. While conceptually simple and general, the framework does not yield polynomial-time inference as a function of the number of taxa studied. This should not come as a surprise: the PEF parameters can be set so that the problem of inference is equivalent to computing the partition function of a non-planar Ising model.[3] Consequently, the topic of the next section is approximate inference.

We conclude by making the observation that exact inference algorithms are needed as subroutines in most approximate inference algorithms (in particular, in those that follow). We therefore expect that the techniques developed in this section have the potential to make an impact, by simplifying the analysis and implementation of inference algorithms for string-valued graphical models.

4.2 Approximate inference

In this section, we introduce a new approximate inference algorithm that exploits the specific structure of PEF. After giving an high-level overview of the algorithm (reviewing related work at the same time, and comparing our algorithm to this previous art), we give a formal description and prove that it is asymptotically consistent.

[3]This is done as follows: assume that there are only two symbols in the alphabet, set the insertion and deletion transducer parameters to zero, and use the faithfulness and markedness features to simulate the spin agreement potentials. Each point has five neighbors (previous and next characters in the current word, parent character, and two descendent characters). The graph can be shown to be non-planar by embedding copies of $K_{3,3}$.

4.2.1 Overview

In PEFs, the string-valued graphical model introduced in the previous section can be misleading. It only encodes one type of independence relation, those between generations. There is another important structure that is not exploited by the exact inference algorithms. Informally, indel events that operate at the beginning of the sequences should not affect, for instance, those at the end. However, because alignments between the sequences are unknown in practice, it is difficult to exploit this structure in a principled way.

In many previous works [100, 59, 48], the following heuristic approach is taken to perform inference on the hidden nodes (refer to Figure 4.11): First, a guide tree (d) and a *multiple sequence alignment* (a) (a transitive alignment between the characters in the sequences of the modern species) are computed using heuristics [6, 29]. Second, the problem is cast into several easy subproblems as follows. For each equivalence class in the multiple sequence alignment (called a *site*, corresponding to a column in Figure 4.11(b)), a new graphical model is created with the same tree structure as the original problem, but where there is exactly one character in each node rather than a string. For nodes with a character in the current equivalence class, the node in this new tree is observed, the rest of the nodes are considered as unobserved data (Figure 4.11(c)). Note that the question marks are not the gaps commonly seen in linearized representations of multiple alignments, but rather phantom characters. Finally, each site is assumed independent of the others, so the subproblems can be solved efficiently by running the forward-backward algorithm on each site.

This heuristic has several problems, the most important being that it does not allow explicit modeling of insertions and deletions (indel), which are frequent in real biological data and play an important role in evolution [88]. If indels are included in the probabilistic model, there is no longer a deterministic notion of site on which independence assumptions can be made. This complicates inference substantially, making inference intractable, as discussed in the Section 4.1.

Holmes et al. [33] developed an approximate Markov chain Monte Carlo (MCMC) inference procedure for the TKF91 model. Their algorithm proceeds by sampling the entire sequence corresponding to a single species conditioning on its parent and children (Figure 4.11(e)). Doing so is possible using techniques such as those described in Section **??**. Since all the taxa but one are held fixed, exact inference on these simpler conditional is cubic.

We will call this type of kernel a *Single Sequence Resampling* (SSR) move. Unfortunately, chains based exclusively on SSR still have performance problems.

There are two factors behind these problems. The first factor is a random walk behavior that arises in tall chains found in large or unbalanced trees [33, 3]: initially, the indel events resampled at the top of the tree are *independent* of all the observations. It takes time for the information from the observations to propagate up the tree. The second factor is the computational cost of each SSR move, which is $O(N^3)$ with the TKF91 model and binary trees. For long sequences, this becomes prohibitive, so it is common to use a "maximum deviation pruning strategy" (i.e., putting a bound on the relative positions of characters that mutate from one to the other) to speed things up [3].

4.2.2 Ancestry resampling

In this section, we present a novel MCMC kernel for phylogenetic indel models that we refer to as *Ancestry Resampling* (AR). AR addresses both of the efficiency and accuracy problems that arise for SSR. Another remarkable property with AR is that the running time does not grow when features with large contexts are used. This is especially important for the large-context features introduced and used in the experiments on biological data of Section 3.5.

The intuition behind the AR approach is to use an MCMC kernel that combines the advantages of the two approaches described above: like the forward-backward algorithm in the site-independent case, AR always directly conditions on some part of the observed data, but, like SSR, it is capable of resampling the indel history. This is illustrated in Figure 4.11(f).

We now define some auxiliary variables that will be useful in the next section. Between each pair of nodes $a, b \in V$ connected by an edge and with respective strings $\boldsymbol{x}, \boldsymbol{y}$, we define an *alignment* random variable: its values are bipartite matchings between the characters of the strings $\boldsymbol{x}$ and $\boldsymbol{y}$. Links in this alignment denote survival of a character (allowing zero or more substitutions). Note that this alignment is monotonic: if character i in $\boldsymbol{x}$ is linked to character j in $\boldsymbol{y}$, then the characters $i' > i$ in $\boldsymbol{x}$ can only be unlinked or linked to a character with index $j' > j$ in $\boldsymbol{y}$. The random variable that consists of the alignments and the strings for all the edges and nodes in the phylogenetic tree τ will be called a *derivation*.

Note also that a derivation D defines another graph that we will call a *derivation graph*.

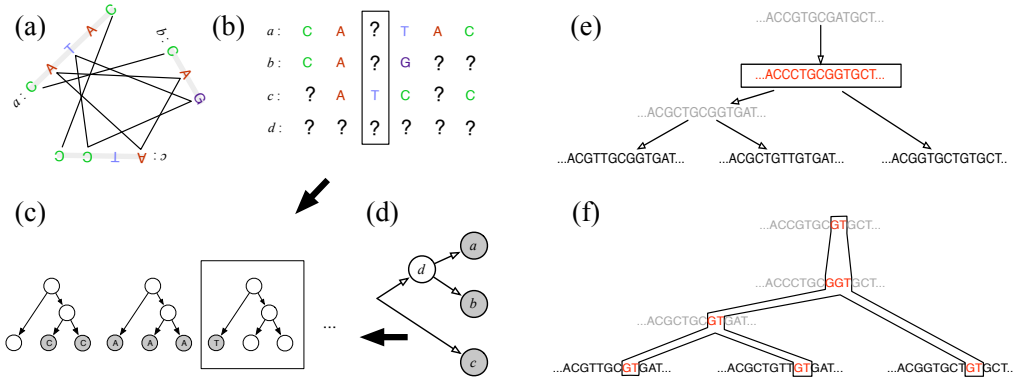


Figure 4.11. Comparison of different approaches for sampling MSAs: (a,b,c,d) heuristics based on site independence; (e) Single Sequence Resampling; (f) Ancestry Resampling. The boxes denote the structures that can be sampled or integrated out in one step by each method.

Its nodes are the characters of all the strings in the tree. We put an edge between two characters x, y in this graph iff two properties hold. Let $a, b \in V$ be the nodes corresponding to the strings from which respectively x, y belongs to. We put an edge between x, y iff (1) there is an edge between a and b in E and (2) there is a link between x, y in the alignment of the corresponding strings. Examples of derivation graphs are shown in Figure 4.12.

The approximate inference algorithm we propose, Ancestry Resampling (AR), is based on the Metropolis-Hastings (MH) framework. While the SSR kernel resamples the whole sequence corresponding to a single node, AR works around the difficulties of SSR by joint resampling of a "thin vertical slice" (Figure 4.11(f)) in the tree that is composed of a short substring in every node. As we will see, if we use the right definition of vertical slice, this yields a valid and efficient MH algorithm.

We will call one of these "thin slices" an *ancestry* $\mathcal{A}$, and we now discuss what its definition should be. Some care will be needed to ensure irreducibility and reversibility of the sampler.

We first augment the state of the AR sampler to include the derivation auxiliary variable described in Section 2.4. Let D be the current derivation and let $\boldsymbol{x}$ be a substring of one of the terminal nodes, say in node e. We will call $\boldsymbol{x}$ an *anchor*. The ancestry will depend on both a derivation and an anchor. The overall MH sampler is a mixture of proposal distributions indexed by a set of anchors covering all the characters in the terminal strings.

Each proposal resamples a new value of $\mathcal{A}(D, \boldsymbol{x})$ given the terminal nodes and keeping $\mathcal{A}(D, \boldsymbol{x})^c$ frozen.

We first let $\mathcal{A}_0(D, \boldsymbol{x})$ be the set of characters connected to some character in $\boldsymbol{x}$ in the derivation graph of D (see Figure 4.12(a)). This set $\mathcal{A}_0(D, \boldsymbol{x})$ is not a suitable definition of vertical slice, but will be useful to construct the correct one. It is unsuitable for two reasons. First, it does not yield an irreducible chain, as illustrated in same figure, where nine of the characters of this sample (those inside the dashed curve) will never be resampled, no matter which substrings of the terminal node is selected as anchor. Secondly, we would like the vertical slices to be contiguous substrings rather than general subsequences to ease implementation.

We therefore modify the definition recursively as follows. See Figure 4.12(b) for an illustration of this definition. For $i > 0$, we will say that a character token y is in $\mathcal{A}_i(D, \boldsymbol{x})$ if one of the following conditions is true:

1. y is connected to $\mathcal{A}_{i-1}(D, \boldsymbol{x})$,
2. y appears in a string $\cdots y' \cdots y \cdots y'' \cdots$ such that both y' and y'' are in $\mathcal{A}_{i-1}(D, \boldsymbol{x})$,
3. y appears in a string $\cdots y' \cdots y \cdots$ such that y' is in $\mathcal{A}_{i-1}(D, \boldsymbol{x})$ and $\boldsymbol{x}$ is a suffix,
4. y appears in a string $\cdots y \cdots y' \cdots$ such that y' is in $\mathcal{A}_{i-1}(D, \boldsymbol{x})$ and $\boldsymbol{x}$ is a prefix.

Then, we define $\mathcal{A}_\infty(D, \boldsymbol{x}) := \cup_{i=0}^{\infty} \mathcal{A}_i(D, \boldsymbol{x})$. In words, a symbol is in $\mathcal{A}_\infty(D, \boldsymbol{x})$ if it is linked to an anchored character through the alignments, or if it is "squeezed" between previously connected characters. Cases 3 and 4 handle the boundaries of strings. With this property, irreducibility could be established with some conditions on the anchors, but it turns out that this definition is still not quite right.

With $\mathcal{A}_\infty$, the main problem arises when one tries to establish reversibility of the chain. This is illustrated in Figure 4.12(d). In this example, the chain first transitions to a new state by altering the circled link. One can see that with the definition of $\mathcal{A}_\infty(D, \boldsymbol{x})$ given above, from the state 4.12 (e), the state in 4.12 (d) is now unreachable by the same resampling operator, the reason being that the substring labeled $\boldsymbol{z}$ in the figure belongs to the frozen part of the state if the transition is visited backwards.

While there exist MCMC methods that are not based on reversible chains [13], we prefer to take a simpler approach: a variation on our definition solves the issue, informally by taking vertical slices $\mathcal{A}(D, \boldsymbol{x})$ to be roughly the "complement of the ancestry taken on the

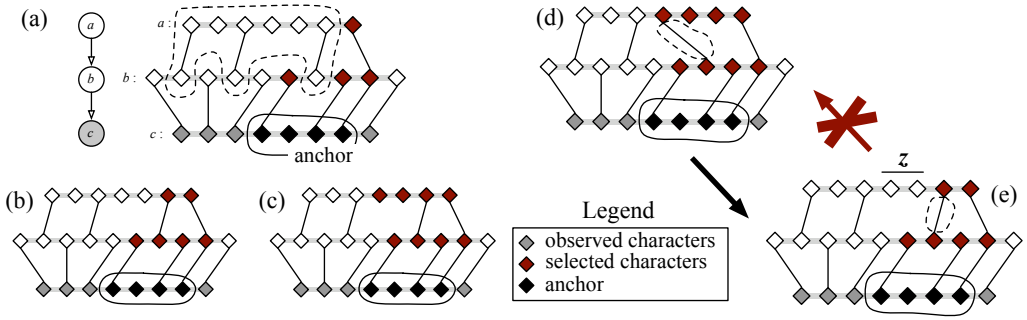


Figure 4.12. (a): the simple guide tree used in this example (left) and the corresponding sequences and alignments (right). (a,b,c): the definitions of $\mathcal{A}_0, \mathcal{A}_\infty, \mathcal{A}$ respectively are shaded (the "selected characters"). (d,e): An example showing the non-reversibility problem with $\mathcal{A}_\infty$.

complement of the anchor". More precisely, if $\boldsymbol{x} = \boldsymbol{x'xx''}$ is the string at the anchor node e, we let the resampled section to be $\mathcal{A}(D, \boldsymbol{x}) := (\mathcal{A}_\infty(D, \boldsymbol{x'}) \cup \mathcal{A}_\infty(D, \boldsymbol{x''}))^c$ This creates slightly thicker slices (Figure 4.12(c)) but solves the reversibility problem. We will call $\mathcal{A}(D, \boldsymbol{x})$ the *ancestry* of the anchor $\boldsymbol{x}$. With this definition, the proposal distribution can be made reversible using a MH acceptance ratio; it is also irreducible.

The problem of resampling a single slice decomposes along the tree structure τ, but an unbounded number of indels could occur a priori inside the thin slice. It may seem at the first glance that we are back at our initial problem: sampling from a tree-structured directed graphical model where the support of the space of the nodes is a countably infinite space. But in fact, we have made progress: the distribution is now concentrated on very short sequences. Indeed, the anchors $\boldsymbol{x}$ can be taken relatively small (we used anchors of length 3 to 5 in our experiments).

Another important property to notice is that given an assignment of the random variable $\mathcal{A}(D, \boldsymbol{x})$, it is possible to compute efficiently and exactly an unnormalized probability for this assignment. The summation over the possible alignments can be done using a standard quadratic dynamic program known in its max version as the Needleman-Wunsch algorithm [66].

4.2.3 Cylindric proposal

We now introduce the second idea that will make efficient inference possible: when resampling an ancestry given its complement, rather than allowing all possible strings for the resampled value of $\mathcal{A}(D, \boldsymbol{x})$, we restrict the choices to the set of substitutes that are close to its current value. We formalize closeness as follows: Let $\mathfrak{a}_1, \mathfrak{a}_2$ be two values for the ancestry $\mathcal{A}(D, \boldsymbol{x})$. We define the *cylindric distance* as the maximum over all the nodes e of the Levenshtein edit distance between the substrings in $\mathfrak{a}_1$ and $\mathfrak{a}_2$ at node e. Fix some positive integer m. The proposal distribution consider the substitution ancestry that are within a ball of radius m centered at the current state in the cylindric metric. The value $m = 1$ worked well in practice.

Here the number of states in the tree-structured dynamic program at each node is polynomial in the lengths of the strings in the current ancestry. A sample can therefore be obtained easily using the observation we have made that unnormalized probability can be computed.[4] Next, we compute the acceptance ratio, i.e.:

$$\min\left\{1, \frac{\mathbb{P}(\mathfrak{a}_p) \times Q(\mathfrak{a}_c|\mathfrak{a}_p)}{\mathbb{P}(\mathfrak{a}_c) \times Q(\mathfrak{a}_p|\mathfrak{a}_c)}\right\},$$

where $\mathfrak{a}_c, \mathfrak{a}_p$ are the current and proposed ancestry values and $Q(\mathfrak{a}_2|\mathfrak{a}_1)$ is the transition probability of the MH kernel, proportional to $\mathbb{P}(\cdot)$, but with support restricted on the cylindric ball centered at $\mathfrak{a}_1$.

4.2.4 Efficiency results

We performed experiments to measure the efficiency of AR. Since the focus is not on absolute accuracy in this section, all the algorithms do inference on the TKF91 model.

The hypothesis discussed in Section 4.2.1, i.e. that AR is faster than SSR since it avoid a random walk along the phylogenetic tree to propagate indels, predicts that the computational gap between SSR and AR should grow as the size of the phylogenetic tree augmented.

To test this, we have generated data from TKF91 along increasingly large trees. Giving

[4]What we are using here is actually a *nested dynamic programs*, meaning that the computation of a probability in the outer dynamic program (DP) requires the computation of an inner, simpler DP. While this may seem prohibitive, this is made feasible by designing the sampling kernels so that the inner DP is executed most of the time on small problem instances. We also cached the small-DP cost matrices.

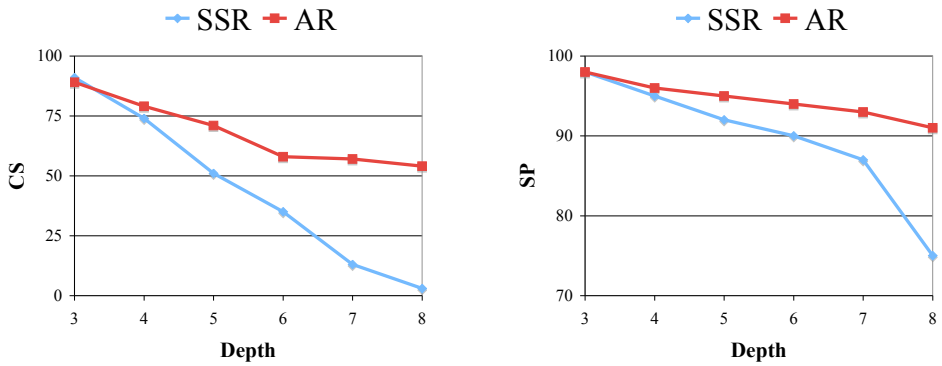


Figure 4.13. Sum-of-Pairs score (SP) and Column Score (CS) as a function of the depth of the generating trees.

both algorithms the true parameters and a fixed time budget, we then measured the SP score for each.

This prediction is confirmed as illustrated in Figure 4.13. For short trees, the two algorithms perform equally, SSR beating AR slightly for trees with three nodes, which is not surprising since SSR actually performs exact inference in this tiny topology. However, as trees get taller, the task becomes more difficult, and only AR maintains good performance.

Note that in these experiments, we have not applied a marginalization technique used Handel, which sums over the nucleotide identity of the internal sequences. Since this technique cannot be applied in the PEF models (because of markedness interactions), we have not to implement this feature in these experiments.

Chapter 5

Multiple Alignment Random Fields

In this chapter, we focus on the task of multiple sequence alignment (MSA), which allows us to use a simpler model family, or more precisely, a family of models with less hidden variables. We call these models multiple alignment random fields (MARF).

The consequence of removing hidden variables (the internal nodes in a phylogenetic tree), is the introduction of a loopy interactions between sets of observed sequences, contrasting with the acyclic interactions found in the model of Chapter 3. We describe in this chapter a new algorithm to perform inference in this loopy graph. In contrast to the methods of Chapter 4 (and also to much of the phylogenetic inference literature), we use a variational methods to formulate this algorithm.

The technique we use to create this variational algorithm applies much more broadly than MSA, covering a wide class of combinatorial inference problems. We present it from this general point of view in Section 5.4. We also apply it to MSA in Section 5.4.7, obtaining state-of-the-art MSA performances.

5.1 Model

The construction of MARF is based on the observation that a MSA is fully specified by the collection of all pairwise alignments. Given L sequences, there are $\binom{L}{2}$ pairwise alignments, and we introduce one random variable for each of these pairwise alignments.

Note that the converse does not hold: given an arbitrary collection of pairwise alignments, there is not necessarily a global MSA that has these pairwise alignments as marginals.

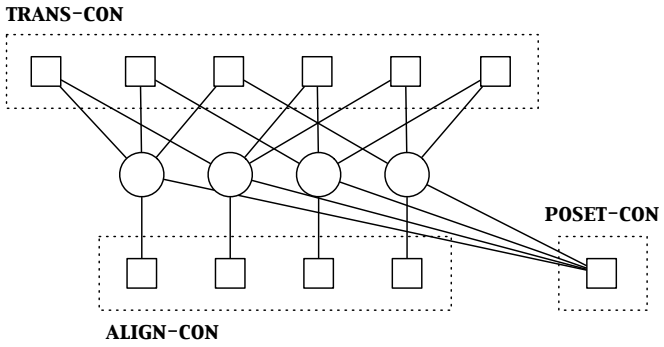


Figure 5.1. MARF graphical model.

For example, no MSA can have the following marginal pairwise alignments: x linked with y in a first pairwise alignment, y linked with z in a second one, but z not linked with x in a third one.

More generally, let us look at the properties that characterize all *consistent pairwise alignments* (a collection of pairwise alignments is consistent if there is a MSA with these pairwise alignment marginals).

Consider first only two sequences of length M and N respectively. A pairwise sequence alignment is a bipartite graph on the characters of the two sequences (where each bipartite component has the characters of one of the sequences) constrained to be *monotonic*: if a character at index $m \in \{1, \ldots, M\}$ is aligned to a character at index $n \in \{1, \ldots, N\}$ and another character at index $m' > m$ is aligned to index n', then we must have $n' > n$. A multiple alignment between K sequences of lengths $N_1, N_2, \ldots, N_K$ is a K-partite graph, where the k-th components' vertices are the characters of the k-th sequence, and such that the following three properties hold (see Figure 5.2):

ALIGN-CON: Each pair of components forms a pairwise alignment as described above.

TRANS-CON: The alignments are *transitive*, i.e., if character c_1 is aligned to c_2 and c_2 is aligned to c_3 then c_1 must be aligned to c_3.

POSET-CON: The alignments satisfy a *partial order* property: there exists a partial order p on the connected components of the graph with the property that if $C_1 <_p C_2$ are two distinct connected components and $c_1 \in C_1, c_2 \in C_2$ are in the same sequence, then the index of c_1 in the sequence is smaller than the index of c_2.

Let us look at a factor graph constructed from these constraints (see Figure 5.1). The

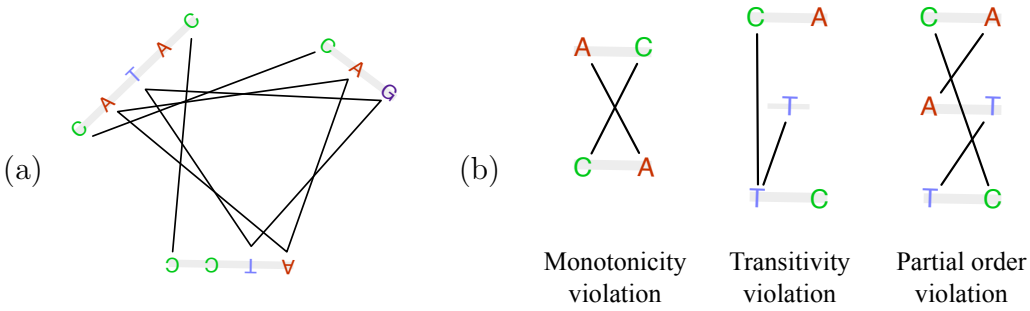


Figure 5.2. (a) An example of a valid multiple alignment between three sequences. (b) Examples of *invalid* multiple sequence alignments illustrating what is left out by the three constraints of Section 5.1.

variables are the pairwise alignments. The constraints in Align-Con can be viewed as unary factors; those in Trans-Con, as binary factors; and those in (3), as factors of degree three and higher, including factors over all variables. While Align-Con and Trans-Con induce a polynomial number of factors (in the number of taxa), Poset-Con induces an exponential number of factors.

In MARF, we only parameterize interactions of type Align-Con. This is done in the same fashion as in the mutation Markov chain of Section 3.2.4, i.e. with locally normalized transducers with local transition probabilities taken from an exponential family. MARF and PEF are therefore similar at a small scale (i.e. in the form of the individual factors), but quite different at a larger scale (i.e. in the way the factors are interconnected).

5.2 Computational aspects

Some aspects of MARF makes inference easier than PEF inference, but other aspects introduce new difficulties. On one hand, the random variables now have a finite domain, and the graphical model admits a simple and efficient learning procedure based on a penalized pseudolikelihood (described below). On the other hand, there is now an exponential number of factors in the graphical model. In this section, we see how this difficulty can be addressed using a new variational framework.

5.2.1 Parameter estimation

We use a penalized pseudolikelihood to learn the mutation Markov chain parameters. This is equivalent to performing training on a dataset of pairwise alignments.

We optimize the objective function using EM, just as in Section 4, but this time the E step is tractable (alignment of two sequences of lengths N and M can be computed in time quadratic $O(NM)$ using for example the transducer algorithms of Section 4.1). This enables fast training.

In our experiments we used the following features (defined in Section 3.5.1): OPERATION, MARKEDNESS, FAITHFULNESS, AFFINE-GAPS, and HYDROPHOBIC-MODELING (the last in the list was only used for analyzing protein sequences).

5.2.2 Inference

After computing parameters, we need to compute MSAs for each group of sequences. Recall that just computing the pairwise alignments, as we did for training, is not sufficient, for two reasons.

1. First, a collection of pairwise alignments computed separately will not be consistent in general, potentially creating malformed outputs.

2. Second, the higher order interactions should be put to contribution to disambiguate hard alignment problems. For example, if it is not clear whether nucleotides x and y in two divergent sequences should be aligned, there might be a nucleotide z in an intermediate sequence, such that the alignments $x - z$ and $y - z$ are more certain. This information should be used to disambiguate the $x - y$ alignment.

To address (1), we use an approximation of the Bayes estimator, described in [80], which we review in Section 5.2.2.1. However this technique does not address (2), which is why we need the variational algorithm described in Section 5.2.2.2. At a high-level, the complete system works as follows: the variational algorithm first computes high-quality pairwise potentials using constraints ALIGN-CON, TRANS-CON, but not POSET-CON. Then, these marginals are fed into the Bayes estimator approximation (where they become coefficients of an optimization problem). The intuition behind this procedure is that most of the disambiguations can be handled by the interactions of type TRANS-CON, but in some

rare cases, POSET-CON is still needed to avoid malformed output MSAs, which would complicated validation and visualization of the produced MSAs.

5.2.2.1 Bayes estimator approximation

In this section, we review the technique of [80] to turn (potentially inconsistent) pairwise posteriors into a point estimate, i.e. a (consistent) global alignment.

This techniques is based on the idea of approximating the Bayes estimator over a linear MSA loss function. Here we describe the Bayes estimator in the simple case where the loss function is the Sum of Pairs (SP) score, but other examples of losses are discussed in [80].[1]

Recall that an estimator is a Bayes estimator if it belongs to the set:

$$\operatorname*{argmin}_{m \in \mathrm{MSA}} \mathbb{E}\left[1 - \mathrm{SP}(M, m) | Y\right],$$

where the random variable M is the model's multiple sequence alignment (a deterministic function of the pairwise alignments), Y is the observed sequences, and $\mathrm{SP}(\cdot, \cdot)$ is the SP score (turned into a loss by subtracting it from one).

Next, note that the SP score is a linear function of the individual alignment links present in the reference MSA m_{g}:

$$\mathrm{SP}(m_{\mathrm{ref}}, m_{\mathrm{g}}) = \frac{1}{|C(m_{\mathrm{ref}})|} \sum_{e \in C(m_{\mathrm{ref}})} \mathbf{1}[e \in m_{\mathrm{ref}}] \mathbf{1}[e \in m_{\mathrm{ref}}],$$

where $C(m)$ is the set of edges in the core blocks (and all the possible edges if m is not a reference). This means that the Bayes estimator can be written as:

$$\operatorname*{argmax}_{m \in \mathrm{MSA}} \sum_{e \in m} \mathbb{P}\left(e \in M | Y\right).$$

This objective function is intractable (because of the constraint set MSA), but a greedy algorithm works well in practice. This algorithm maintains a priority queue of edges, ordered by posterior, and builds a MSA incrementally, starting with the trivial, empty MSA over the observations. At each step, the edge with highest priority is popped, and

[1] Recall that the SP score is a recall metric on the pairwise alignment edges, measured on the core blocks (the part of the sequence that has been annotated).

the algorithm attempts to add it to the current MSA. Testing whether the edge can be added without violating the MSA constraints ALIGN-CON, TRANS-CON and POSET-CON is non-trivial, but efficient algorithms do exist [71].

5.2.2.2 Computing the posterior

We now turn to the problem of computing the alignment edge posteriors $\mathbb{P}(e \in M|Y)$. This is done using a new variational framework described in full generality in Section 5.4. In this section, we describe how it applies to the problem of MSA.

Our variational framework is applicable in the following setup: let $\mathcal{C} \subset \boldsymbol{\mathcal{X}}$ denote a *combinatorial space*, by which we mean a finite but large set, where testing membership is tractable, but enumeration is not, and suppose that the goal is to compute $\sum_{x\in\mathcal{C}} f(x)$, where f is a positive function. In our case, the combinatorial space $\mathcal{C}$ is the set of pairwise alignments over the sequences Y and satisfying constraints ALIGN-CON, TRANS-CON and POSET-CON; the function f is the joint probability distribution (which we try to normalize over the $\mathcal{C}$); and $\boldsymbol{\mathcal{X}}$ is the power set over the set E of edges, $\boldsymbol{\mathcal{X}} = 2^E$.

We approach this problem by exploiting a finite collection of sets $\{\mathcal{C}_i\} \subset \boldsymbol{\mathcal{X}}$ such that $\mathcal{C} = \cap_i \mathcal{C}_i$. Each $\mathcal{C}_i$ is larger than $\mathcal{C}$, but paradoxically it is often possible to find such a decomposition where for each i, $\sum_{x\in\mathcal{C}_i} f(x)$ is tractable.

Equivalently, this decomposition can be seen from point of view of exponential families, where the set intersection corresponds to a product of base measures,

$$\begin{aligned}\nu(x) &= \mathbf{1}[x \in \mathcal{C}] \\ &= \prod_i \nu_i(x) \\ &= \prod_i \mathbf{1}[x \in \mathcal{C}_i]\end{aligned}$$

For MSAs, the factorization we use has two types of factors. There are $\binom{K}{2}$ pairwise *alignment base measures*, and $T = \sum_{k,k',k'':k\neq k'\neq k''\neq k} N_k N_{k'} N_{k''}$ *transitivity base measures*, where N_k is the length of observed sequence k.

Alignment base measures enforce constraint ALIGN-CON between a pair of sequences. Transitivity base measures enforce constraint TRANS-CON between a triplet of characters. Both types can be computed in polynomial time, the first, by using the transducer

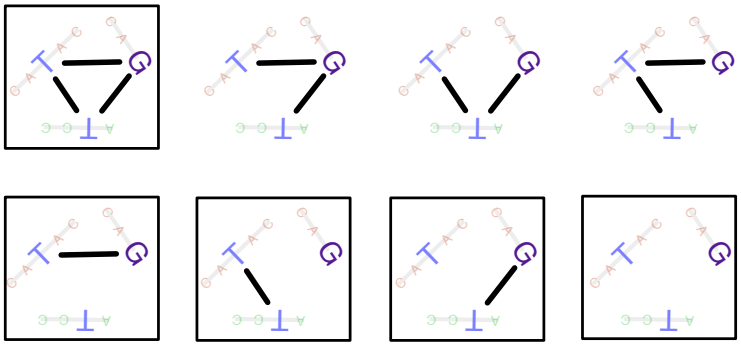


Figure 5.3. Pictorial representation of the terms involved in the transitivity super-partition computation. The boxed alignment triplets correspond to the transitive cases.

machinery of Section 4.1, the second, by enumeration (see Figure 5.3, and Appendix B.4 for details).

Once this factorization is provided, our framework automatically makes a collection of efficient and accurate variational algorithms available for performing inference. In our experiments, we used the loopy belief propagation (BP) algorithm (see Section 5.4 for details).

Note that we have excluded the intractable base measures of type Poset-Con in the factorization. We characterize in Section 5.4.5 the effect of this omission.

5.3 Experiments

We first applied our model and inference algorithms to BAliBASE [86], a standard protein multiple sequence alignment benchmark. We compared our system to Clustal 2.0.12 [29], the most popular multiple alignment tool, and ProbCons 1.12, a state-of-the-art system [15] that also relies on enforcing transitivity constraints, but which is not easily interpretable as optimizing an objective function. The main advantage of our system over the other systems is the better optimization technique, based on the measure factorization. The posterior over the alignments is used to approximate the minimum Bayes risk objective function over the "sum of pairs" (SP) metric, as described in Section 5.2.2.1.

We used the following experimental protocol: first, we trained parameters for HMMs using EM ran on all pairs of sequences in the test1/ref1 directory, without using the gold alignment information. Second, we ran BPMF with an annealing exponent of 1/10 on the

	Sum of Pairs score (SP)				
BAliBASE protein group	BPMF-1	BPMF-2	BPMF-3	Clustal [29]	ProbCons [15]
short, $< 25\%$ identity	0.68	0.74	**0.76**	0.71	0.72
short, 20% — 40% identity	0.94	**0.95**	**0.95**	0.89	0.92
short, $> 35\%$ identity	0.97	**0.98**	**0.98**	0.97	**0.98**
All	0.88	**0.91**	**0.91**	0.88	0.89

Table 5.1. Average SP scores in the ref1/test1 directory of BAliBASE. BPMF-i denotes the average SP of the BPMF algorithm after i iterations of (parallel) message passing.

consistency messages to avoid convergence problems. Third, we decoded (i.e. transformed the marginals into a single multiple sequence alignment) using the minimum Bayes risk approximation of [80]. Finally, we computed the standard SP (Sum of Pairs) metric on the annotated core blocks.

We show the results in Table 5.1. Our system outperformed both baselines after three BPMF parallel message passing iterations.

The algorithm converged in all protein groups, and performance was identical after more than three iterations. Although the overall performance gain is not statistically significant according to a Wilcoxon signed-rank test, the larger gains were obtained in the small identity subset, the "twilight zone" where research on multiple sequence alignment has focused.

We also tested our system on the comparative RNA dataset [39]. Since the annotation are much more dense in this dataset, it allows us to meaningfully compare the precisions of the alignments as well. We sampled 100 groups each containing nine 5S sequences from all three phylogenetic domains. More iterations were required for the messages to converge (12) in this larger dataset. We outperformed Clustal according to all metrics (note that Probcons results are not included since it supports only protein sequences). The results are shown Table 5.2.

To validate the importance of long indel modeling, we ran a separate experiment on the RNA without the **AFFINE-GAPS** feature. We obtained a relative precision reduction of 32% and a recall reduction of 45%, confirming the critical role played by this feature in biological experiments.

One caveat of this multiple alignment approach is its running time, which is cubic in the

	Scores			
Metric	BPMF-1	BPMF-6	BPMF-12	Clustal [29]
Edge recall (SP)	0.80	0.81	**0.82**	0.79
Edge precision	0.80	**0.81**	**0.81**	0.78
Edge F1	0.80	0.81	**0.82**	0.79

Table 5.2. MARF results on the comparative RNA dataset

length of the longest sequence, while most multiple sequence alignment approaches are quadratic. For example, the running time for one iteration of BPMF in this experiment was 364.67s, but only 0.98s for Clustal—this is why we have restricted the experiments to the short sequences section of BAliBASE. Fortunately, several techniques are available to decrease the computational complexity of this algorithm: the transitivity factors can be subsampled using a coarse pass, or along a phylogenetic tree; and computation of the factors can be entirely parallelized. We leave these improvements for future work.

5.4 General variational framework for combinatorial spaces

In this section, we present the variational framework for combinatorial spaces in its full generality. The description and analysis of the algorithms is easier to follow in the abstract setup.

While previous authors have proposed mean field or loopy belief propagation algorithms to approximate the partition function of a few specific combinatorial models—for example [82, 7] for parsing, and [35, 96] for computing the permanent of a matrix—we are not aware of a general treatment of variational inference in combinatorial spaces.

There has been work on applying variational algorithms to the problem of *maximization* over combinatorial spaces [84, 85, 18, 9], but maximization over combinatorial spaces is rather different than summation. For example, in the bipartite matching example considered in both [18] and this chapter, there is a known polynomial algorithm for maximization, but not for summation. Our approach is also related to agreement-based learning

[50, 49], although agreement-based learning is defined within the context of unsupervised learning using EM, while our framework is agnostic with respect to parameter estimation.

To make the exposition simpler, we use the running example of approximating the value and gradient of the log-partition function of a Bipartite Matching model (BM) over $K_{N,N}$, a well-known #P problem [91]. Unless we mention otherwise, we will consider bipartite perfect matchings; non-bipartite and non-perfect matchings are discussed in Section 5.4.6.1.

We start by setting some notation: Since we are dealing with discrete-valued random variables $\boldsymbol{X}$, we can assume without loss of generality that the probability distribution for which we want to compute the partition function and moments is a member of a regular exponential family with canonical parameters $\boldsymbol{\theta} \in \mathbb{R}^J$:

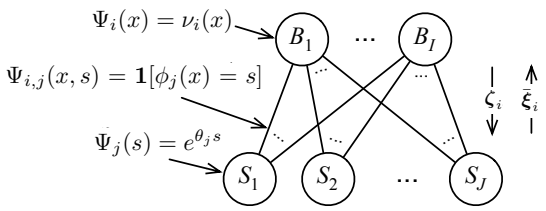
$$\mathbb{P}(\boldsymbol{X} \in B) = \sum_{x \in B} \exp\{\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle - A(\boldsymbol{\theta})\} \nu(x), \tag{5.1}$$

$$A(\boldsymbol{\theta}) = \log \sum_{x \in \boldsymbol{\mathcal{X}}} \exp\{\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle\} \nu(x), \tag{5.2}$$

for a J-dimensional sufficient statistic $\boldsymbol{\phi}$ and base measure ν over $\mathcal{F} = 2^{\boldsymbol{\mathcal{X}}}$, both of which are assumed (again, without loss of generality) to be indicator functions: $\phi_j, \nu : \boldsymbol{\mathcal{X}} \to \{0, 1\}$. Here $\boldsymbol{\mathcal{X}}$ is a superset of both $\mathcal{C}$ and all of the $\mathcal{C}_i$s. The link between this setup and the general problem of computing $\sum_{x \in \mathcal{C}} f(x)$ is the base measure ν, which is set to the indicator function over $\mathcal{C}$: $\nu(x) = \mathbf{1}[x \in \mathcal{C}]$, where $\mathbf{1}[\cdot]$ is equal to one if its argument holds true, and zero otherwise.

The goal is to approximate $A(\boldsymbol{\theta})$ and $\nabla A(\boldsymbol{\theta})$ (recall that the j-th coordinate of the gradient, $\nabla_j A$, is equal to the expectation of the sufficient statistic ϕ_j under the exponential family with base measure ν [95]). We want to exploit situations where the base measure can be written as a product of I measures $\nu(x) = \prod_{i=1}^{I} \nu_i(x)$ such that each factor $\nu_i : \boldsymbol{\mathcal{X}} \to \{0, 1\}$ induces a *super-partition function* assumed to be tractable: $A_i(\boldsymbol{\theta}) = \log \sum_{x \in \boldsymbol{\mathcal{X}}} \exp\{\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle\} \nu_i(x)$. This computation is typically done using dynamic programming (DP). We also assume that the gradient of the super-partition functions is tractable, which is typical for DP formulations.

In the case of BM, the space $\boldsymbol{\mathcal{X}}$ is a product of N^2 binary alignment variables, $x = x_{1,1}, x_{1,2}, \ldots, x_{N,N}$. In the Standard Bipartite Matching formulation (which we denote by SBM), the sufficient statistic takes the form $\phi_j(x) = x_{m,n}$. The measure factorization we



BPMF($\boldsymbol{\theta}, A_1, \ldots, A_I$)

1: $\zeta_{i,j}^{(1)} = 0$
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $\quad \bar{\boldsymbol{\xi}}_i^{(t)} = \boldsymbol{\theta} + \sum_{i':i' \neq i} \boldsymbol{\zeta}_{i'}^{(t-1)}$
4: $\quad \boldsymbol{\zeta}_i^{(t)} = \text{logit}\left(\nabla A_i\left(\bar{\boldsymbol{\xi}}_i^{(t)}\right)\right) - \bar{\boldsymbol{\xi}}_i^{(t)}$
5: **end for**
6: **return** $\hat{\boldsymbol{\mu}} = \text{logistic}\left(\boldsymbol{\theta} + \sum_i \boldsymbol{\zeta}_i^{(T)}\right)$

Figure 5.4. Left: the bipartite graphical model used for the MRF construction described in Section 5.4.1. Right: pseudocode for the BPMF algorithm.

use to enforce the matching property is $\nu = \nu_1 \nu_2$, where:

$$\nu_1(x) = \prod_{m=1}^{N} \mathbf{1}[\sum_{n=1}^{N} x_{m,n} \leq 1], \tag{5.3}$$

$$\nu_2(x) = \prod_{n=1}^{N} \mathbf{1}[\sum_{m=1}^{N} x_{m,n} \leq 1]. \tag{5.4}$$

We show in Appendix B.3 that A_1 and A_2 can be computed in time $O(N^2)$ for the SBM.

The last assumption we make is that given a vector $\boldsymbol{s} \in \mathbb{R}^J$, there is at most one possible configuration x with $\boldsymbol{\phi}(x) = \boldsymbol{s}$. We call this the *rich sufficient statistic condition* (RSS). Since we are concerned in this framework with computing expectations, not with parameter estimation, this can be done without loss of generality. For example, if the original exponential family is curved (e.g., by parameter tying), for the purpose of computing expectations one can always work in the over-complete parameterization, and then project back to the coarse sufficient statistic for parameter estimation.

5.4.1 Markov random field reformulation

We start by constructing an equivalent but more convenient exponential family. This general construction has an associated bipartite Markov Random Field (MRF) with structure $K_{I,J}$, shown in Figure 5.4. This new bipartite structure should not be confused with the bipartite graph from the $K_{N,N}$ bipartite graph specific to the BM example: the former is part of the general theory, the latter is specific to the bipartite matching example.

The bipartite MRF has I random variables in the first graph component, $B_1, \ldots, B_I$, each having a copy of $\boldsymbol{\mathcal{X}}$ as its domain. In the second component, the graph has J

random variables, $S_1, \ldots, S_J$, where S_j has a binary domain $\{0, 1\}$. The pairwise potential between an event $\{B_i = x\}$ in the first component and one $\{S_j = s\}$ in the second is given by $\Psi_{i,j}(x, s) = \mathbf{1}[\phi_j(x) = s]$. The following one-node potentials are also included: $\Psi_i(x) = \nu_i(x)$ and $\Psi_j(s) = e^{\theta_j s}$.

The equivalence between the two formulations follows from the rich sufficient statistic condition, which implies (for a full proof of the equivalence, see Appendix B.1):

$$\sum_{s_1 \in \{0,1\}} \sum_{s_2 \in \{0,1\}} \cdots \sum_{s_J \in \{0,1\}} \prod_{i=1}^{I} \prod_{j=1}^{J} \mathbf{1}[\phi_j(x_i) = s_j] = \begin{cases} 1 & \text{if } x_1 = x_2 = \cdots = x_I \\ 0 & \text{otherwise.} \end{cases} \tag{5.5}$$

This transformation into an equivalent MRF reveals several possible variational approximations. We show in the next section how loopy belief propagation [101] can be modified to tractably accommodate this transformed exponential family, even though some nodes in the graphical model—the B_is—have a domain of exponential size. We then describe similar updates for mean field [72] and tree-reweighted [94] variational algorithms. We will refer to these algorithms as BPMF (Belief Propagation on Measure Factorizations), MFMF (Mean Field on Measure Factorizations) and TRWMF (Tree-Reweighted updates on Measure Factorizations). In contrast to BPMF, MFMF is guaranteed to converge[2], and TRWBF is guaranteed to provide an upper bound on the partition function.[3]

5.4.2 Implicit message representation

The variables B_i have a domain of exponential size, hence if we applied belief propagation updates naively, the messages going from B_i to S_j would require summing over an exponential number of terms, and messages going from S_j to B_i would require an exponential amount of storage. To avoid summing explicitly over exponentially many terms, we adapt an idea from [82] and exploit the fact that an efficient algorithm is assumed for computing the super-partition function A_i and its derivatives. To avoid the exponential storage of messages going to B_i, we use an implicit representation of these messages in the canonical parameter space.

Let us denote the messages going from S_j to B_i by $M_{j \to i}(s), s \in \{0, 1\}$ and the reverse messages by $m_{i \to j}(x), x \in \boldsymbol{\mathcal{X}}$. From the definitions of $\Psi_{i,j}, \Psi_i, \Psi_j$, the explicit belief

[2]Although we did not have convergence issues with BPMF in our experiments.

[3]Surprisingly, MFMF does not provide a lower bound (see Appendix B.6).

propagation updates are:

$$m_{i\to j}(s) \propto \sum_{x\in\boldsymbol{\mathcal{X}}} \mathbf{1}[\phi_j(x)=s]\nu_i(x) \prod_{j':j'\neq j} M_{j'\to i}(x)$$
$$M_{j\to i}(x) \propto \sum_{s\in\{0,1\}} e^{\theta_j s}\mathbf{1}[\phi_j(x)=s] \prod_{i':i'\neq i} m_{i'\to j}(s). \tag{5.6}$$

The task is to get an update equation that does not represent $M_{j\to i}(x)$ explicitly, by exploiting the fact that the super-partition functions A_i and their derivatives can be computed efficiently. To do so, it is convenient to use the following equivalent representation for the messages $m_{i\to j}(s)$: $\zeta_{i,j} = \log m_{i\to j}(1) - \log m_{i\to j}(0) \in [-\infty, +\infty]$.[4]

If we also let $f_{i,j}(x)$ denote any function proportional to $\prod_{j':j'\neq j} M_{j'\to i}(x)$, we can write:

$$\zeta_{i,j} = \log\left(\frac{\sum_{x\in\boldsymbol{\mathcal{X}}}\phi_j(x)f_{i,j}(x)\nu_i(x)}{\sum_{x\in\boldsymbol{\mathcal{X}}}(1-\phi_j(x))f_{i,j}(x)\nu_i(x)}\right) = \operatorname{logit}\left(\frac{\sum_{x\in\boldsymbol{\mathcal{X}}}\phi_j(x)f_{i,j}(x)\nu_i(x)}{\sum_{x\in\boldsymbol{\mathcal{X}}}f_{i,j}(x)\nu_i(x)}\right), \tag{5.7}$$

where $\operatorname{logit}(x) = \log x - \log(1-x)$. This means that if we can find a parameter vector $\boldsymbol{\xi}_{i,j} \in \mathbb{R}^J$ such that

$$f_{i,j}(x) = \exp\langle\boldsymbol{\phi}(x), \boldsymbol{\xi}_{i,j}\rangle \propto \prod_{j':j'\neq j} M_{j'\to i}(x),$$

then we could write $\zeta_{i,j} = \operatorname{logit}\left(\nabla_j A_i(\boldsymbol{\xi}_{i,j})\right)$. We derive such a vector $\boldsymbol{\xi}_{i,j}$ as follows:

$$\begin{aligned}\prod_{j':j'\neq j} M_{j'\to i}(x) &= \prod_{j':j'\neq j}\sum_{s_{j'}\in\{0,1\}} e^{\theta_{j'}s_{j'}}\mathbf{1}[\phi_{j'}(x)=s_{j'}] \prod_{i':i'\neq i} m_{i'\to j'}(s_{j'}) \\ &= \prod_{j':j'\neq j} e^{\theta_{j'}\phi_{j'}(x)} \prod_{i':i'\neq i} m_{i'\to j'}(\phi_{j'}(x)) \\ &\propto \exp\left\{\sum_{j':j'\neq j}\phi_{j'}(x)\left(\theta_{j'} + \sum_{i':i'\neq i}\zeta_{i',j'}\right)\right\},\end{aligned}$$

where in the last step we have used the assumption that ϕ_j has domain $\{0,1\}$, which implies that $m_{i\to j}(\phi_j(x)) = \exp\{\phi_j(x)\log m_{i\to j}(1) + (1-\phi_j(x))\log m_{i\to j}(0)\} \propto \exp\{\phi_j(x)\zeta_{i,j}\}$. The required parameters are therefore: $\left(\boldsymbol{\xi}_{i,j}\right)_{j'} = \mathbf{1}[j \neq j']\left(\theta_{j'} + \sum_{i':i'\neq i}\zeta_{i',j'}\right)$.

[4] In what follows, we will assume that $\zeta_{i,j} \in (-\infty, +\infty)$. The extended real line is treated in Appendix B.7.

5.4.3 Reuse of partition function computations

Naively, the updates derived so far would require computing each super-partition function J times at each message passing iteration. We show that this can be reduced to computing each super-partition function only once per iteration, a considerable gain.

We first define the vectors:

$$\bar{\boldsymbol{\xi}}_i = \boldsymbol{\theta} + \sum_{i':i' \neq i} \boldsymbol{\zeta}_{i'},$$

and then rewrite the numerator inside the logit function in Equation (5.7) as follows:

$$\begin{aligned}\sum_{x \in \boldsymbol{\mathcal{X}}} \phi_j(x) f_{i,j}(x) \nu_i(x) &= \sum_{s \in \{0,1\}} \sum_{x:\phi_j(x)=s} \exp\{\langle \phi(x), \bar{\boldsymbol{\xi}}_i \rangle\} \cdot e^{-\bar{\xi}_{i,j} s} \cdot s \cdot \nu_i(x) \\ &= e^{A_i(\bar{\boldsymbol{\xi}}_i) - \bar{\xi}_{i,j}} \nabla_j A_i(\bar{\boldsymbol{\xi}}_i),\end{aligned}$$

and similarly for the denominator:

$$\begin{aligned}\sum_{x \in \boldsymbol{\mathcal{X}}} f_{i,j}(x) \nu_i(x) &= e^{A_i(\bar{\boldsymbol{\xi}}_i) - \bar{\xi}_{i,j}} \nabla_j A_i(\bar{\boldsymbol{\xi}}_i) + e^{A_i(\bar{\boldsymbol{\xi}}_i)} (1 - \nabla_j A_i(\bar{\boldsymbol{\xi}}_i)) \\ &= e^{A_i(\bar{\boldsymbol{\xi}}_i)} \left(1 + (e^{-\bar{\xi}_{i,j}} - 1) \nabla_j A_i(\bar{\boldsymbol{\xi}}_i)\right).\end{aligned}$$

After plugging in the reparameterization of the numerator and denominator back into the logit function in Equation (5.7) and doing some algebra, we obtain the more efficient update $\zeta_{i,j} = \text{logit}\left(\nabla A_i(\bar{\xi}_{i,j})\right) - \bar{\xi}_{i,j}$, where the logit function of a vector, $\text{logit}\, \boldsymbol{v}$, is defined as the vector of the logit function applied to each entry of the vector $\boldsymbol{v}$. See Figure 5.4 for a summary of the BPMF algorithm.

5.4.4 Other variational algorithms

The ideas used to derive the BPMF updates can be extended to other variational algorithms with minor modifications. We sketch here two examples: a naive mean field algorithm, and a TRW approximation. See Appendix B.2 for details.

In the case of naive mean field applied the graphical model described in Section 5.4.1, the updates take a form similar to Equations (5.6), except that the reverse incoming message is not omitted when computing an outgoing message. As a consequence, the updates are

not directional and can be associated to nodes in the graphical model rather than edges:

$$M_j(s) \propto \sum_{x \in \boldsymbol{\mathcal{X}}} \mathbf{1}[\phi_j(x) = s] \nu_i(x) \prod_j m_i(x)$$
$$m_i(x) \propto \sum_{s \in \{0,1\}} e^{\theta_j s} \mathbf{1}[\phi_j(x) = s] \prod_i M_j(s).$$

This yields the following implicit updates:[5]

$$\boldsymbol{\xi}^{(t)} = \boldsymbol{\theta} + \sum_i \boldsymbol{\zeta}_i^{(t-1)}$$
$$\boldsymbol{\zeta}_i^{(t)} = \text{logit}\left(\nabla A_i\left(\boldsymbol{\xi}^{(t)}\right)\right), \tag{5.8}$$

and the moment approximation $\hat{\boldsymbol{\mu}} = \text{logistic}(\boldsymbol{\xi})$.

In the case of TRW, lines 3 and 6 in the pseudocode of Figure 5.4 stay the same, while the update in line 4 becomes:

$$\left(\boldsymbol{\xi}_{i,j}\right)_{j'} = \left(\theta_{j'} - \rho_{i \to j'} \zeta_{i,j'} + \sum_{i' : i' \neq i} \rho_{i' \to j'} \zeta_{i',j'}\right) \cdot \begin{cases} \rho_{j' \to i} & \text{if } j' \neq j \\ (1 - \rho_{i \to j}) & \text{otherwise,} \end{cases} \tag{5.9}$$

where $\rho_{i \to j}$ are marginals of a spanning tree distribution over $K_{I,J}$. We show in Appendix B.2 how the idea in Section 5.4.3 can be exploited to reuse computations of super-partition functions in the case of TRW as well.

5.4.5 Large factorizations

In some cases, it might not be possible to write the base measure as a succinct product of factors. Fortunately, there is a simple and elegant workaround to this problem that retains good theoretical guarantees. The basic idea is that dropping measures with domain $\{0, 1\}$ in a factorization can only increase the value of the partition function. This solution is especially attractive in the context of outer approximations such as the TRW algorithm, because it preserves the upper bound property of the approximation.

5.4.6 Other examples of factorization

In this section, we show four other examples of measure factorizations, to demonstrate the applicability beyond MSA inference.

[5] Assuming that naive mean field is optimized coordinate-wise, with an ordering that optimizes all of the m_i's, then all of the M_j's.

5.4.6.1 More matchings

Our approach extends naturally to matchings with higher-order (augmented) sufficient statistic, and to non-bipartite/non-perfect matchings.

Let us first consider an Higher-order Bipartite Model (HBM), which has all the basic sufficient statistic coordinates found in SBM, plus those of the form $\phi_j(x) = x_{m,n} \cdot x_{m+1,n+1}$. We claim that with the factorization of Equation (5.3), the super-partition functions A_1 and A_2 are still tractable in HBM. To see why, note that computing A_1 can be done by building an *auxiliary exponential family* with associated graphical model given by a chain of length N, and where the state space of each node in this chain is $\{1, 2, \ldots, N\}$. The basic sufficient statistic coordinates $\phi_j(x) = x_{m,n}$ are encoded as node potentials, and the augmented ones as edge potentials in the chain. This yields a running time of $O(N^3)$ for computing one super-partition function and its gradient (see Appendix B.3 for details). The auxiliary exponential family technique used here is reminiscent of [2].

Extension to non-perfect and non-bipartite matchings can also be done easily. In the first case, a dummy "null" node is added to each bipartite component. In the second case, where the original space is the set of $\binom{N}{2}$ alignment indicators, we propose a decomposition into N measures. Each one checks that a single node is connected to at most one other node: $\nu_n(x) = \mathbf{1}[\sum_{n'=1}^{N} x_{n,n'} \leq 1]$.

5.4.6.2 Linearization of partial orders

A *linearization* of a partial order p over N objects is a total order t over the same objects such that $x \leq_p y \Rightarrow x \leq_t y$. Counting the number of linearizations is a well-known #P problem [5]. Equivalently, the problem can be view as a matching between a DAG $G = (V, E)$ and the integers $\{1, 2, \ldots, N\}$ with the order constraints specified on the edges of the DAG.

To factorize the base measure, consider a collection of I directed forests on V, $G_i = (V, E_i), i \in I$ such that their union covers G: $\cup_i E_i = E$. See Figure 5.4.6.2 for an example. For a single forest G_i, a straightforward generalization of the algorithm used to compute HBM's super-partition can be used. This generalization is simply to use sum-product with graphical model G_i instead of sum-product on a chain as in HBM (see Appendix B.5 for details). Again, the state space of the node of the graphical model is

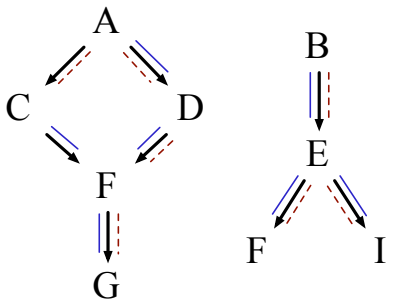


Figure 5.5. The DAG representation of a partial order. An example of linearization is A,C,D,B,E,F,G,H,I. The fine red dashed lines and blue lines demonstrate an example of two forests covering the set of edges, forming a measure decomposition with two factors. The linearization A,D,B,E,F,G,H,I,C is an example of a state allowed by one factor but not the other.

$\{1, 2, \ldots, N\}$, but this time the edge potentials enforce the ordering constraints of the current forest.

5.4.6.3 Partition of the plane

Counting plane partitions is a classical problem in statistical physics, combinatorics and probability theory [98]. A *plane partition* is an array of non-negative integers, $p_{n,m}, 0 \leq n, m \leq N$ such that $p_{n+1,m} \geq p_{n,m}$, $p_{n,m+1} \geq p_{n,m}$. There is a well-known connection between these arrays and a certain type of *routing*, exemplified in Figure 5.6. We will describe the factorization in the routing formulation, which represents plane partitions as a collection of N non-crossing integer paths, each of length $2N + 1$. Path n starts and ends at position n, and its transitions are either the identity, an increase by one (only allowed in the first N transitions), or a decrease by one (in the last N transitions).

We propose an approximation based on $N - 1$ factors. Each factor relaxes the problem to enforcing non-crossing only for two consecutive paths. With this relaxation, the partition function can be computed in $O(N^3)$ by using forward-backward on a chain of length $2N + 1$ with state space $O(N^2)$ that keeps track of the position of the two consecutive paths.

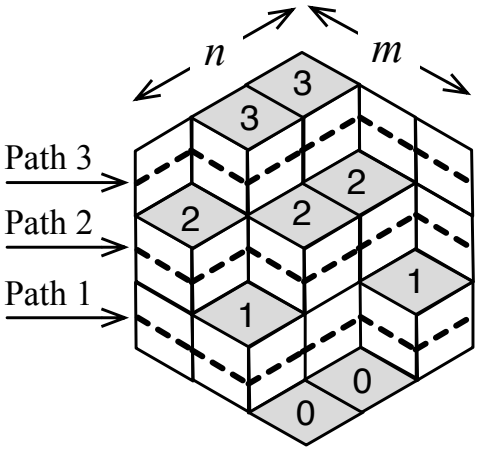


Figure 5.6. The 3D representation of a plane partition with $N = 3$. The value $p_{n,m}$ is the height of the pile at (n, m), for example $p_{3,3} = 3$. The equivalent *routing* representation is shown as a dashed line. For example, path 1 is $(1, 1, 1, 2, 2, 1, 1)$.

5.4.6.4 Traveling salesman problem

The method is not limited to #P problems derived from decision problems in P: we show in this section for example that the counting version of the traveling salesman problem, which is NP in its decision version [40], can be attacked with the same tools.

We consider a set of N cities $\{c_1, \ldots, c_N\}$, where each pair of cities has an associated parameter $\theta(c_n, c_m) \in \mathbb{R}$. A *tour* t is a list of cities, $t = t_1, t_2, \ldots, t_N : t_n \in \{c_1, \ldots, c_N\}$ where each city is visited exactly once, i.e. $\{t_1, \ldots, t_N\} = \{c_1, \ldots, c_N\}$. The *weight* of a tour is the product of the weights of the pairs of consecutive cities in the tour (modulo N): $w(t) = \exp\{\sum_{n=1}^{N-1} \theta(t_n, t_{n+1}) + \theta(t_1, t_N)\}$. By normalization, this yields a probability model:

$$\mathbb{P}(T = t) = \exp\left\{\sum_{n=1}^{N-1} \theta(t_n, t_{n+1}) + \theta(t_1, t_N) - A(\boldsymbol{\theta})\right\}$$
$$A(\boldsymbol{\theta}) = \log \sum_{\text{tour } t} w(t),$$

and also an exponential family indexed by $\boldsymbol{\theta}$.

Fix without loss of generality an arbitrary city c_1 as the starting and ending point, and take $\boldsymbol{\mathcal{X}}$ to be the set of all paths of length N that starts and ends at c_1, but without the

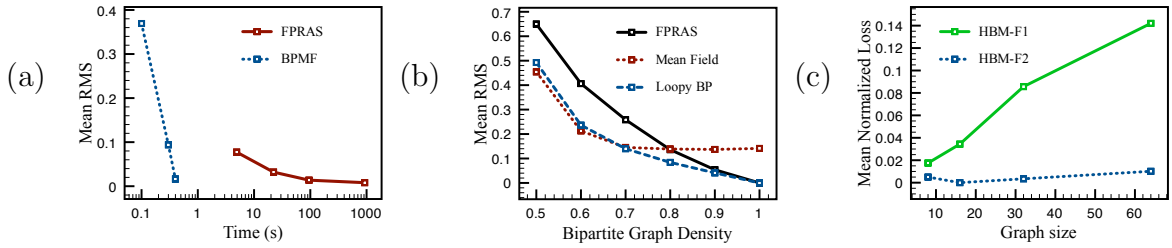


Figure 5.7. Experiments discussed in Section 5.4.7 on two of the matching models discussed. (a) and (b) on SBM, (c), on HBM.

coverage restriction. One factorization for this problem can be constructed by looping over the $N-1$ other cities, $c_n \neq c_1$, and building for each one a factor that enforces that c_n be visited exactly once. Computation over a single factor can be computed using dynamic programming (by maintaining the number of steps left and whether c_n was visited or not). Moreover, a state that satisfies all factors is a valid tour.

5.4.7 Matching experiments

As an additional set of experiments, we compared the approximation of SBM described in Section 5.4 to the Fully Polynomial Randomized Approximation Scheme (FPRAS) described in [73]. We performed all our experiments on 100 iid random bipartite graphs of size N, where each edge has iid appearance probability p, a random graph model that we denote by $\mathrm{RB}(N, p)$. In the first and second experiments, we used $\mathrm{RB}(10, 0.9)$. In this case, exact computation is still possible, and we compared the mean Root Mean Squared (RMS) of the estimated moments to the truth. In Figure 5.7(a), we plot this quantity as a function of the time spent to compute the 100 approximations. In the variational approximation, we measured performance at each iteration of BPMF, and in the sampling approach, we measured performance after powers of two sampling rounds. The conclusion is that the variational approximation attains similar levels of error in at least one order of magnitude less time in the $\mathrm{RB}(10, 0.9)$ regime.

Next, we show in Figure 5.7(b) the behavior of the algorithms as a function of p, where we also added the mean field algorithm to the comparison. In each data point in the graph, the FPRAS was run no less than one order of magnitude more time than the variational algorithms. Both variational strategies outperform the FPRAS in low-density regimes, where mean field also slightly outperforms BPMF. On the other hand, for high-density

regimes, only BPMF outperforms the FPRAS, and mean field has a bias compared to the other two methods.

The third experiment concerns the augmented matching model, HBM. Here we compare two types of factorization and investigate the scalability of the approaches to larger graphs. Factorization F1 is a simpler factorization of the form described in Section 5.4.6.1 for non-bipartite graphs. This ignores the higher-order sufficient statistic coordinates, creating an outer approximation. Factorization F2, described in Section 5.4.6.1 specifically for HBM, is tighter. The experimental setup is based on a generative model over noisy observations of bipartite perfect matchings described in Appendix B.8. We show in Figure 5.7(c) the results of a sequence of these experiments for different bipartite component sizes $N/2$. This experiments demonstrates the scalability of sophisticated factorizations, and their superiority over simpler ones.

Chapter 6

The Poisson Sequence Change Process

In this chapter, we attack the problem of joint phylogenetic tree and MSA inference using a new stochastic process. This joint inference task has been an active field in the last decade, and a large body of work suggests that the efforts spent in this field are justified. Several studies have shown for example that using a fixed MSA creates biases in the inferred phylogeny [99], and conversely, that a single guide tree may decreases MSA accuracy [67].

So far, the state-of-the-art systems for MSA and phylogeny inference have been based on system combination [53]: one uses a high performance, off the shelf algorithm for tree inference, then feeds the result to an off the shelf MSA inference algorithm. In some cases, this is iterated several times. The drawbacks of these systems include a lack of theoretical understanding, over-alignment problems [80], the difficulty of getting calibrated confidence intervals and challenging estimation problems.

As discussed in Section 2.8, defining string-valued continuous time Markov chains (CTMCs) is the main alternative to system combination. Starting from a CTMC supporting insertions and deletions (indels), we review in Section 6.1 how one automatically gets a maximum likelihood estimate over both MSAs and phylogenetic trees, an estimator with well-understood theoretical properties. By adding a prior over trees and a loss function, one also gets a Bayesian estimator giving uncertainty estimates over trees and alignments, as well as an easy way to model uncertainty over parameters.

The main obstacle to high performance CTMC-based joint inference is a specific probabilistic calculation: the marginal density of a tree and MSA pair, integrating over all CTMC sample along the tree leading to the specified MSA. If this quantity can be computed exactly and efficiently, estimating the maximum likelihood (and, the posterior distribution) can be done using standard local search techniques (respectively, standard Metropolis-Hastings moves). We emphasize that this quantity should be computed not only efficiently, but also exactly. This is especially important in the Bayesian setting, where the acceptance probability will involve taking a ratio of two marginal densities, and where the asymptotic consistency guarantees of MCMC only hold if the acceptance probability is calculated exactly. Even for simple indel models such as TKF91, the best published algorithm is the technique of [62], which has worst-case time complexity exponential in the number of taxa.[1]

In this chapter, we show that a slight modification to the TKF91 process makes it possible to compute exact marginal densities in time linear in both the sequence length and the number of taxa. This is possible thanks to an equivalent Poisson process representation where alignment columns become *exchangeable*, a very useful property for deriving algorithms. The algorithm we propose is very simple to implement, and the Poisson process representation provides a framework for extending inference to more complicated evolutionary models, for example long indel models.

There has been previous work on fully polynomial MCMC samplers for the TKF91 model [36, 33], but they do not permit resampling the tree topology jointly with the MSA. The approaches of [74] and [76] are closer to ours, but neither of these algorithms corresponds to computing the marginal likelihood of a joint stochastic process: in the former case, the probability of indel is independent of branch lengths, and in the latter case, it varies in a way inspired by but not explicitly corresponding to marginalization of a stochastic process. In neither case does the quality of the inferred MSAs tested empirically.

This chapter is organized as follows: after providing some background and preliminary definitions in Section 6.1, we describe in Section 6.2 the new CTMC, the *Poisson Sequence Change Process* (PSCP) and its Poisson process representation. In Section 6.3, we describe a (fully) linear time algorithm that computes the exact marginal density of a MSA and tree pair. In Section 6.4, we compare empirically the new process to other MSA and tree inference approaches. We conclude with a discussion in Section 6.5 of how the new process

[1]Note that the algorithm has been described as linear in [54], which is true in terms of the sequence length, but not in terms of the number of taxa.

and the inference techniques described in this chapter can be used as the foundation of more elaborate models.

6.1 Background

In this section, we define formally some of the concepts introduced informally in Section 2. To model a finite inventory of nucleotides, let Σ denote a finite set, which we identify with integers $\Sigma = \{1, 2, \ldots, K-1\}$.

We start this section by giving a constructive definition of the TKF91 process, reviewing how to generate a path of DNA sequences from the TKF91 model. Let us assume that at some point in time t, the sequence has length n. With probability one, the sequence will stay unchanged for an interval of time Δt, and then a single random substitution, insertion or deletion (mutation) will change it. To sample Δt and the nature of this mutation, we simulate $3n+1$ independent random variable with exponential distributions. The value of the smallest of these random variables (the "winner") determines Δt, and the index of the winner determines the nature (whether it is a substitution, deletion or insertion, the position of the operation, and the value of the introduced character, if applicable) of the next event as follows.

First, for each of the n nucleotides in the sequence, there is one exponential with rate μ_{TKF}; if one of these wins, that will cause the nucleotide to be deleted. Second, for each of the n current nucleotides, we simulate an exponential variable corresponding to a mutation event. These random variables have rates that depend on the current nucleotides (these rates are organized in a rate matrix θ). If one of these wins, an extra multinomial with parameters derived from θ is drawn to determine the new value of the nucleotide. Finally, for each of the $n+1$ positions before or after a nucleotide (and one "immortal" position if the sequence is empty), there is one exponential with rate λ_{TKF}; if one of these variables wins, it will cause an insertion. Again, an extra multinomial is drawn to determine the nature of the inserted nucleotide, with parameters generally taken to be from the stationary distribution π of the substitution rate matrix θ. That completes the description of TKF91.

This describes a sequence indel processes on a single edge, i.e. a single time interval with one ancestral species at one end of the interval, and one modern species at the end.

Extending a process defined on an interval to one defined on a rooted phylogenetic tree is conceptually easy since a phylogeny can be seen as a collection of lines attached together:

Definition 5 (Rooted phylogenetic tree)**.** *A* rooted phylogenetic tree τ *is specified by two objects: a finite directed tree* $(\mathscr{V}, \mathscr{E})$*, called the* topology*, and a* branch length *map* $b : \mathscr{E} \to (0, \infty)$*. Given a topology and a branch length map, the phylogenetic tree is the set* $\tau = \{(e, t) : e \in \mathscr{E}, t \in [0, b(e))\} \cup L$*, where we use* $L \subset \mathscr{V}$ *to denote the leaves, and we identify the elements of* $\mathscr{V}$ *with points in the tree, i.e. if* $e = (v \to w) \in \mathscr{E} \implies w = (e, b(e))$*. We will also use the notation* $b(v) = b(e)$ *when* $e = (w \to v)$*,* $\mathrm{pa}(v) = w$ *when* $(w \to v) \in \mathscr{V}$*, and* $\mathrm{child}(v) = \{v : (w \to v) \in \mathscr{E}\}$*.*

Given a CTMC and a phylogenetic tree, one can apply the CTMC process on all edges of the trees in preorder, starting with the sequence produced by the edge above to generate the sequences of any edge (a distribution over strings at the root is also needed; it is generally taken to be the stationary distribution of the substitution rate matrix θ).

We conclude this section by a two structures on phylogenetic trees that will be useful in the rest of this chapter. First, for any given phylogenetic tree τ, define a partial order on the points of τ: $(e, t) \preceq (e', t')$ if $e = e'$ and $t \leq t'$, or if there is a directed path in $(V, \mathscr{E})$ from e to e'. If $x \in \tau$, we will denote the subtree rooted at x by $\tau_x = \{y \in \tau : y \preceq x\}$ (see Figure 6.1). Second, endow the set τ with the obvious topology so that later on we can define Poisson processes on trees.

6.2 Model

The stochastic process we propose is very similar to the TKF91 process, with the exception of removing the dependence of the insertion rate on the sequence length. Therefore, instead of having $3n+1$ competing exponential random variables as in the TKF91 model (n for substitutions, $n+1$ for insertions, and n for deletions), we now have $2n+1$ variables (n for substitutions, 1 for insertion, with rate λ, n for deletion, each of rate μ. When an insertion occurs, its position is selected uniformly at random.[2]

This means that if the sequence has length $\frac{\lambda}{\lambda_{\mathrm{TKF}}} - 1$ at some point in time, the distribution

[2]More precisely, assume there is a real numbers in the interval $[0, 1]$ assigned to each nucleotide before the insertion. When an insertion occurs, sample a new real number uniformly in the interval $[0, 1]$ and insert the new nucleotide at the unique position (with probability one) such that an increasing sequence of real numbers is maintained.

over the time and type of the next mutation will the same as TKF91,[3] but in general will be different.

Does this modification makes the model significantly less appropriate than TKF91 for evolutionary modeling? In the typical cases where the lengths of the sequences have the same order of magnitude across all species in the window of time studied, we show in Section 6.4.1 that a model based on this new new process can be used effectively to do inference on TKF91-distributed data. We also discuss in Section 6.5 how to accommodate sequences of varying sizes. Finally, we show in Section 6.2.3 that the asymptotic sequence length as time goes to infinity is the same in the modified process as in TKF91.

On the other hand, this modification does have drastic simplifying effects on the complexity of computing expectations. The simplifications are made possible by an equivalent Poisson representation of the process described in the next section. Let us call the process described in this section the Constant Insertion Rate Process (CIRP). In the next section, we describe another process, the Poisson Sequence Change Process (PSCP), and then show that CIRP and PSCP actually have the same distribution.

6.2.1 Poisson process representation

Before describing the Poisson Process representation, let us start by introducing a reparameterization of the deletion and substitution rates μ, θ. To do this, we will need the following notation: let $g =$ '-' $\notin \Sigma$ denote an extra symbol, with index K, called the *gap symbol*. Also set $\Sigma_g = \Sigma \cup g$.

The reparameterization is a matrix Q with entries given by:

$$Q_{k,k'} = \begin{cases} -\sum_{k'' \neq k'} Q_{k,k''} & \text{if } k = k' \\ 0 & \text{if } k = K \\ \mu & \text{if } k' = K \\ \theta_{k,k'} & \text{o.w.} \end{cases}$$

Note that in contrast to the model of [76], deletion (the gap symbol with index K) is an absorbing state in this reparameterization. As we will see in Section 6.3, this avoids the complications of the inference algorithms used in [76].

[3] Using the fact that the minimum of exponential variables with λ_i is exponential, with rate equal to the sum of the λ_i.

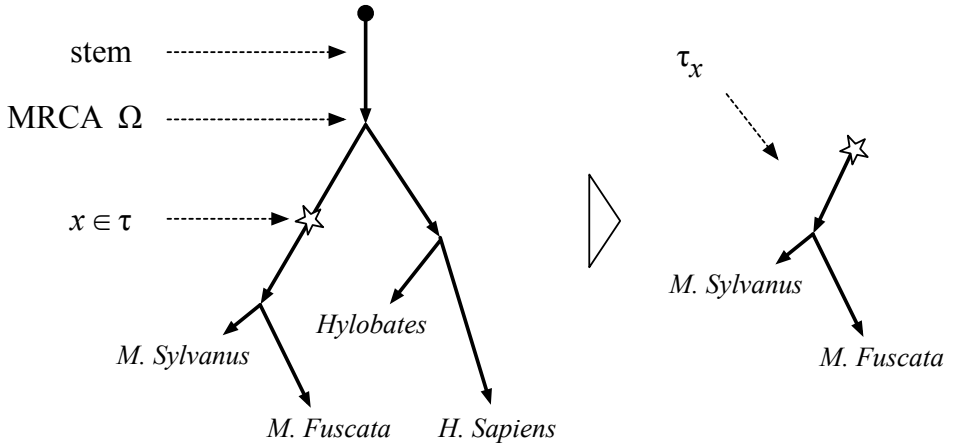


Figure 6.1. Notation used for describing the PSCP.

We will also assume that the phylogenetic tree τ has a branch above the most recent common ancestor Ω (a "stem": see Figure 6.1). Since we will let the length of this stem go to infinity, this branch does not need to be represented in implementations, but it simplifies the notation. Equivalently, we assume that the prior over sequence lengths at Ω is at equilibrium.

A sample from a PSCP prior can be computed as follows (see Figure 6.2):

1. Sample insertion locations from a Poisson process with uniform intensity λ on τ: $\mathbf{X} \sim \mathrm{PP}(\lambda \times \tau)$.

2. Order the insertions uniformly at random: $(X_1, X_2, \ldots, X_I) \sim \mathrm{Perm}(\mathbf{X})$.

3. For each $i \in \{1, 2, \ldots, I\}$, sample a *homology path* H_i from a CTMC along the subtree τ_{X_i} rooted at X_i. This CTMC is based on the rate matrix Q, and an initial distribution π. We use the notation $H_i|X_i \sim \mathrm{CTMC}(\tau_{X_i}, Q, \pi)$ for this sampling step. A homology path is a map from the points of the subtree τ_{X_i} to nucleotides (including the gap character).

4. Define the sample to return by concatenation as follows:

$$H(x,i) = \begin{cases} H_i(x) & \text{if } X_i \preceq x \\ g & \text{o.w.} \end{cases}$$

$$H(x) = H(x,1) \circ H(x,1) \circ \cdots \circ H(x,I)$$

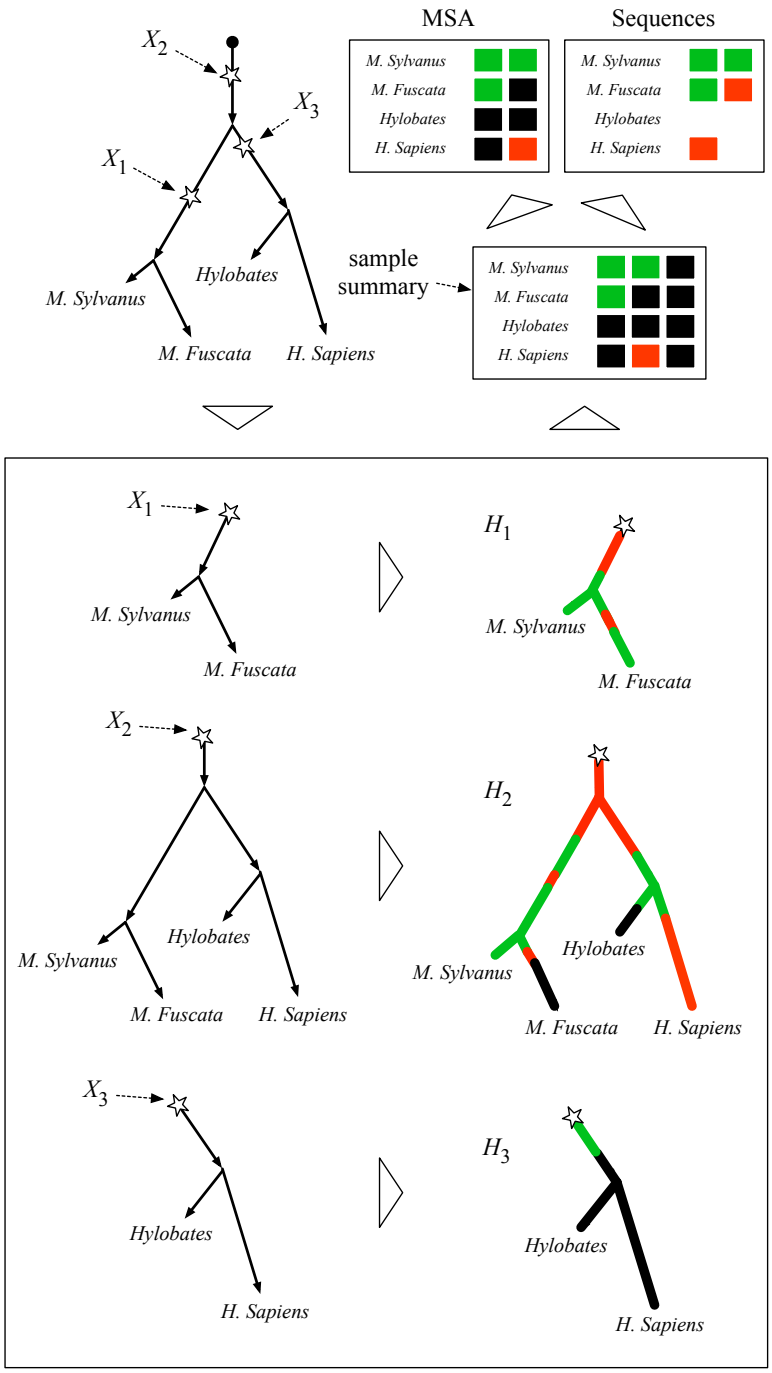


Figure 6.2. Example of a PSCP sample. Here Σ has two symbols, represented by red and green squares.

Given a sample H from the PSCP, one can extract the observed (modern) sequences as

follows: $Y(v) = \rho_g(H(v))$, where $v \in L$, and ρ_g is the projection erasing the gap symbol g from the sequences.

Samples can also be projected into MSAs, which are sets of views on homology paths called columns. Formally:

$$M = \{C : L \to \Sigma_g, C(v) = H(v,i), 1 \leq i \leq I, v \in L, \exists w \in L \text{ s.t. } C(w) \neq g\}.$$

The observed columns are required to have at least one character that is not equal to the gap symbol.[4]

Note that this process assumes that a rooted phylogeny τ has been fixed. We will denote by $p_\tau(y, m)$ the marginal probability that this process generates a MSA m and observations y, $p_\tau(y, m) = \mathbb{P}_\tau(Y = y, M = m)$, integrating over all homology paths. For joint inference, we make phylogenies T random, with a distribution specified by a prior with density $p(\tau)$.

6.2.2 Equivalence of the processes

In this section, we show equality in distribution of the two new processes introduced so far, $H \sim$ CIRP.

The first step is to show that the distribution of the locations of the insertions in τ is the same in the two processes. This fact is established by the following extension to the Poisson Interval Theorem:

Theorem 6. *Let Π be a Poisson Process of constant rate λ on a phylogenetic tree τ, and let $(X_1, \ldots, X_I)$ be an ordering such that $X_i \prec X_j$ for all $i < j$. Let Δ_i be the distance in τ from X_i to the closest insertion point X_j with $X_j \prec X_i$ (setting Δ_i to the distance to the root (top of the stem) if there are no such X_j). Then Δ are independent exponential distributions with rate λ.*

Proof. Let $v \in L$ be an arbitrary but fixed leaf. Consider the smallest subset $\tau' \subset \tau$ connecting v to the root of τ. If we let I_v denote the set of X_i's falling inside τ', we get by applying the Restriction Theorem [42], that the random set $\{X_i : i \in I_v\}$ is distributed according to a Poisson process as well, $\{X_i : i \in I_v\} \sim \text{PP}(\lambda \times \tau')$.

[4]Note that this definition is slightly different than the one of Section 2 (in this new definition, the order of the columns is not maintained, but for the purpose of computing the marginal likelihood in PSCPs, the two representation are equivalent by exchangeability.

Since τ' is Borel isomorphic with the real line, it follows from the Mapping Theorem and the standard Poisson Interval Theorem [42] that the conclusion of the theorem is true for all X_i with $i \in I_v$. By repeating the same argument for all $v \in L$, we get the theorem for all X_i with $i \in \cup_{v \in L} I_v = \{1, \ldots, I\}$. □

The second step is to show that conditioning on insertions, the deletion-substitution distribution is equal in the two processes. This can be easily seen as a consequence of the well-known jump-hold construction of CTMCs [19]:

Theorem 7. *Let X_t be a CTMC with rate matrix $Q = (q_{i,j})$ and let $Y_{i,j}$ be independent exponential random variables with rate $q_{i,j}$. Then*

$$(\Delta, J)|(X_0 = i) \stackrel{d}{=} (\min_{j \neq i} Y_{i,j}, \operatorname*{argmin}_{j \neq i} Y_{i,j}),$$

where $\Delta = \inf\{t : X_t \neq i\}$, $J = X_\Delta$.

6.2.3 Asymptotic behavior

We compute the asymptotic length of the observed sequences under the modified process.

We have:

$$\begin{aligned}
\mathbb{E}[|Y|] &= \mathbb{E}[\mathbb{E}[|Y|||X|]] \\
&= \sum_{n=0}^{\infty} p_{\lambda \times L}(n) \mathbb{E}[|Y|||X| = n] \\
&= \sum_{n=0}^{\infty} p_{\lambda \times L}(n) n \mathbb{P}[H(L) \neq g] \\
&= \mathbb{P}[H(L) \neq g] \sum_{n=0}^{\infty} p_{\lambda \times L}(n) n \\
&= \mathbb{P}[H(L) \neq g](\lambda L).
\end{aligned}$$

At the same time,

$$\begin{aligned}
\mathbb{P}[H(L) \neq g] &= \int_0^L \frac{1}{L}(e^{-\mu t})\, \mathrm{d}t \\
&= \frac{1}{L}\frac{1}{\mu}\left(1 - e^{-\mu L}\right),
\end{aligned}$$

so it follows that:

$$\lim_{L\to\infty} \mathbb{E}[|Y|] = \frac{\lambda}{\mu}.$$

Using this result, we get a more intuitive re-parameterization of the PSCP, in terms of asymptotic expected length L and intensity I:

$$\begin{cases} L = \frac{\lambda}{\mu} \\ I = \lambda \cdot \mu \end{cases} \quad \begin{cases} \mu = \sqrt{\frac{I}{L}} \\ \lambda = L \cdot \sqrt{\frac{I}{L}} \end{cases}$$

6.3 Computational aspects

We start this section by showing that for two popular joint estimators for MSA and phylogeny inference, the main challenge is to compute the marginal probabilities $p_\tau(y, m)$ for a sequence of candidate trees and MSAs.

Maximum likelihood

The objective function typically takes the form:

$$\max_{\tau,m} p_\tau(y, m).$$

This can be optimized for example using simulated annealing, where a candidate phylogeny and MSA (τ', m') are proposed at each step i, and are accepted (meaning that they replaces the previous candidate (τ, m)) according to a sequence of acceptance functions $f^{(i)}(p, p')$ depending only on the marginal probabilities $p = p_\tau(y, m), p' = p_{\tau'}(y, m')$. Provided $\lim_{i\to\infty} f^{(i)}(p, p') = \mathbf{1}[p' > p]$ sufficiently slowly, this algorithm converges to the maximum likelihood phylogeny and MSA.

Bayes estimators

In order to define a Bayes estimator, one typically specifies a decision space D (we have considered for example the space of MSAs in Section 5.2.2.1, so let us consider now the example of the space of multifurcating tree topologies), a projection into this space,

$(\tau, m) \mapsto \rho(\tau, m) \in D$ and a loss function $l : D \to [0, \infty)$ on D (for example, the symmetric clade difference, or partition metric [4]).

In the Bayesian framework, given these objects, the optimal decision (called the consensus tree for the choice of loss function given in the example above), is obtained by minimizing:

$$\underset{d \in D}{\operatorname{argmin}} \, \mathbb{E}[l(d, \rho(T, M))|Y = y].$$

Even when MSAs are held fixed, this expectation is intractable, so it is usually approximated with the empirical distribution of the output $(\tau^{(i)}, m^{(i)})$ of a Markov chain Monte Carlo algorithm.

Producing MCMC samples involves computing acceptance ratios of the form:

$$\frac{p(\tau', m', y)}{p(\tau, m, y)} q\left((\tau, m) \to (\tau', m')\right),$$

for some proposal having a Radon-Nikodym derivative q.

From these two examples, it is clear that joint MSA and phylogeny inference requires a fast way of computing $p_\tau(m, y)$. The other main ingredient is to be able to propose trees and MSAs efficiently. This is discussed in Section 6.3.2.

6.3.1 Computing the marginal likelihood

In this section, we demonstrate the significance of the Poisson Process representation, showing that it drastically simplifies the computation of marginal likelihoods.

The first step is to condition on the number of homology paths, $|\mathbf{X}|$. While the number of homology paths is random and unknown, we know it must be greater than the number of columns $|m|$ in the observed alignment. We need to consider an unknown and unbounded number of birth events with no observed offsprings in the MSA, but as they

are exchangeable, they can be easily marginalized analytically. This is done as follows:

$$\begin{aligned}
\mathbb{P}(Y = y, M = m) &= \mathbb{E}[\mathbb{P}(Y = y, M = m||\mathbf{X}|)] \\
&= \sum_{n=0}^{\infty} \mathbb{P}(|\mathbf{X}| = n) \cdot \mathbb{P}(Y = y, M = m||\mathbf{X}| = n) \\
&= \sum_{n=|m|}^{\infty} \mathbb{P}(|\mathbf{X}| = n) \cdot (Z(c_\emptyset))^{n-|m|} \prod_{c \in m} Z(c) \\
&= \left(\prod_{c \in m} Z(c) \right) \sum_{n=|m|}^{\infty} \xi_n \cdot (Z(c_\emptyset))^{n-|m|}
\end{aligned}$$

where $\xi_n = \mathbb{P}(|\mathbf{X}| = n) = \frac{\lambda^n}{e^\lambda n!}$ is a Poisson density, $Z(c) = \mathbb{P}(C = c)$ is the likelihood of a single MSA column c, and $c_\emptyset$ is a column with a gap at every leaf $v \in L$: $c_\emptyset \equiv g$.

This expression can be simplified by using the following identity:[5]

$$\begin{aligned}
\varphi(q, N) &= \sum_{n=N}^{\infty} \xi_n q^{n-N} \\
&= \frac{1}{q^N} \left(\sum_{n=0}^{\infty} \xi_n q^n - \sum_{n=0}^{N-1} \xi_n q^n \right) \\
&= \frac{1}{q^N} \left(e^{\lambda(q-1)} - \sum_{n=0}^{N-1} \xi_n q^n \right).
\end{aligned}$$

We get the simple formula:

$$\mathbb{P}(Y = y, M = m) = \varphi(Z(c_\emptyset), |m|) \prod_{c \in m} Z(c),$$

The next step is to compute the marginal $Z(c)$ of an alignment column c. We compute this by first conditioning on the random edge E on which is located the unique insertion point X from where all the nucleotides in the current column c descend from. Computing the prior probability that an insertion happens along a given edge is greatly simplified by the following property of Poisson Processes:

Theorem 8. *Let $\Pi \sim \mathrm{PP}(\nu)$ be the sample from a Poisson Process over S with finite intensity measure ν, $\nu(S) = \lambda < \infty$, $N \sim \mathrm{Poi}(\lambda)$, and X_i be iid $\frac{\nu}{\lambda}$, then $\Pi \stackrel{d}{=} \{X_1, X_2, \ldots, X_N\}$.*

[5] In implementations, we found that the expression $\sum_{n=N}^{\infty} \xi_n q^{n-N}$ numerically worked better than $\frac{1}{q^N} \left(e^{\lambda(q-1)} - \sum_{n=0}^{N-1} \xi_n q^n \right)$, however the second expression is more convenient for deriving parameter estimation updates in Appendix C.5.

Suppose that we want to compare the likelihoods that an insertion occurred in two edges $e, e' \in \mathscr{E}$ of finite lengths (the case where one of the edge is the infinite length stem is covered in Appendix C.1). We apply the theorem as follows: we set ν to the uniform measure over τ, and we get the following formula:

$$\frac{P(E = e | E \in \{e, e'\})}{P(E = e' | E \in \{e, e'\})} = \frac{b(e)}{b(e')}.$$

This means that the probability of insertion along an edge (other than the stem) is proportional to the branch length of this edge. We use the following notation for this probability:

$$\begin{aligned} \iota_v &= \mathbb{P}(\mathcal{E}_v) \\ \mathcal{E}_v &= (E = (\mathrm{pa}(v) \to v)) . \end{aligned} \tag{6.1}$$

Again see Appendix C.1 for the general formula for computing ι_v.

Note that since the ι's need not be recomputed for each column $c \in m$ (as they depend only on τ), facilitating implementation.

From the ι's, the column likelihoods are computed as follows:

$$\begin{aligned} \mathbb{P}(\mathcal{C}) &= \sum_{v \in \mathscr{V}} \mathbb{P}(\mathcal{E}_v) \mathbb{P}(\mathcal{C} | \mathcal{E}_v) \\ &= \sum_{v \in \mathscr{V}} \iota_v f_v, \end{aligned}$$

where f_v is just the output of a slight modification Felsenstein's peeling recursion [21] applied on the subtree rooted at v (see Appendix C.3).

Since computing the peeling recursion for one column takes time $O(|L|)$, we get a total running time of $O(|L| \cdot |m|)$, where $|L|$ is the number of observed taxa, and $|m|$ is the number of sites in the alignment.

6.3.2 Proposal distributions

Several objects need to be resampled in the joint MCMC inference algorithm: the tree topology, the branch lengths, the MSA, and the parameters.

For trees and branch lengths, we use standard proposal mechanisms, described in [46]. More precisely: we unrooted the tree while keeping track of the edge e where the rooting

was sitting. We then apply a stochastic neighbor interchange, a branch length resampling, or both, to the unrooted tree. If the edge e is unchanged after this step, we reroot at the same place; if the length of e was rescaled, we reroot proportionally; if e does not exists after the resampling move, we pick a rerooting uniform at random.

The proposal over MSAs we used operates on linearized MSAs, so let m_0 denote the current MSA, and l_0, its linearization (i.e. the linearization is an auxiliary variable). Our proposals over MSAs works as follows: First, partition the leaves into two sets A, B.[6] The support of the proposal is the set S of linearized MSAs m, l satisfying the following constraints, for all :

1. If e has both end points in A (or both in B), then $e \in m \iff e \in m_0$.
2. If e, e' have both end points in A (or both in B), then $e \prec_l e' \impliedby e \prec_{l'} e'$.

We propose an element $m \in S$ with probability proportional to $\prod_{c \in m} Z(c)$. The set S has exponential size, but can be sampled efficiently using a technique similar to the "constrained sampling procedure" of [74]. Note that the proposal induces an irreducible chain: one possible outcome of the move is to remove all links between two groups of sequences. The chain can therefore move to the empty MSA and then construct any MSA incrementally.

For parameters, we used the multiplicative proposals of [46] on the λ, μ parameterization.

6.4 Experiments

6.4.1 Simulations

We used synthetic data to assess the quality of the tree reconstructions produced by PSCP, compared to the reconstructions of PhyML, a state-of-the-art tree inference system based on maximum likelihood [79]. We also compared the inferred MSAs to those produced by Clustal [29], the most popular MSA inference system.

In this study, we explored four types of potential improvements:

[6] In practice, we found that using the bipartitions such that $|A| = 1$ is sufficient.

Exp.	Tree resampled?	No	Yes	No	Yes
	MSA resampled?	No	No	Yes	Yes
MSAs	Edge recall (SP)	**0.25**	-	0.22	0.24
	Edge Precision	0.22	-	0.56	**0.58**
	Edge F1	0.23	-	0.31	**0.32**
Trees	Partition Metric	0.24	0.22	-	**0.19**
	Robinson-Foulds	0.45	0.38	-	**0.33**

Table 6.1. PSCP results on simulated data. Note that scores for trees are losses (lower is better), while scores for MSAs are accuracies (higher is better).

1. Resampling trees and MSAs increasing the quality of inferred MSAs, compared to resampling only MSAs.

2. Resampling trees and MSAs increasing the quality of inferred trees, compared to resampling only trees.

3. Resampling trees increasing the quality of inferred trees, compared to trees inferred by PhyML.

4. Resampling MSAs increasing the quality of inferred MSAs, compared to MSAs inferred by Clustal.

The results are show in Table 6.1. We observed improvements of all four types. Comparing Edge F1 relative improvements to Robinson-Foulds relative improvements, the relative additional improvement of type (2) is larger (13%) than that of type (1) (3%). But overall (i.e. comparing the baselines to the joint system), the full improvements of both trees and MSAs are substantial: 43% Edge F1 improvement, and 27% Robinson-Foulds improvement. All these experiments are based on 100 replica, each having 7 taxa at the leaves, a topology sampled from the uniform distribution, and branch lengths sampled from rate 2 exponential distributions.

We also tested our system on data generated from the TKF91 model instead of the PSCP model. We used the same tree distribution and number of replica as in the previous experiments, and the same generating TKF91 parameters as [33].

We again observed improvements over the baselines, both in terms of MSA and tree quality. For MSAs, the relative improvement over the baseline was actually larger on the

	5S	
Metric	PSCP	Clustal [29]
Edge recall (SP)	0.43	**0.68**
Edge precision	**0.71**	0.66
Edge F1	0.47	**0.67**

Table 6.2. PSCP results on the comparative RNA dataset

	5S+16S	
Metric	PSCP	Clustal [29]
Edge recall (SP)	0.47	**0.60**
Edge precision	**0.64**	0.44
Edge F1	**0.49**	0.45

Table 6.3. PSCP results on the comparative RNA dataset with outliers.

TKF91-generated data than on the PSCP-generated data (47% versus 43%, as measured by Edge F1 improvement over Clustal), but lower for phylogenetic trees (13% versus 27%, as measured by Robinson-Foulds improvement over PhyML).

6.4.2 Real data

We have used the same protocol as in Section 5.3 to generate random subsets of the 5S portion of the Comparative RNA dataset [39], and following [80], we have created another version with one outlier, taken from 16S. We have additionally truncated annotated alignments to 100 sites to speed-up the comparisons.

In the 5S dataset, we obtained a higher precision than Clustal, but a lower recall and F1 scores (see Table 6.2). We believe that this is due to the lack of long indel modeling in these experiments. We discuss in Section 6.5 how this limitation can be removed.

The 5S+16S dataset results (Table 6.3) demonstrate that our system is less sensitive to outliers than Clustal: in this case, PSCP outperforms Clustal both in edge precision and edge F1 scores.

6.5 Discussion

We have shown that our system achieves state-of-the-art performance in terms of the quality of the inferred trees. In terms of MSA performance, we achieved higher precision than Clustal, but at the cost of lower recall.

We believe that we could increase alignment recall by incorporating atomic long indel into the model. Our system outperforming in recall and precision Clustal in the synthetic, point-mutation experiments supports this thesis. We have also seen in Section 5.3, that long indel modeling is crucial to get MSA performance that is competitive across all metrics on real data. It is remarkable that the joint system is already competitive in some of the conditions studied in the last section (e.g. in the presence of outliers).

A natural question is then how to create stochastic processes incorportating long indel modeling. Extending the TKF91 model to handle long indel explicitly is non-trivial. There has been several attempts in the past, but none have been fully satisfactory. These approaches can be categorized as follows:

1. One approach [88] has been to globally segment the observed sequences and model the evolution of each segment using a TKF91 model. This model is known as the TKF92 model. Unfortunately, when there are several taxa under study, this approach does not scale well—one ends up needing most segments having length one, returning to TKF91.

2. Other approaches have attempted to add to TKF91 long indels as atomic events. Unfortunately, this make computation of finite-dimensional marginals difficult. A closed form expression for the marginals has been developed for the case where only long insertions are considered [61], but not for the case where both long insertions and deletions are present. Numerical computation of the marginal has been attempted in [60], but the empirical results were mixed. Moreover, numerical solutions are less attractive in joint models, as resampling trees required repeated recomputation of these marginals.

We believe that PSCP is a better foundation to construct long indel models. The method we propose is close to the approach of the second category above. Fortunately, the computations are much easier than in the case of TKF91 extensions, thanks to the following theorem [42]:

Theorem 9. *Let* Π *be a Poisson process with mean measure* μ*. Let the points of* Π *be colored randomly with* k *colors, the probability that a point receives the* i*-th color being* p_i *and the colors of different points being independent (of one another and of the position of the points). Let* Π_i *be the set of points with the* i*-th color. Then* Π_i *are independent Poisson processes with mean measures* $\mu_i = p_i\mu$*.*

Using the Poisson representation, this means that a long indel process can be constructed as the union of Poisson processes, $P_1 \cup (\cup_{n=2}^{\infty} I_n) \cup (\cup_{n=2}^{\infty} D_n)$, where P_1 is a PSCP, I_n are insertion processes (PSCPs where insertions have length n instead of 1), and D_n are long deletion processes (which are simpler than PSCPs: just a Poisson process picking the location of the long indel).

In contrast to long indel processes generalizing TKF91, an efficient MCMC sampler for this long-indel model can be constructed easily. This is done as follows: first, we can exploit the decomposition and the algorithm of Section 6.3.1 to marginalize out P_1. Second, the other terms of the sum are represented explicitly as auxiliary variables. Since we have an efficient algorithm for computing the marginal likelihood, the auxiliary variables can be resampled easily. Note that designing an irreducible sampler without marginalizing P_1 would be very difficult. Integrating out P_1 creates a bridge of positive probability between any patterns of long indels.

Superpositions of PSCPs have other applications outside of long indel modeling. For example, they could be used to represent punctuated changes. An important example of punctuated changes, systematic sound changes, was described in the context of diachronic phonology in Section 2.6. To model these, we would again write the process as a superposition, $P_1 \cup P_{\text{type}}$, where P_1 is the standard PSCP (which act at the token level), and P_{type} is a Poisson process inducing type-level changes. Again, inference can be carried in this representation by marginalization of P_1 and representing P_{type} as an auxiliary variable.

Finally, another avenue to improve PSCP models is to make the insertion rate mean measure more realistic: instead of being uniform across the tree, it could be modeled using a parametric function, hence forming a Cox process. This would be most useful when the sequences under study have very different lengths.

Chapter 7

Conclusions

Phylogenetics is a great challenge for unsupervised learning, both in statistical and computational terms. In this thesis, we have shown how the challenging aspects of phylogenetics can motivate new advances in unsupervised learning, and how this in turn move forwards the field of phylogenetics. In this concluding chapter, we summarize our contributions and propose new research directions.

7.1 Summary

The main contributions can be summarized as follows:

A probabilistic approach to language change: We developed the first computational approach that can scale to large scale phylogenies. Sound changes and markedness are taken into account using a flexible feature-based unsupervised learning framework. We systematically and quantitatively evaluated and validated the system using held-out reconstruction experiments. Using this model, we attacked a 50-year-old open problem in linguistics regarding the role of functional load in language change.

New algorithms for MSA: We presented three novel algorithms for inferring multiple sequence alignments. We analyze and explained the comparative strengths and weaknesses of each of these algorithms.

Exact phylogenetic inference: We introduced a matrix-based framework for analyzing exact inference algorithms for string-valued graphical models. As a corollary, we get simple proofs for the complexity upper bound of computing exact marginals in various string-valued stochastic processes.

Approximate phylogenetic inference: We described two approximation algorithms for computing conditional expectations in phylogenetic models. One is based on MCMC, and the other, on variational inference.

Variational inference over combinatorial space: The technique we used to develop the variational phylogenetic inference algorithm extends to many other situations. We created a framework for developing variational algorithms in these situations. As special cases, we obtained new algorithms for summing over linearizations of partial orders, traveling salesman problems, and plane partitions.

Stochastic processes: We created a new string-valued CTMC to model evolution and language change. In contrast to previous processes, the marginal likelihood of the new process can be computed in fully polynomial time (polynomial in the length of the sequences and in the number of taxa). We used this process to create a joint stochastic process over MSAs and trees, and evaluated quantitatively the quality of both outputs.

7.2 The future

We conclude by giving a few examples of potential research avenues building on the work of this thesis.

Joint biological and linguistic phylogenetic inference: While there is an impressive literature on how to perform phylogenetic inference in biology and linguistics, there is currently no work that jointly leverages biological and linguistic sources of information. Modeling both domains jointly is attractive for many reasons. For example, adding genetic data to linguistic studies can help dating ancestral events: dates estimated from purely linguistic character are not widely accepted, because the linguistic change process is not as well understood as the biological one.

Sequential Monte Carlo (SMC) for phylogenetic inference: We have seen inference algorithms from the MCMC and variational frameworks. Is it possible to use

SMC algorithms as well? Compared to MCMC and variational methods, SMC algorithms have interesting tradeoffs. The challenge is to extend SMC, usually restricted to state-space models, to combinatorial spaces.

Characterization of the set of tractable string-valued stochastic processes: Given a string-valued jump process where the jumps depend only a localized context, are the marginals guaranteed to be weighted transducers? If so, can we bound the size of the state space and give efficient algorithms to approximate the weights?

Non-Parametric Phylogenetic Forests: The assumption that there is a single, tree-shaped phylogenetic graph is frequently violated in biology and linguistics. These violations can be caused by language contact, lateral gene transfers, population structure, hybridization and creolization. The traditional approach to handle these violations has been to use networks instead of trees [64], but this has limitations. Some of these limitations are computational (the marginal likelihood becomes intractable), others are representational (knowing that a taxon comes from two ancestors for example does not tell which part of the sequences comes from which ancestor). One alternative is to allow each site to be modeled by its own tree, where the trees can vary across sites, thus forming a phylogenetic forest. A suitable model should encourage agreements across sites, but not require perfect agreement. One avenue to approach this requirement is to use non-parametric priors (so that a bound on the number of trees in the forest is not set a priori) that have these properties while allowing tractable inference.

Computational Approaches to Syntactic and Morphological Change: Many types of linguistic data (syntactic and morphological in particular) are being used by historical linguists to study language change [31]. This project seeks to accommodate these other types of data in a computational framework. For syntax, one potential approach is to represent grammars as vectors in an abstract vector space (for example, the weight vector of a weighted context-free grammar), and to use a Brownian motion on these vectors to model change.

More broadly, these contributions, current and future, fit inside a long-term program. The objectives of this program include:

Statistical Regularities: What are the most frequent and important types of change? In diachronic phonology in particular (the branch of historical linguistics concerned

with sound changes), quantitative properties of these regularities are essentially unknown [32]. The ramifications of this problem go far beyond the simple functional load statistics described in Section 3.4.6. In biology, basic substitution regularities have been intensely studied, but more complex types of change are still poorly understood (such as context-sensitive changes and long indels, role of fitness beyond affecting the overall rate of change).

Large Scale Topologies: What is the broad topology and rooting of the phylogenetic graph of life and languages? In particular, which of the current language families are likely to be related or unrelated? These questions are highly challenging because of the amount of change involved [24]. For example, a relationship between a language family in the New World and one in the Old World was only confirmed in 2008 [90].

Dating: Can we put date estimates on important speciation and migration events? Examples include human migrations into America, or the age of the last universal common ancestor. Many of these questions are likely to be fertile grounds for research, involving both phylogenetics and population genetics [27].

Appendix A

Exact Inference Equations

A.1 Epsilon removal and normalization

Let $s \in \Sigma^*$ be a string of length N. We first prove that the epsilon-removed automaton M'_α of Section 4.1.3 assigns the same weights $w'(s)$ to strings as the original automaton $M_{\hat{\alpha}}$, $w(s)$. Let $s \in \Sigma^*$. We have:

$$\begin{aligned}
w(s) &= \phi\left(\sum_{\hat{s}\equiv s}\prod_{\hat{\alpha}\in\hat{s}} M_{\hat{\alpha}}\right) \\
&= \phi\left(\sum_{(k_1,\ldots,k_N)\in\mathbb{N}^N} M_\epsilon^{k_1} M_{s_1} M_\epsilon^{k_2} M_{s_2} \cdots M_\epsilon^{k_N} M_{s_1}\right) \\
&= \phi\left(\prod_{n=1}^{N}\left(\sum_{m=0}^{\infty} M_\epsilon^m\right) M_{s_n}\right) \\
&= \phi\left(\prod_{n=1}^{N} M'_{s_n}\right) \\
&= w'(s)
\end{aligned}$$

Here we used the nonnegativity of the weights, which implies that if the automaton has a finite normalization, the infinite sums above will be absolutely convergent, and can therefore be rearranged.

Next, we prove the formula for the normalization:

$$\begin{aligned}\phi\left(\left(\sum_{\alpha\in\hat{\Sigma}} M_{\hat{\alpha}}\right)^*\right) &= \phi\left(\sum_{N=0}^{\infty}\left(\sum_{\alpha\in\hat{\Sigma}} M_{\hat{\alpha}}\right)^N\right)\\ &= \phi\left(\sum_{N=0}^{\infty}\sum_{(\hat{\alpha}_1,\ldots,\hat{\alpha}_N)\in\hat{\Sigma}^L}\prod_{n=1}^{N} M_{\alpha_n}\right)\\ &= Z\end{aligned}$$

A.2 Implementation of probabilistic operations

Fix $s \in \Sigma$. We start by establishing the marginalization formula:

$$\begin{aligned}\sum_{s'\in\Sigma^*} w(s,s') &= \phi\left(\sum_{s'\in\Sigma^*}\sum_{N=0}^{\infty}\sum_{\hat{s}\equiv_L s}\sum_{\hat{s}'\equiv_L s'}\prod_{n=1}^{N} M_{\hat{s}(n),\hat{s}'(n)}\right)\\ &= \phi\left(\sum_{N=0}^{\infty}\sum_{\hat{s}\equiv_L s}\sum_{\hat{s}'\in\hat{\Sigma}:|\hat{s}'|=L}\prod_{n=1}^{N} M_{\hat{s}(n),\hat{s}'(n)}\right)\\ &= \phi\left(\sum_{N=0}^{\infty}\sum_{\hat{s}\equiv_L s}\prod_{n=1}^{N}\sum_{\hat{\alpha}\in\hat{\Sigma}} M_{\hat{s}(n),\hat{\alpha}}\right)\\ &= w'(s)\end{aligned}$$

Here, we write $\hat{s} \equiv_L s$ if $\hat{s}$ has length L and $\hat{s} \equiv s$.

Next, we prove the formula for pointwise products of the first kind:

$$\begin{aligned}w^{(1)}(s)\cdot w^{(2)}(s) &= \phi\left(\sum_{\hat{s}\equiv s}\prod_{\hat{\alpha}\in s} M^{(1)}_{\hat{\alpha}}\right)\phi\left(\sum_{\hat{s}'\equiv s}\prod_{\hat{\alpha}'\in s'} M^{(1)}_{\hat{\alpha}'}\right)\\ &= \phi\left(\prod_{\alpha\in s} M^{(1)}_{\alpha}\right)\phi\left(\prod_{\alpha\in s} M^{(2)}_{\alpha}\right)\\ &= \phi\left(\prod_{\alpha\in s}\left\{M^{(1)}_{\alpha}\middle| M^{(2)}_{\alpha}\right\}\right)\\ &= w^{(\mathrm{p})}(s),\end{aligned}$$

where we have use the epsilon-free assumption between lines 1 and 2.

Next, we establish the formula for pointwise products of the second kind:

$$
\begin{aligned}
w^{(1)}(s, s') \cdot w^{(2)}(s) &= \phi\left(\sum_{N=0}^{\infty} \sum_{\hat{s} \equiv_N s} \sum_{\hat{s}' \equiv_N s} \prod_{n=1}^{N} M^{(1)}_{\hat{s}_n, \hat{s}'_n}\right) \phi\left(\prod_{\alpha \in s} M_\alpha\right) \\
&= \sum_{N=0}^{\infty} \sum_{\hat{s} \equiv_N s} \sum_{\hat{s}' \equiv_N s} \phi\left(\prod_{n=1}^{N} M^{(1)}_{\hat{s}_n, \hat{s}'_n}\right) \phi\left(\prod_{n=1}^{N} \begin{cases} I & \text{if } \hat{s}_n = \epsilon \\ M^{(2)}_{\hat{s}_n} & \text{o.w} \end{cases}\right) \\
&= \sum_{N=0}^{\infty} \sum_{\hat{s} \equiv_N s} \sum_{\hat{s}' \equiv_N s} \phi\left(\prod_{n=1}^{N} M^{(\mathrm{p})}_{\hat{s}_n, \hat{s}'_n}\right) \\
&= w^{(\mathrm{p})}(s, s')
\end{aligned}
$$

Appendix B

Derivations of the Variational Framework

B.1 Markov random field reformulation

We prove in this section that under the Rich Sufficient Statistics condition (RSS)[1], the log-partition function is the same in the original exponential family and in the bipartite MRF described in Section 5.4.1. Let us denote the latter log-partition function by $\tilde{A}(\boldsymbol{\theta})$.

We first prove the following identity, introduced previously as Equation (5.5):

Lemma 10.

$$\sum_{s_1 \in \{0,1\}} \sum_{s_2 \in \{0,1\}} \cdots \sum_{s_J \in \{0,1\}} \prod_{i=1}^{I} \prod_{j=1}^{J} \mathbf{1}[\phi_j(x_i) = s_j] = \begin{cases} 1 & \text{if } x_1 = x_2 = \cdots = x_I \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Suppose first that there are indices i', i'' such that $x_{i'} \neq x_{i''}$. By the RSS condition, this means that there is at least one j_0 such that $\phi_{j_0}(x_{i'}) \neq \phi_{j_0}(x_{i''})$. Since the product $\prod_{i=1}^{I} \prod_{j=1}^{J} \mathbf{1}[\phi_j(x_i) = s_j]$ contains both the factor $\mathbf{1}[\phi_{j_0}(x_{i'}) = s_{j_0}]$ and $\mathbf{1}[\phi_{j_0}(x_{i''}) = s_{j_0}]$, for any fixed term in the iterated sum, at least one of the two factors will be equal to zero.

[1] We make the observation in passing that the RSS condition can also be described tersely as the requirement that the σ-algebra generated by the sufficient statistics be equal to the base σ-algebra: $\sigma(\phi_1, \ldots, \phi_J) = \mathcal{F}$.

Conversely, if $x = x_1 = x_2 = \cdots = x_I$, then only the multi-index $(s_1, s_2, \ldots, s_J) = (\phi_1(x), \phi_2(x), \ldots, \phi_J(x))$ in the iterated sum induces a non-zero term. □

A slight extension of this argument yields:

Lemma 11. *For all $\boldsymbol{x} = (x_1, \ldots, x_I)$, where $x_i \in \boldsymbol{\mathcal{X}}$, we have:*

$$\begin{aligned}\Psi(\boldsymbol{x}) &= \sum_{s_1 \in \{0,1\}} \sum_{s_2 \in \{0,1\}} \cdots \sum_{s_J \in \{0,1\}} \left\{ \prod_{i=1}^{I} \prod_{j=1}^{J} \Psi_{i,j}(x_i, s_j) \right\} \left\{ \prod_{j=1}^{J} \Psi_j(s_j) \right\} \left\{ \prod_{i=1}^{I} \Psi_i(x_i) \right\} \\ &= \begin{cases} \exp\left\{ \langle \boldsymbol{\phi}(x_1), \boldsymbol{\theta} \rangle \right\} \nu(x_1) & \textit{if } x_1 = x_2 = \cdots = x_I \\ 0 & \textit{otherwise.} \end{cases}\end{aligned}$$

Using this lemma, we can prove the main proposition:

Proposition 12. *Under RSS, $\tilde{A}(\boldsymbol{\theta}) = A(\boldsymbol{\theta})$.*

Proof. We have:

$$\begin{aligned}\exp \tilde{A}(\boldsymbol{\theta}) &= \sum_{x_1 \in \boldsymbol{\mathcal{X}}_1} \sum_{x_2 \in \boldsymbol{\mathcal{X}}_2} \cdots \sum_{x_I \in \boldsymbol{\mathcal{X}}_I} \Psi(\boldsymbol{x}) \\ &= \sum_{x_1 \in \boldsymbol{\mathcal{X}}_1} \sum_{x_2 \in \boldsymbol{\mathcal{X}}_2} \cdots \sum_{x_I \in \boldsymbol{\mathcal{X}}_I} \begin{cases} \exp\left\{ \langle \boldsymbol{\phi}(x_1), \boldsymbol{\theta} \rangle \right\} \nu(x_1) & \text{if } x_1 = x_2 = \cdots = x_I \\ 0 & \text{otherwise} \end{cases} \\ &= \sum_{x \in \boldsymbol{\mathcal{X}}} \exp\left\{ \langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle \right\} \nu(x) \\ &= \exp A(\boldsymbol{\theta}).\end{aligned}$$

□

B.2 More information on the algorithms

In this appendix, we provide more information regarding the derivation of the variational algorithms discussed in this chapter.

MFMF$(\boldsymbol{\theta}, A_1, \ldots, A_I)$

1: $\zeta_{i,j}^{(1)} = 0$
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $\quad \boldsymbol{\xi}^{(t)} = \boldsymbol{\theta} + \sum_i \boldsymbol{\zeta}_i^{(t-1)}$
4: $\quad \boldsymbol{\zeta}_i^{(t)} = \text{logit}\left(\nabla A_i\left(\boldsymbol{\xi}^{(t)}\right)\right)$
5: **end for**
6: **return** $\hat{\boldsymbol{\mu}} = \text{logistic}(\boldsymbol{\xi})$

TRWMF$(\boldsymbol{\theta}, A_1, \ldots, A_I, \boldsymbol{\rho})$

1: $\zeta_{i,j}^{(1)} = 0$
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $\quad \lambda_{i,j}^{(t)} = \theta_j + \sum_{i':i'\neq i} \rho_{i'\to j}\zeta_{i',j}^{(t-1)} - \rho_{i\to j}\zeta_{i,j}^{(t-1)}$
4: $\quad \bar{\xi}_{i,j}^{(t)} = \rho_{j\to i}\lambda_{i,j}^{(t)}$
5: $\quad \delta_{i,j}^{(t)} = (1 - 2\rho_{j\to i})\lambda_{i,j}^{(t)}$
6: $\quad \boldsymbol{\zeta}_i^{(t)} = \text{logit}\left(\nabla A_i\left(\bar{\boldsymbol{\xi}}_i^{(t)}\right)\right) - \boldsymbol{\delta}_i^{(t)}$
7: **end for**
8: **return** $\hat{\boldsymbol{\mu}} = \text{logistic}\left(\boldsymbol{\theta} + \sum_i \boldsymbol{\zeta}_i^{(T)}\right)$

Figure B.1. Pseudocode for Mean Field Measure Factorization and Tree-Reweighted Measure Factorization. The vector $\boldsymbol{\rho}$ is the collection of marginals of a distribution of spanning trees over $K_{I,J}$. Note that these marginals can also be updated, see [94] for details.

BPMF

The BPMF algorithm maintains at each iteration the quantities $\boldsymbol{\zeta}_i, \bar{\boldsymbol{\xi}}_i$, and super-partition functions $A_i(\bar{\boldsymbol{\xi}}_i)$. Starting with $\zeta_{i,j}^{(0)} = 0$, we use the following updates at each iteration $t = 1, 2, \ldots, T$:

$$\bar{\boldsymbol{\xi}}_i^{(t)} = \boldsymbol{\theta} + \sum_{i':i'\neq i} \boldsymbol{\zeta}_{i'}^{(t-1)}$$
$$\boldsymbol{\zeta}_i^{(t)} = \text{logit}\left(\nabla A_i\left(\bar{\boldsymbol{\xi}}_i^{(t)}\right)\right) - \bar{\boldsymbol{\xi}}_i^{(t)},$$

where the logit function of a vector $\text{logit}\,\boldsymbol{v}$ is the vector of the logit function applied to each entry of the vector $\boldsymbol{v}$, and we use the convention $(\pm\infty) - (\pm\infty) = \pm\infty$.

The approximation of the moments $\mu_j = \nabla_j A(\boldsymbol{\theta})$ is proportional to the product of all the incoming messages at the last iteration T, times the local potential, $\prod_j m_{i\to j}(s)\Psi_j(s)$:

$$\frac{\hat{\mu}_j}{1 - \hat{\mu}_j} = \frac{\prod_j m_{i\to j}^{(T)}(1) e^{\theta_j}}{\prod_j m_{i\to j}^{(T)}(0) e^0}.$$

Using the notation $\text{logistic}(\boldsymbol{v})_j = (1 + \exp(-v_j))^{-1}$, this is equivalent to:

$$\hat{\boldsymbol{\mu}} = \text{logistic}\left(\boldsymbol{\theta} + \sum_i \boldsymbol{\zeta}_i^{(T)}\right).$$

TRWMF

We now derive Equation (5.9). We start from the explicit TRW updates, and show how to make the large messages implicit:

$$m_{i\to j}(s) \propto \sum_{x\in\boldsymbol{\mathcal{X}}} \mathbf{1}[\phi_j(x)=s]\nu_i(x)\frac{\prod_{j':j'\neq j}\left(M_{j'\to i}(x)\right)^{\rho_{j'\to i}}}{\left(M_{j\to i}(x)\right)^{1-\rho_{i\to j}}},$$

where $\rho_{i\to j}$ are marginals of a spanning tree distribution over $K_{I,J}$.

Again, the idea is to find a parameter vector $\boldsymbol{\xi}_{i,j}\in\mathbb{R}^J$ such that

$$\frac{\prod_{j':j'\neq j}\left(M_{j'\to i}(x)\right)^{\rho_{j'\to i}}}{\left(M_{j\to i}(x)\right)^{1-\rho_{i\to j}}} \propto \exp\langle\boldsymbol{\phi}(x),\boldsymbol{\xi}_{i,j}\rangle. \tag{B.1}$$

To do this, we start by rewriting the numerator of the left hand side of Equation (B.1):

$$\begin{aligned}
\prod_{j':j'\neq j}\left(M_{j'\to i}(x)\right)^{\rho_{j'\to i}} &= \prod_{j':j'\neq j}\left(\frac{e^{\theta_{j'}\phi_{j'}(x)}\prod_{i':i'\neq i}\left(m_{i'\to j'}(\phi_{j'}(x))\right)^{\rho_{i'\to j'}}}{\left(m_{i\to j'}(\phi_{j'}(x))\right)^{\rho_{i\to j'}}}\right)^{\rho_{j'\to i}}\\
&= \exp\left\{\sum_{j':j'\neq j}\rho_{j'\to i}\Bigg(\theta_{j'}\phi_{j'}(x)+\sum_{i':i'\neq i}\rho_{i'\to j'}\log m_{i'\to j'}(\phi_{j'}(x))\right.\\
&\quad\left.-\rho_{i\to j'}\log m_{i\to j'}(\phi_{j'}(x))\Bigg)\right\}\\
&\propto \exp\left\{\sum_{j':j'\neq j}\rho_{j'\to i}\phi_{j'}(x)\left(\theta_{j'}+\sum_{i':i'\neq i}\rho_{i'\to j'}\zeta_{i',j'}-\rho_{i\to j'}\zeta_{i,j'}\right)\right\},
\end{aligned}$$

where we have used in the last step the assumption that ϕ_j has domain $\{0,1\}$, which implies that $m_{i\to j}(\phi_j(x)) = \exp\{\phi_j(x)\log m_{i\to j}(1)+(1-\phi_j(x))\log m_{i\to j}(0)\} \propto \exp\{\phi_j(x)\zeta_{i,j}\}$.

A similar argument on the denominator of the left hand side of Equation (B.1) yields:

$$\left(M_{j\to i}(x)\right)^{1-\rho_{i\to j}} \propto \exp\left\{(1-\rho_{i\to j}\phi_j(x)\left(\theta_j+\sum_{i':i'\neq i}\rho_{i'\to j}\zeta_{i',j}-\rho_{i\to j}\zeta_{i,j}\right)\right\}.$$

Combining these gives the update:

$$\left(\boldsymbol{\xi}_{i,j}\right)_{j'} = \left(\theta_{j'}+\sum_{i':i'\neq i}\rho_{i'\to j'}\zeta_{i',j'}-\rho_{i\to j'}\zeta_{i,j'}\right)\cdot\begin{cases}\rho_{j'\to i} & \text{if } j'\neq j\\ (1-\rho_{i\to j}) & \text{otherwise.}\end{cases}$$

Finally, applying the argument introduced in Section 5.4.3 yields the reparameterized updates shown in Figure B.1.

B.3 Computing matching factorizations

In this section, we show how to compute efficiently the super-partition functions described in the matching examples of Section 5.4 and 5.4.6.1.

Proposition 13. *For perfect bipartite matchings in SBM, computing one super-partition function $A_i(\boldsymbol{\theta})$ takes time $O(N^2)$.*

Proof. For this type of super-partition function, we claim that computation simply involves renormalizing rows or columns of a matrix. We first introduce some notation: let the sufficient statistic coordinate j corresponds to the indicator $x_{m,n}$, and $\boldsymbol{X}_i$ denote a random variable distributed according to the member indexed by $\boldsymbol{\theta}$ in the exponential family with base measure ν_i and sufficient statistics $\boldsymbol{\phi}$. Note that in the case of SBM, this corresponds to a distribution over functions of the form $f : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, N\}$.

With this notation, we can write:

$$\nabla_j A_1(\boldsymbol{\theta}) = \mathbb{E}[\phi_j(\boldsymbol{X}_1)] \tag{B.2}$$

$$= \mathbb{P}(\boldsymbol{X}_1(m, n) = 1) \tag{B.3}$$

$$= \frac{\exp \theta_{m,n}}{\sum_{n'=1}^{N} \exp \theta_{m,n'}}, \tag{B.4}$$

and similarly:

$$\nabla_j A_2(\boldsymbol{\theta}) = \frac{\exp \theta_{m,n}}{\sum_{m'=1}^{N} \exp \theta_{m',n}}. \tag{B.5}$$

Therefore by caching the normalizations, it is possible to compute all the gradient in time $O(N^2)$. □

Proposition 14. *For perfect bipartite matchings in HBM, computing one super-partition functions $A_i(\boldsymbol{\theta})$ takes time $O(N^3)$.*

Proof. As described earlier, at a high level, the technique we use to compute $\nabla_j A_i(\boldsymbol{\theta})$ involves constructing an *auxiliary exponential family* with associated graphical model given by a chain of length N, and where the state space of each node in this graph is $\{1, 2, \ldots, N\}$. The basic sufficient statistic coordinates are encoded as node potentials, and the augmented ones, as edge potentials in the chain.

To make this precise, let us introduce some notation. Let $\boldsymbol{\tau}$, $B(\boldsymbol{\tau})$ and $\boldsymbol{\varphi}(y)$ denote the parameters, log-partition function and sufficient statistics of the auxiliary exponential family, $y = (y_1, \ldots, y_N), y_n \in \{1, \ldots, N\}$. We construct the sufficient statistic vectors such that that they have the same dimensionality as $\boldsymbol{\phi}$. The coordinates of $\boldsymbol{\varphi}$ correspond naturally to those of $\boldsymbol{\phi}$: if $\phi_j(x) = x_{n,m}$, then $\boldsymbol{\varphi}_j(y) = \mathbf{1}[y_n = m]$, and if $\phi_j(x) = x_{n,m}x_{n+1,m+1}$, then $\boldsymbol{\varphi}_j(y) = \mathbf{1}[y_n = m]\mathbf{1}[y_{n+1} = m+1]$. With this construction and by setting $\boldsymbol{\tau} = \boldsymbol{\theta}$, we have $\nabla_j A_i(\theta) = \nabla_j B(\boldsymbol{\tau})$. This computation can be done with forward-backward on chain of length N and state space of size N, hence a total running time of $O(N^3)$. □

B.4 Multiple sequence alignment factorization

We start by defining formally the state space, sufficient statistic and the measure factors involved. The state space is the collection of all pairwise alignment indicators, and we use the notation $x^{k,k'}_{m,n}$ to denote the indicator function on the alignment between character m of sequence k and character n of sequence k'. In this section, we will assume for simplicity that the sufficient statistic coordinates have the form $\phi_j(x) = x^{k,k'}_{m,n}$, but higher order statistics were added for the experiments of Section 5.3. Handling those is no more complicated than what was demonstrated for matchings in Section 5.4.6.1.

There are two types of factors in the measure decomposition:

Monotonicity: each pair of components k, k' forms a pairwise alignment:

$$\nu_i(x) = \prod_{m=1}^{N_k} \prod_{m'=1}^{N_{k'}} \mathbf{1}\left[x^{k,k'}_{m,n} = 1, x^{k,k'}_{m',n'} = 1 \Longrightarrow (m > m', n > n') \text{ or } (m < m', n < n')\right].$$

Transitivity: for each triplet of sequences k, k', k'' and positions m, n, p, transitivity holds:

$$\nu_i(x) = \mathbf{1}\left[x^{k,k'}_{m,n} = 1, x^{k',k''}_{n,p} = 1 \Longrightarrow x^{k,k''}_{m,p} = 1\right].$$

Proposition 15. *Each monotonicity factor can be computed in time $O(N_k N_{k'})$, where $N_k, N_{k'}$ are the lengths of the sequences involved.*

Proof. The idea is to use a non-homogeneous pair HMM, or weighted transducer [63]. In

the pair HMM terminology, weights of matching two symbols in this transducer are given by $\exp \theta^{k,k'}_{m,n}$, while the weights of deletions and insertions are set to one.[2]

Once the weighted transducer is constructed, there are standard polynomial-time algorithms for finding its partition function and natural parameter gradient [63]. □

Proposition 16. *Each transitivity factor can be computed in constant time, and the super-partition functions take the form:*

$$A_i(\boldsymbol{\theta}) = 1 + \exp\left(\theta^{k,k'}_{m,n} + \theta^{k',k''}_{n,p} + \theta^{k,k''}_{m,p}\right) + \exp\left(\theta^{k,k'}_{m,n}\right) + \exp\left(\theta^{k',k''}_{n,p}\right) + \exp\left(\theta^{k,k''}_{m,p}\right)$$

Proof. The eight possible cases to consider are shown in Figure 5.3, and the ones in the support of the factor are boxed. They each correspond to a term in the sum above by inspection. □

B.5 Linearization of partial orders factorization

Proposition 17. *The partition function and gradient of the factors proposed in Section 5.4.6.2 can be computed in time $O(N^3)$.*

Proof. Let $G_i = (V, E_i)$ be the current forest in the factorization. We now introduce a new auxiliary family: let $\boldsymbol{\tau}$, $B(\boldsymbol{\tau})$ and $\boldsymbol{\varphi}(y)$ denote its parameters, log-partition function and sufficient statistics of the auxiliary exponential family, $y = (y_1, \ldots, y_N), y_n \in \{1, \ldots, N\}$. We construct the sufficient statistic vectors in the same way as in the proof of Proposition 14. The base measure μ of the auxiliary family enforces the portions of $\leq_p$ that are in E_i. Formally, it is defined as:

$$\mu(y) = \prod_{(n \to n') \in E_i} \mathbf{1}[y_n < y_{n'}].$$

This computation can be done with the sum product algorithm on a forest of size N and state space of size N, hence a total running time of $O(N^3)$. □

[2] Special costs for deletion and insertion are encoded in the matching costs as a ratio, and long gap/hydrophobic core modeling are encoded by augmenting the state of the transducers.

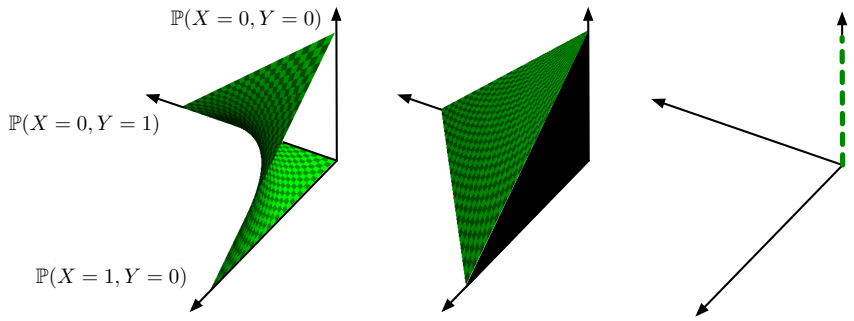


Figure B.2. Left: the mean field realizable moments surface $\mathscr{M}_{\text{MF}}$, center, the realizable moments volume $\mathscr{M}$ without a structured base measure, right, the realizable moment line $\mathscr{M}$ with a structured base measure (in green, parallel to the z-axis).

B.6 MFMF does not guarantee a log partition lower bound

In contrast to what one would expect with a mean field algorithm, MFMF is not guaranteed to lower bound the log partition function. In this section, we show why the argument used in [95] to prove the bound in the case of standard mean field does not apply to MFMF, and then show a simple counter example. As one would expect, the difference comes from the structured base measure.

We first review the argument of [95], Section 5.4, specializing it to our situation, where the graphical model it described in Section 5.4.1, and the tractable subgraph is the fully disconnected graphical model on $S_1, S_2, \ldots, S_J, B_1, B_2, \ldots, B_I$ (the *naive mean field*). We let $\mathscr{M} = \nabla A(\mathbb{R}^J \times \boldsymbol{\mathcal{X}} \times \cdots \times \boldsymbol{\mathcal{X}})$ denote the set of *realizable moments*. We will also use the following definition:

Definition 18. *For an extended real-valued function f, the Legendre-Fenchel transformation is defined as:*

$$f^*(x) = \sup\{\langle x, y\rangle - f(y) : y \in \operatorname{dom}(f)\}.$$

When f is convex and lower semi-continuous, $f = f^{**}$, we can use convexity of A to obtain:

$$A(\boldsymbol{\theta}) = \sup\{\langle \boldsymbol{\theta}, \boldsymbol{\mu}\rangle - A^*(\boldsymbol{\mu}) : \boldsymbol{\mu} \in \mathscr{M}\}. \tag{B.6}$$

Formulation (B.6) is no more tractable than the definition of A, but gives a constrained optimization problem that can be relaxed. Mean field methods can be seen as a particular type of relaxation where the sup is taken over the set of realizable moment induced by a simpler exponential family. In the case of naive mean field on our graphical model, the simpler family is defined as

$$\text{NMF} = \left\{ p_{\boldsymbol{\gamma}}(s_1, \ldots, s_J, b_1, \ldots, b_I) = \exp\left(\sum_i \sum_{x \in \boldsymbol{\mathcal{X}}} \mathbf{1}[b_i = x] \gamma_{i,x} + \sum_j s_j \gamma_j \right) : \gamma_{i,x}, \gamma_j \in \mathbb{R} \right\},$$

from which we define

$$\mathscr{M}_{\text{MF}} = \left\{ \boldsymbol{\mu} \in \mathbb{R}^J : \exists p \in \text{NMF with } \boldsymbol{\mu} = \mathbb{E}[\boldsymbol{\phi}(\boldsymbol{X})], \boldsymbol{X} \sim p \right\}.$$

With this notation, the mean field objective function is:

$$A_{\text{MF}}(\boldsymbol{\theta}) = \sup\{ \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle - A^*(\boldsymbol{\mu}) : \boldsymbol{\mu} \in \mathscr{M}_{\text{MF}} \}.$$

Without a structured base measure (meaning, when the base measure is the uniform counting measure over the full state space), we have $\mathscr{M}_{\text{MF}} \subseteq \mathscr{M}$, and it follows that the mean field estimate is a lower bound. On the other hand, since the edge potentials are deterministic, this inclusion does not hold in our case.

To see why, we show a simple counter-example in Figure B.2: a graphical model on a pair of binary random variables, X, Y. One can check easily that if an indicator edge potential $\mathbf{1}[X = Y]$ is added, then $\mathscr{M}$ is neither included nor enclosing $\mathscr{M}_{\text{MF}}$.

B.7 Handling extended real parameters

Note that in order to handle the cases where a canonical parameter coordinate is $+\infty$, we need to slightly redefine the super-partition functions as follows:

$$A_i(\theta) = \sum_{x \in \mathcal{C}} \exp\left\{ \sum_{j=1}^{J} \mathbf{1}[\theta_j < +\infty] \theta_j \phi_j(x) \right\} \nu_i(x) \prod_{j=1}^{J} \mathbf{1}[\theta_j = +\infty \Rightarrow \phi_j(x) = 1].$$

We also use the convention $(\pm\infty) - (\pm\infty) = \pm\infty$.

B.8 Matching experiments

The generative model used in the third experiment works as follows: first, generate a bipartite perfect matching according to the exponential family HBM $M \sim \text{HBM}(\boldsymbol{\theta})$, next, generate a noisy observation for each edge, $Y_{m,n}|M \sim N(\mathbf{1}(e_{m,n} \in M), \sigma^2)$. The observations $Y_{m,n}$ and parameters $\boldsymbol{\theta}, \sigma^2$ are given to the algorithm, but not the value of M, which is reconstructed using the minimum Bayes risk estimator $\min_m \mathbb{E}[l(m, M)|Y]$ over the 0-1 loss l. The coefficients of this objective are approximated using BPMF. One can check that forming the objective function involves computing moments over HBM with parameters θ_j for the higher order sufficient statistic coordinates j, and with parameters $\theta_j + 1/\sigma^2$ for the basic sufficient statistic coordinates j. We then optimized the objective using the Hungarian algorithm [45]. The zero-one loss is computed against the true (generating) matching, and averaged over 100 random noisy generated datasets.

Appendix C

Derivation of the Poisson Sequence Change Process Inference Algorithm

C.1 Computing the insertion weights

In this appendix, we show how to compute the insertion weights ι_v introduced in the column likelihood Equation (6.1).

In order to deal with the infinite branch length of the stem, we will assume that the joint probability of the model conditions on the following event:

$$\mathcal{R} = (X \prec \Omega \Longrightarrow H(\Omega) \neq g).$$

This is a reasonable construction since homology paths going extinct before Ω do not affect the topology and MSA likelihood computations.

We begin by computing $\mathbb{P}(E = e|\mathcal{R})$ for all $e \in \mathscr{E}$. The strategy we take is to get an expression for $\mathbb{P}(E = e, \mathcal{R})$ for all $e \in \mathscr{E}$ up to a normalization constant that does not depend on v. We will compute the normalization afterwards.

If $\Omega \prec e, e'$, then $\mathcal{R} \subseteq (E \in \{e, e'\})$, and so we can use Theorem 8 to get:

$$\begin{aligned}\frac{\mathbb{P}(E = e, \mathcal{R})}{\mathbb{P}(E = e', \mathcal{R})} &= \frac{\mathbb{P}(E = e)}{\mathbb{P}(E = e')} \\ &= \frac{b(e)}{b(e')}.\end{aligned}$$

To handle the infinite length stem, let us divide it into unit length segments $e_0, e_1, e_2, \ldots$ starting from Ω and going backward in time. Now let $e = e_n$ for some n, and let e' be under Ω, i.e. $\Omega \prec e'$. We have:

$$\begin{aligned}\frac{\mathbb{P}(E = e, \mathcal{R})}{\mathbb{P}(E = e', \mathcal{R})} &= \frac{\mathbb{P}(E = e)\mathbb{P}(\mathcal{R}|E = e)}{\mathbb{P}(E = e')} \\ &= \frac{b(e)}{b(e')}\mathbb{P}(\mathcal{R}|E = e).\end{aligned}$$

The factor $\mathbb{P}(\mathcal{R}|E = e_n)$ can be further decomposed as follows:

$$\mathbb{P}(\mathcal{R}|E = e_n) = \beta(1)(\alpha(1))^n,$$

where:

$$\begin{aligned}\alpha(t) &= \mathbb{P}(H(X_t) \neq g|H(X_0) \sim \pi) \\ \beta(t) &= \mathbb{P}(H(X_T) \neq g|H(X_0) \sim \pi),\end{aligned}$$

where $T \sim \text{Uniform}[0, t]$.

The functions $\alpha(t), \beta(t)$ can be evaluated efficiently and analytically using standard Markov chain properties (see Appendix C.2).

Summing over these segments, we get:

$$\begin{aligned}\mathbb{P}(\mathcal{R}) &= \sum_{v \neq \Omega} \mathbb{P}(E = (\text{pa}(v), v), \mathcal{R}) + \sum_{n=0}^{\infty} \mathbb{P}(E = e_n, \mathcal{R}) \\ &= \sum_{v \neq \Omega} b(v) + \frac{\beta}{1 - \alpha},\end{aligned}$$

where $\alpha = \alpha(1)$, and $\beta = \beta(1)$.

Putting it all together, we get:

$$\iota_v = \frac{1}{\mathbb{P}(\mathcal{R})} \begin{cases} \frac{\beta}{1-\alpha} & \text{if } v = \Omega \\ b(\text{pa}(v) \to v) & \text{o.w.} \end{cases}$$

C.2 Computing the survival probabilities

In this section, we show how to compute the functions $\alpha(t)$ and $\beta(t)$ defined and used in the previous section.

We begin with $\alpha(t)$:

$$\begin{aligned}\alpha(t) &= 1 - \sum_{s \neq g} \pi(s) \mathbb{P}(H(X_t) = g | H(X_0) = s) \\ &= 1 - \left(\pi^{\mathrm{T}} \exp(tQ)\right)_K\end{aligned}$$

The second function is obtained by first conditioning on T:

$$\begin{aligned}\beta(t) &= 1 - \sum_{s \neq g} \mathbb{E}\left[\mathbb{P}(H(X_T) = g | H(X_0) = s, T)\right] \\ &= 1 - \frac{1}{t}\left(\pi^{\mathrm{T}} \int_0^t \exp(uQ)\,\mathrm{d}u\right)_K\end{aligned}$$

The matrix-valued integral can be computed analytically by diagonalization: if we let U, D be such that $Q = UDU^{-1}$ with D diagonal, then:

$$\begin{aligned}I_t &= \int_0^t \exp(uQ)\,\mathrm{d}u \\ &= U\left(\int_0^t \exp(uD)\,\mathrm{d}u\right)U^{-1} \\ &= UD'U^{-1},\end{aligned}$$

with D' diagonal where:

$$D'_{i,i} = \begin{cases} \frac{1}{D_{i,i}}(\exp(tD_{i,i}) - 1) & \text{if } D_{i,i} \neq 0 \\ t & \text{o.w.} \end{cases}$$

To summarize:

$$\beta(t) = 1 - \frac{1}{t}\left(\pi^{\mathrm{T}} I_t\right)_K$$

C.3 Computing the modified Felsenstein peeling recursions

In this section, we show how to compute $f_v = \mathbb{P}(\mathcal{C}|\mathcal{E}_v)$ for all $v \in \mathscr{V}$, a function used and described in Section 6.3.1. We will proceed by conditioning on the value of the current homology path H at v, $H(v)$.

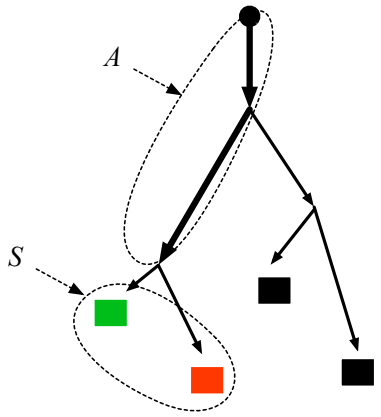


Figure C.1. Edges with nonzero modified felsenstein peeling weight.

First, note that f_v can be zero for some vertices. To see where and why, consider the set of leaves S that that have a non-gap observation in the current column c. Then f_v will be non-zero only for the vertices ancestral to some leaf in S. Let us call this set of vertices A (see Figure C.1).

Next, we use the standard Felsenstein peeling recursion to compute $\tilde{f}_v = \mathbb{P}(\mathcal{C}|\mathcal{E}_v, H(v) \neq g)$. This is done by dynamic programming for $v \in \mathcal{V}, s \in \Sigma_g$ as follows:

$$\tilde{f}_v(s) = \begin{cases} \mathbf{1}(c(v) = s) & \text{if } v \in L \\ \sum_{s' \in \Sigma_g} \exp(b(v)Q)_{s,s'} \prod_{w \in \text{child}(v)} \tilde{f}_w(s') & \text{o.w.} \end{cases}$$
$$\tilde{f}_v = \pi^{\mathrm{T}} \tilde{f}_v.$$

Finally, for $c \neq c_\emptyset$, and using the fact that we always assume conditioning on $\mathcal{R}$ (as described in Appendix C.1, we get:

$$\begin{aligned} f_v &= \mathbb{P}(\mathcal{C}|\mathcal{E}_v) \\ &= \mathbb{E}[\mathbb{P}(\mathcal{C}|\mathcal{E}_v, H(v))] \\ &= \begin{cases} \tilde{f}_v & \text{if } v = \Omega \\ \mathbf{1}[v \in A]\beta(b(v))\tilde{f}_v & \text{o.w.} \end{cases} \end{aligned}$$

For $c = c_\emptyset$, we get:

$$f_v = \begin{cases} \tilde{f}_v & \text{if } v = \Omega \\ 1 + \beta(b(v))(\tilde{f}_v - 1) & \text{o.w.} \end{cases}$$

C.4 Analytic MSA marginalization

In our experiments, we resampled both trees and MSAs, but in other cases, one may want to fix the set of alignments to condition on, for example by using a high recall alignment m (e.g., one produced by the variational framework of Chapter 5). We show in this appendix how to sum over subsets of a high recall alignment (i.e. alignments such that the set of edges is contained in the set of edges of m). This set has exponential size, but we show that an efficient algorithm exists if the subset is defined properly.

Definition 19. *A* clade *is a set of leaves induced by a tree topology as follows: for each vertex $v \in V$, the induced clade f_v is the subset of the leaves that are descendants of v. We call this set of sets* $\text{clades}(\tau)$.

Definition 20. *Let p be a PSCP sample path, with MSA m and observed sequences y. For all observed taxon $v \in L$, an* alignment coordinate *is a pair $a = (v, i)$ where $i \in \{1, \ldots, |y(v)|\}$ is an index in the observed sequence. Note that a multiple sequence alignment can be viewed as an equivalence relation on the alignment coordinates, where two coordinates are related if they are emitted by the same homology path. We will denote these equivalence classes by $[a]_m$.*

Definition 21. *For all set of coordinates S (in particular, for equivalence classes $[a]_m$), we define the* taxon projection *t as the set of taxon represented in the coordinates, $t(S) = \{v \in L : \exists (v, i) \in S\}$.*

We now define a partial order on the space of multiple sequence alignments (note that this is a different partial order than the one described in Section 6.1). The set we sum over will be derived from it.

Definition 22. *If m and n are multiple sequence alignments over the same observations y, we write $m \preceq n$ if the following condition is met: for all coordinate a, there is a clade f such that $t([a]_m) = t([a]_n) \cap f$.*

This can be easily shown to be a partial order. There is a minimal elements: the empty alignment. There is no maximal element however, so the partial order is not a lattice.

We can now formulate the probability we will compute in this section:

$$\mathbb{P}\left(Y = y, M \in (\downarrow m^*)\right),$$
$$\downarrow m = \{n : n \preceq m\}.$$

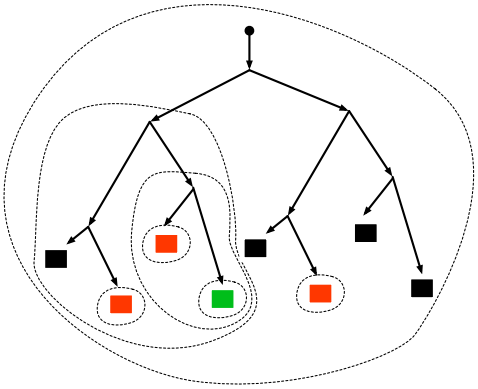


Figure C.2. Example of c-minimal clades.

We will call m^* the *bounding alignment*.

We will need two more definitions:

Definition 23. *Given a column c and a clade f, the induced* survivors *set, is defined as*

$$\text{survivors}(c, f) = \{v \in f : c(v) \neq g\}.$$

Definition 24. *A subset of clades $\mathcal{P} \subset \text{clades}(\tau)$ is* c-minimal *is for all $f \neq f' \in \mathcal{P}$, we have $f' \subset f \implies \text{survivors}(c, f) \neq \text{survivors}(c, f')$. We will denote the largest c-minimal set by $\mathcal{P}_c$ (see Figure C.2).*

Note that for each $f \in \mathcal{P}_c$ with $|f| > 1$, there are unique $f_L, f_R \in \mathcal{P}_c$ such that $f_L \cup f_R = f$. We will also use the notation f_T for the element of $\mathcal{P}_c$ of largest cardinality.

We define the following function on $\mathcal{P}_c$: $Z(f) = Z(c|_f)$, where $c|_f$ denotes restriction of the column to the clade f:

$$\left(c|_f\right)(v) = \begin{cases} c(v) & \text{if } c \in f \\ g & \text{o.w.} \end{cases}$$

The next step is to define two recurrences, r and s, which have the following semantics:

$$s(n) = \sum_{\substack{m \preceq m^* \\ \text{s.t. } |m|=n}} \prod_{c \in m} Z(c)$$
$$r_i(n) = \sum_{\substack{m \preceq \{c_i\} \\ \text{s.t. } |m|=n}} \prod_{c \in m} Z(c).$$

To computing these, first, corresponding to each column c_i in the bounding alignment, we pre-compute following recurrences for $f \in \mathcal{P}_c$:

$$r_i(f, n) = \begin{cases} Z(f) & \text{if } n = 1 \\ \sum_{m=1}^{n-1} r_i(f_L, m) r_i(f_R, n-m) & \text{o.w.} \end{cases}$$
$$r_i(n) = r_i(f_T, n).$$

Next, define a global recurrence collecting the results of the sub-recurrences:

$$s(i, n) = \begin{cases} \mathbf{1}[n = 0] & \text{if } i = 0 \\ \sum_{m=1}^{n} r_i(m) s(i-1, n-m) & \text{if } i \geq 1 \text{ and } n \geq 2 \\ 0 & \text{o.w.} \end{cases}$$
$$s(n) = s(|m^*|, n).$$

Computing all of these recurrences take time $O(|L|^2 \cdot |m^*| + |y| \cdot |m^*| \cdot |L|)$ with practically no hidden constants (in particular, they do not depend on the size of Σ).

Let $q = Z(c_\emptyset)$. The result of these recurrences can be used to compute the probability of interest as follows:

$$\begin{aligned}
\mathbb{P}\left(Y = y, M \in (\downarrow m^*)\right) &= \mathbb{E}\left[\mathbb{P}\left(Y = y, M \in (\downarrow m^*) \middle| |\mathbf{X}|\right)\right] \\
&= \sum_{n=0}^{\infty} \xi_n \sum_{m=0}^{n} q^{n-m} s(m) \\
&= \sum_{n=0}^{|y|} \xi_n \sum_{m=0}^{n} q^{n-m} s(m) + \sum_{n=|y|+1}^{\infty} \xi_n \sum_{m=0}^{|y|} q^{n-m} s(m) \\
&= \sum_{n=0}^{|y|} \xi_n q^n S(n) + \left(\sum_{n=|y|+1}^{\infty} \xi_n q^n \right) S(|y|) \\
&= \sum_{n=0}^{|y|} \xi_n q^n S(n) + q^{|y|+1} \varphi(|y|+1) S(|y|),
\end{aligned}$$

where $|y| = \sum_{v \in L} |y_v|$, and we have used the fact that $s(n) = 0$ for $n > |y|$ and defined $S(M) = \sum_{m=0}^{M} \frac{s(m)}{q^m}$.

C.5 Parameter estimation

In this section, we outline how to compute the gradient of the marginal log likelihood with respect to the parameters $\lambda, \pi_k, Q_{k,k'}$. Note that we did not use these computations in our experiments (we used Bayesian estimators), but we include them for reference. These computations can be used for example to construct maximum likelihood estimators.

The partial derivatives for λ can be derived directly:

$$\begin{aligned}\frac{\partial}{\partial \lambda} \log \mathbb{P}(Y = y, M = m) &= \frac{\partial}{\partial \lambda} \log \left(\varphi(Z(c_\emptyset), |m|) \prod_{c \in m} Z(c) \right) \\ &= \frac{\partial}{\partial \lambda} \log \varphi(Z(c_\emptyset), |m|).\end{aligned}$$

Where, for any $q \in [0, 1], N > 1$:

$$\begin{aligned}\frac{\partial}{\partial \lambda} \varphi(q, N) &= \sum_{n=N}^{\infty} q^{n-N} \frac{\partial}{\partial \lambda} \xi_n(\lambda) \\ &= \sum_{n=N}^{\infty} \frac{q^{n-1-(N+1)} e^{-\lambda} \lambda^{n-1}}{(n-1)!} - \sum_{n=N}^{\infty} \frac{q^{n-N} e^{-\lambda} \lambda^n}{n!} \\ &= \varphi(q, N-1) - \varphi(q, N).\end{aligned}$$

so that we have:

$$\begin{aligned}\frac{\partial}{\partial \lambda} \log \mathbb{P}(Y = y, M = m) &= \frac{\partial}{\partial \lambda} \log \varphi(q, N) \\ &= \frac{\left(\frac{\partial}{\partial \lambda} \varphi(q, N) \right)}{\varphi(q, N)} \\ &= \frac{\varphi(q, N-1)}{\varphi(q, N)} - 1.\end{aligned}$$

For the derivative with respect to the other parameters, we use a different strategy: we use the left-hand side of the following result, which we prove in Appendix C.6:

Proposition 25. *Let $\{\mathbb{P}_\theta : \theta \in \Theta \subseteq \mathbb{R}^n\}$ be an indexed collection of probability measures dominated by μ. For a given observed event E, we let $\pi_\theta(A) = \mathbb{P}_\theta(A \cap E)$ denote the likelihood with density $f_\theta = \frac{\mathrm{d}\pi_\theta}{\mathrm{d}\mu}$, and $\lambda_\theta(A) = \frac{\pi_\theta(A)}{\pi_\theta(E)}$, the posterior probability distribution. If $\ell_\omega(\theta)$ and $L(\theta)$ are both differentiable at $\omega \in \Theta$, then*

$$\nabla \ell_\omega(\omega) = \nabla L(\omega), \pi\text{-a.s.} \tag{C.1}$$

where:

$$\ell_\omega(\theta) = \int \log f_\theta \, \mathrm{d}\lambda_\omega,$$
$$L(\theta) = \pi_\theta(E).$$

In our case, the terms of the expression on the LHS of Equation (C.1) (the expected complete loglikelihood) that depend on π_k and $Q_{k,k'}$ are:

$$\sum_{c \in m} \langle \phi_\tau, \mathbb{E}[N(c)|Y = y, M = m] \rangle + \left(\mathbb{E}[|\mathbf{X}||Y = y, M = m] - |m| \right) \langle \phi_\tau, \mathbb{E}[N(c_\emptyset)|Y = y, M = m] \rangle,$$

where $N(c)$ is the sufficient statistics vector for the CTMC that created column c (i.e. the waiting time for each state and the transition counts), and ϕ_τ is the corresponding parameter vector (i.e. the marginal transition probabilities computed by matrix exponentiation using τ).

How to compute $\mathbb{E}[N(c)|Y = y, M = m]$ is explained in [30], so we only show how to compute $\mathbb{E}[|X||Y = y, M = m]$.

First, note that

$$\mathbb{P}[|\mathbf{X}||Y = y, M = m] \propto \xi_n q^{n-|m|},$$

so the normalization is $\varphi(q, |m|)$, which implies that:

$$\mathbb{E}[N(c)|Y = y, M = m] = \frac{\lambda q e^{\lambda(q-1)} - \sum_{n=0}^{|m|-1} n \xi_n q^n}{q^{|m|} \varphi(q, |m|)}.$$

C.6 Proof of proposition 25

We start by proving the following lemma:

Lemma 26. *If ϕ, ψ are real-valued functions such that:*

1. *$\phi(\boldsymbol{x}_0) = \psi(\boldsymbol{x}_0)$ for some $\boldsymbol{x}_0$,*
2. *$\phi(\boldsymbol{x}) \leq \psi(\boldsymbol{x})$ on an open set S containing $\boldsymbol{x}_0$,*
3. *ϕ and ψ are differentiable at $\boldsymbol{x}_0$,*

then $\nabla\psi(\boldsymbol{x}_0) = \nabla\phi(\boldsymbol{x}_0)$.

Proof. Without loss of generality, ϕ, ψ are univariate functions with $\phi(x_0) = \psi(x_0) = 0$, and $x_0 = 0$.

Let $\delta = \psi'(x_0) - \phi'(x_0)$ and consider a sequence $a_n > 0$ converging to zero with $a_n \in S$. We have:

$$\lim_{n\to\infty} \frac{\psi(a_n) - \phi(a_n)}{a_n} = \delta,$$

and since the numerator and denominator are both positive for all n, we conclude that $\delta \geq 0$.

By doing the same argument with a sequence $b_n < 0$ converging to zero, we get that $\delta \leq 0$, hence the derivatives are equal. □

We can now prove Proposition 25:

Proof. Let $g_\theta = \frac{\mathrm{d}\lambda_\theta}{\mathrm{d}\mu}$ denote the densities for the posteriors, fix $\omega \in \Theta$, and define

$$\begin{aligned}\phi(\theta) &= \ell_\omega(\theta) - \int \log g_\omega \,\mathrm{d}\lambda_\omega \\ \psi(\theta) &= L(\theta).\end{aligned}$$

To apply the lemma, we need to show (1) and (2).

For (1), we have:

$$\begin{aligned}\phi(\omega) &= \int \log \frac{\mathrm{d}\pi_\omega}{\mathrm{d}\lambda_\omega} \,\mathrm{d}\lambda_\omega \\ &= \int \mathbb{P}_\omega(E) \,\mathrm{d}\lambda_\omega \\ &= \mathbb{P}_\omega(E) = L(\omega),\end{aligned}$$

where we have used the a.s. uniqueness of Radon-Nikodym derivative to establish the simplification from the first to second line.

For proving (2), we use Jensen's inequality:

$$\begin{aligned}\phi(\theta) &= \int \log \frac{\mathrm{d}\pi_\theta}{\mathrm{d}\lambda_\omega}\,\mathrm{d}\lambda_\omega \\ &\leq \log \int \frac{\mathrm{d}\pi_\theta}{\mathrm{d}\lambda_\omega}\,\mathrm{d}\lambda_\omega \\ &= \int \log\,\mathrm{d}\pi_\theta = L(\theta).\end{aligned}$$

□

Bibliography

[1] R. Blust. Central and Central-Eastern Malayo-Polynesian. *Oceanic Linguistics*, 32:241–293, 1993.

[2] A. Bouchard-Côté and M. I. Jordan. Optimization of structured mean field objectives. In *Proceedings of Uncertainty in Artifical Intelligence*, 2009.

[3] A. Bouchard-Côté, M. I. Jordan, and D. Klein. Efficient inference in phylogenetic InDel trees. In *Advances in Neural Information Processing Systems 21*, 2009.

[4] M. Bourque. *Arbres de Steiner et réseaux dont certains sommets sont à localisation variable*. PhD thesis, Université de Montréal, 1978.

[5] G. Brightwell and P. Winkler. Counting linear extensions. *Order*, 1991.

[6] W. J. Bruno, N. D. Socci, and A. L. Halpern. Weighted neighbor joining: A likelihood-based approach to distance-based phylogeny reconstruction. *Mol. Biol. Evol.*, 17:189–197, 2000.

[7] D. Burkett, J. Blitzer, and D. Klein. Joint parsing and alignment with weakly synchronized grammars. In *North American Association for Computational Linguistics*, Los Angeles, 2010.

[8] S. F. Chen. Conditional and joint models for grapheme-to-phoneme conversion. In *Proceedings of Eurospeech*, 2003.

[9] A. Culotta, A. McCallum, B. Selman, and A. Sabharwal. Sparse message passing algorithms for weighted maximum satisfiability. In *New England Student Symposium on Artificial Intelligence*, 2007.

[10] C. Daskalakis and S. Roch. Alignment-free phylogenetic reconstruction. In *RECOMB*, 2010.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1977.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39:1–38, 1977.

[13] P. Diaconis, S. Holmes, and R. M. Neal. Analysis of a non-reversible Markov chain sampler. Technical report, Cornell University, 1997.

[14] J. Diamond. *Guns, Germs, and Steel: The Fates of Human Societies*. W.W. Norton, 1999.

[15] C. B. Do, M. S. P. Mahabhashyam, M. Brudno, and S. Batzoglou. PROBCONS: Probabilistic consistency-based multiple sequence alignment. *Genome Research*, 15:330–340, 2005.

[16] M. Dreyer, J. R. Smith, and J. Eisner. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of EMNLP 2008*, 2008.

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

[18] J. Duchi, D. Tarlow, G. Elidan, and D. Koller. Using combinatorial optimization within max-product belief propagation. In *Advances in Neural Information Processing Systems*, 2007.

[19] R. Durrett. *Probability: Theory and Examples*. Duxbury, 2005.

[20] S. N. Evans, D. Ringe, and T. Warnow. Inference of divergence times as a statistical inverse problem. In P. Forster and C. Renfrew, editors, *Phylogenetic Methods and the Prehistory of Languages*. McDonald Institute Monographs, 2004.

[21] J. Felsenstein. Evolutionary trees from dna sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.

[22] J. Felsenstein. PHYLIP - phylogeny inference package (version 3.2). *Cladistics*, 5:164–166, 1989.

[23] S. Goldwater and M. Johnson. Learning OT constraint rankings using a maximum entropy model. *Proceedings of the Workshop on Variation within Optimality Theory*, 2003.

[24] R. D. Gray and Q. D. Atkinson. Language-tree divergence times support the Anatolian theory of Indo-European origin. *Nature*, 426:435–439, Nov 2003.

[25] R. D. Gray and F. M. Jordan. Language trees support the express-train sequence of Austronesian expansion. *Nature*, 2000.

[26] S.J. Greenhill, R. Blust, and R.D. Gray. The Austronesian basic vocabulary database: From bioinformatics to lexomics. *Evolutionary Bioinformatics*, 4:271–283, 2008.

[27] R. C. Griffiths and S. Tavare. Sampling theory for neutral alleles in a varying environment. *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, 1994.

[28] T. Hastie, R. Tibshiranim, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2009.

[29] D. G. Higgins and P. M Sharp. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73:237–244, 1988.

[30] A. Hobolth and J.L. Jensen. Statistical inference in evolutionary models of DNA sequences via the EM algorithm. *Statistical applications in Genetics and Molecular Biology*, 2005.

[31] H. H. Hock. *Principles of Historical Linguistics*. Mouton de Gruyter, 1991.

[32] C. F. Hockett. The quantification of functional load. *Word*, 23:320–339, 1967.

[33] I. Holmes and W. J. Bruno. Evolutionary HMM: a Bayesian approach to multiple alignment. *Bioinformatics*, 17:803–820, 2001.

[34] I. Holmes and G. M. Rubin. An expectation maximization algorithm for training hidden substitution models. *J. Mol. Biol.*, 2002.

[35] B. Huang and T. Jebara. Approximating the permanent with belief propagation. *ArXiv e-prints*, 2009.

[36] Jensen J. and Hein J. Gibbs sampler for statistical multiple alignment. Technical report, Dept of Theor Stat, U Aarhus, 2002.

[37] R. Jakobson. *Why mama and papa*, chapter Why mama and papa?, page 538545. The Hague: Mouton., 1962.

[38] W. Jank. Stochastic variants of the EM algorithm: Monte Carlo, quasi-Monte Carlo and more. *Proc. of the American Statistical Association*, 2005.

[39] Cannone J.J., Subramanian S., Schnare M.N., Collett J.R., D'Souza L.M., Du Y., Feng B., Lin N., Madabusi L.V., Mller K.M., Pande N., Shang Z., Yu N., , and Gutell R.R. The comparative RNA web (CRW) site: An online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BioMed Central Bioinformatics*, 2002.

[40] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, page 85103, 1972.

[41] R. King. Functional load and sound change. *Language*, 43, 1967.

[42] J.F.C. Kingman. *Poisson Processes*. Oxford Studies in Probabilities, 1993.

[43] G. Kondrak. A new algorithm for the alignment of phonetic sequences. In *Proceedings of NAACL 2000*, 2000.

[44] G. Kondrak. *Algorithms for Language Reconstruction*. PhD thesis, University of Toronto, 2002.

[45] Harold W. Kuhn. The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, 1955.

[46] C. Lakner, P. van der Mark, J. P. Huelsenbeck, B. Larget, and F Ronquist. Efficiency of Markov Chain Monte Carlo Tree Proposals in Bayesian Phylogenetics. *Systematic Biology*, 57(1):86–103, 2008.

[47] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10, February 1966.

[48] S. Li, D. K. Pearl, and H. Doss. Phylogenetic tree construction using Markov chain Monte Carlo. *Journal of the American Statistical Association*, 2000.

[49] P. Liang, D. Klein, and M. I. Jordan. Agreement-based learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[50] P. Liang, B. Taskar, and D. Klein. Alignment by agreement. In *North American Association for Computational Linguistics (NAACL)*, pages 104–111, 2006.

[51] B. Litow. The hamiltonian circuit problem and automaton theory. *ACM SIGACT*, 2004.

[52] D. C. Liu, J. Nocedal, and C. Dong. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[53] K. Liu, S. Raghavan, S. Nelesen, C. R. Linder, and T. Warnow. Rapid and Accurate Large-Scale Coestimation of Sequence Alignments and Phylogenetic Trees. *Science*, 324(5934):1561–1564, 2009.

[54] G. Lunter, I. Miklos, A. Drummond, J. Jensen, and J. Hein. Bayesian coestimation of phylogeny and sequence alignment. *BMC Bioinformatics*, 6(1):83, 2005.

[55] G. A Lunter, I. Miklós, Y. S. Song, and J. Hein. An efficient algorithm for statistical multiple alignment on arbitrary phylogenetic trees. *J. Comp. Biol.*, 10:869–889, 2003.

[56] J. Lynch, editor. *Issues in Austronesian*. Pacific Linguistics, 2003.

[57] A. Lyovin. *An Introduction to the Languages of the World*. Oxford University Press, 1997.

[58] A. Martinet. *Économie des Changements Phonétiques*. Bern, 1955.

[59] B. Mau and M.A. Newton. Phylogenetic inference for binary data on dendrograms using Markov chain Monte Carlo. *Journal of Computational and Graphical Statistics*, 1997.

[60] I. Miklos, GA Lunter, and I. Holmes. A long indel model for evolutionary sequence alignment. *Mol Biol Evol.*, 2004.

[61] I. Miklós and Z. Toroczkai. An improved model for statistical alignment. In *First Workshop on Algorithms in Bioinformatics*, Berlin, Heidelberg, 2001. Springer-Verlag.

[62] I. Mikls, A. Drummond, G. Lunter, and J. Hein. *Algorithms in Bioinformatics*, chapter Bayesian Phylogenetic Inference under a Statistical Insertion-Deletion Model. Springer, 2003.

[63] M. Mohri. *Handbook of Weighted Automata*, chapter 6, pages 213–254. Monographs in Theoretical Computer Science. Springer, 2009.

[64] B.M.E. Moret, L. Nakhleh, T. Warnow, C.R. Linder, A. Tholse, A. Padolina, J. Sun, and R. Timme. Phylogenetic networks: modeling, reconstructibility, and accuracy. *IEEE/ACM Transactions on Computational Biology and Biocomputing*, 2004.

[65] L. Nakhleh, D. Ringe, and T. Warnow. Perfect phylogenetic networks: A new methodology for reconstructing the evolutionary history of natural languages. *Language*, 81:382–420, 2005.

[66] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. of Mol. Biol.*, 48:443–453, 1970.

[67] S. Nelesen, K. Liu, D. Zhao, C. R. Linder, and T. Warnow. The effect of the guide tree on multiple sequence alignments and subsequent phylogenetic analyses. *Pac Symp Biocomput*, pages 25–36, 2008.

[68] J. Nichols. *Archaeology and Language: Correlating archaeological and linguistic hypotheses*, chapter The Eurasian Spread Zone and the Indo-European Dispersal. Routledge, 1999.

[69] B. Nothofer. *The reconstruction of Proto-Malayo-Javanic*. M. Nijhoff, 1975.

[70] M.P. Oakes. Computer estimation of vocabulary in a protolanguage from word lists in four daughter languages. *Journal of Quantitative Linguistics*, 7(3):233–244, 2000.

[71] D J. Pearce and P H. J. Kelly. A dynamic topological sort algorithm for directed acyclic graphs. *J. Exp. Algorithmics*, 11:2006, 2006.

[72] C. Peterson and J. R. Anderson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019, 1987.

[73] L. E. Rasmussen. Approximating the permanent: A simple approach. *Random Structures and Algorithms*, 1992.

[74] B. Redelings and M. Suchard. Joint bayesian estimation of alignment and phylogeny. *Syst. Biol*, 2005.

[75] D. Ringe, T. Warnow, and A. Taylor. Indo-European and computational cladistics. *Transactions of the Philological Society*, 100:59–129, 2002.

[76] E. Rivas. Evolutionary models for insertions and deletions in a probabilistic modeling framework. *BMC Bioinformatics*, 6(1):63, 2005.

[77] C. P. Robert. *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. Springer, 2001.

[78] M. Ross, A. Pawley, and M. Osmond. *The lexicon of Proto Oceanic: The culture and environment of ancestral Oceanic society*. Pacific Linguistics, 1998.

[79] Guindon S. and Gascuel O. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic Biology*, 2004.

[80] A. Schwartz and L. Pachter. Multiple alignment by sequence annealing. *Bioinformatics*, 23:e24–e29, 2006.

[81] M. P. Schtzenberger. On a theorem of R. Jungen. *Proceedings of the American Mathematical Society*, 1962.

[82] D. A. Smith and J. Eisner. Dependency parsing by belief propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 145–156, Honolulu, October 2008.

[83] D. Surendran and P. Niyogi. *Competing Models of Linguistic Change. Evolution and beyond*, chapter Quantifying the functional load of phonemic oppositions, distinctive features, and suprasegmentals. Amsterdam/Philadelphia: Benjamins, 2006.

[84] B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-margin parsing. In *EMNLP*, 2004.

[85] B. Taskar, S. Lacoste-Julien, and D. Klein. A discriminative matching approach to word alignment. In *EMNLP 2005*, 2005.

[86] J. Thompson, F. Plewniak, and O. Poch. BAliBASE: A benchmark alignments database for the evaluation of multiple sequence alignment programs. *Bioinformatics*, 15:87–88, 1999.

[87] J. L. Thorne, H. Kishino, and J. Felsenstein. An evolutionary model for maximum likelihood alignment of DNA sequences. *Journal of Molecular Evolution*, 33:114–124, 1991.

[88] J. L. Thorne, H. Kishino, and J. Felsenstein. Inching toward reality: an improved likelihood model of sequence evolution. *J. of Mol. Evol.*, 34:3–16, 1992.

[89] L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1728, 1994.

[90] Edward J V. Siberian link with Na-Dene Languages. *Anthropological Papers of the University of Alaska*, 2010.

[91] L. G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 1979.

[92] A. Varadarajan, R.K. Bradley, and I.H. Holmes. Tools for simulating evolution of aligned genomic regions with integrated parameter estimation. *Genome Biology*, 9:R147, 2008.

[93] M. Ventris and J. Chadwick. *Documents in Mycenaean Greek*. Cambridge University Press, 1973.

[94] M. J. Wainwright and M. I. Jordan. Variational inference in graphical models: The view from the marginal polytope. In *Forty-first Annual Allerton Conference on Communication, Control, and Computing*, 2003.

[95] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1:1–305, 2008.

[96] Y. Watanabe and M. Chertkov. Belief propagation and loop calculus for the permanent of a non-negative matrix. *J. Phys. A: Math. Theor.*, 2010.

[97] C. Wilson. Learning phonology with substantive bias: An experimental and computational study of velar palatalization. *Cognitive Science*, 30.5:945–982, 2006.

[98] D. Wilson. Mixing times of lozenge tiling and card shuffling Markov chains. *The Annals of Applied Probability*, 14:274–325, 2004.

[99] K. M. Wong, M. A. Suchard, and J. P. Huelsenbeck. Alignment uncertainty and genomic analysis. *Science*, 319(5862):473–6, 2008.

[100] Z. Yang and B. Rannala. Bayesian phylogenetic inference using DNA sequences: A Markov Chain Monte Carlo method. *Molecular Biology and Evolution 14*, 1997.

[101] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems*, pages 689–695, Cambridge, MA, 2001. MIT Press.

Index