Inference of population history and patterns from molecular data

Paula Tataru

PhD Dissertation

Bioinformatics Research Centre
Department of Computer Science
Aarhus University
Denmark
Inference of population history and patterns from molecular data

A dissertation presented to the Faculty of Science and Technology of Aarhus University in partial fulfilment of the requirements for the PhD degree

by

Paula Tataru
October 31, 2014
Abstract

The progress achieved by sequencing technologies has revolutionized data collection, leading to an exponential increase in the genomic sequences available. The large amount of data places the field of bioinformatics in a unique position, where novel biological questions can be proposed. Accordingly, the existing mathematical models and computational methods need to be reformulated. I address this from an inference perspective in two areas of bioinformatics.

Population genetics studies the influence exerted by various factors on the dynamics of a population’s genetic variation. These factors cover evolutionary forces, such as mutation and selection, but also changes in population size. The aim in population genetics is to untangle the history of a population from observed genetic variation. This subject is dominated by two dual models, the Wright-Fisher and coalescent. I first introduce a new approximation to the Wright-Fisher model, which I show to accurately infer split times between populations. This approximation can potentially be applied for inference of mutation rates and selection coefficients. I then illustrate how the coalescent process is the natural framework for detecting traces of common ancestry. Lastly, I discuss and extend efficient methods for calculating expectations of certain summary statistics of unobserved data, which are required by some of the population genetics analysis.

The identification of the intricate patterns resulting from biological processes can often shed light on these mechanisms. I address two independent problems of pattern inference within bioinformatics. The first one is the occurrence of patterns described by regular expressions in observed or hidden sequences. I present how to detect statistically significant patterns in a list of ranked sequences, such as RNA sequences ranked after expression level. I then show how standard algorithms can be improved by including pattern occurrence in the hidden structure of observed sequences. Such a hidden structure could be the localization and composition of genes within a DNA sequence. The second problem I target is the computational prediction of the pattern of basepairs resulting in RNA secondary structure. I introduce an evolutionary algorithm to search for a good predictor. Additionally, given a predictor, I present how to improve it using the kinetics of RNA folding coupled with evolutionary information contained within an RNA alignment.
Fremskridt i sekventeringsteknologier har revolutioneret dataindsamling, hvilket har ført til en eksponentiel stigning i disponible genomiske sekvenser. Den store mængde data placerer det bioinformatiske område i en unik position, hvor der kan stilles nye biologiske spørgsmål. Derfor skal de eksisterende matematiske modeller og beregningsmetoder forbedres. Jeg behandler disse fra et inferensperspektiv i to forskellige felter under bioinformatik.


Acknowledgments

It was five years ago that I moved to Denmark to study bioinformatics. Since then, I met a number of wonderful people that played crucial roles in my learning process and development. This is for them.

First and foremost, I would like to thank my supervisors for accepting me as their PhD student, Christian N.S. Pedersen and Asger Hobolth. If it wasn’t for Christian, I might have never applied for the PhD program, while Asger introduced me to interesting research projects. Both of them have provided guidance and help throughout my studies, from which I benefited greatly.

I am grateful to Thomas Mailund for his invaluable help and for the scientific and non-scientific discussions we often had. I would also like to acknowledge Thomas Bataillon for his involvement during the last year of my studies and his interest and choice of offering me a postdoc position.

I would like to thank Yun S. Song for hosting me and advising me while at the University of California, Berkeley. To Jotun Hein, for accepting me, as a student, and inviting me, as an instructor, at his summer school in Oxford.

It was a pleasure to work with all my co-authors. Special thanks go to those who also quickly became good friends, Andreas Sand, James Anderson and Jasmine Nirody. I would also like to thank my colleagues at BiRC and members of Yun’s group, who have made my PhD studies a very enjoyable experience. I am indebted to Ellen Noer for her amazing administrative skills.

My family and friends from Romania supported me and had a large part in making me in who I am today. For that, I thank them.

Last but not least, to Jesper, for being who he is.

Paula Cristina Tataru Bjerg,
Aarhus, October 31, 2014.
## Contents

Abstract i  
Resumé iii  
Acknowledgments v  
Contents vii  
1 Introduction 1  

### I Overview  

2 Discrete Markov models 7  
  2.1 Discrete time Markov chains ...................... 7  
  2.2 Hidden Markov models ................................. 8  

3 Inference of population history 13  
  3.1 The Wright-Fisher model ............................. 16  
  3.2 The Beta with spikes approximation ................ 20  
  3.3 The coalescent process ............................... 22  
  3.4 Inference of identity-by-descent .................... 30  
  3.5 Inference on a tree ................................. 33  
  3.6 Conclusions and future work ......................... 35  

4 Inference of patterns 39  
  4.1 Markov chain embedding and DFAs .................... 40  
  4.2 Patterns in lists of ranked sequences ............... 45  
  4.3 Patterns in the hidden structure of sequences .......... 52  
  4.4 SCFGs and the RNA secondary structure ............. 55  
  4.5 An evolutionary algorithm for SCFGs ................ 60  
  4.6 Oxfold: kinetics and evolutionary information .......... 61  
  4.7 Conclusions and future work ......................... 63  

vii
II Publications and Manuscripts

5 Modeling allele frequency data under a Wright-Fisher model of drift, mutation and selection: the Beta distribution approach 69

6 diCal-IBD: demography-aware inference of identity-by-descent tracts in unrelated individuals 81

7 Comparison of methods for calculating conditional expectations of sufficient statistics for continuous time Markov chains 97

8 Motif discovery in ranked lists of sequences 111

9 Algorithms for hidden Markov models restricted to occurrences of regular expressions 123

10 Evolving stochastic context-free grammars for RNA secondary structure prediction 133

11 Oxfold: kinetic folding of RNA using stochastic context-free grammars and evolutionary information 145

Bibliography 155
Introduction

“The permeation of biology by mathematics is only beginning, but unless the history of science is an inadequate guide, it will continue, and the investigations here summarized represent the beginning of a new branch of applied mathematics.” – J.B.S. Haldane, 1932

With this prediction J.B.S. Haldane ended the mathematical appendix of his 1932 book [77]. Haldane was right. That new branch of mathematics is known today as population genetics. But what he could not have foreseen was that this very theoretical subject would be transformed to one where the development of theory and methodology could not keep up with the torrent of data.

The explosion in the volume of sequence data produced every year is led by the advances in biotechnology and sequencing techniques. The main driver is the reduced cost of sequencing technologies (Fig. 1.1). The Human Genome Project was finalized in 2003 and aimed at determining the first full genome encoded in one single human. It cost $3-billion and took more than ten years to complete. Newer projects, like the 1000 Genomes Project, have much larger objectives, are considerably cheaper and carried out much faster. Today, the goal is to reach the cost of $1,000 per human genome.

The massive amount of available data now puts pressure on model and computational methods development. The data enables us to propose and answer new exciting biological questions. For this, novel mathematical models need to be formulated. Additionally, faster and memory efficient software tools must be developed to be able to handle and analyze quickly the growing volume of data.

Population genetics falls under the wide umbrella of bioinformatics, an interdisciplinary scientific field that combines mathematics and computer science and focuses on processing and studying biological data. Being educated in computer science, I entered the field of bioinformatics with a focus on the computational methods. Much like an evolutionary algorithm, after an apparent random sampling of the field, my interest slowly converged towards population genetics and model development seen from both a mathematical but also a computational perspective. The almost random sampling resulted in this present diverse dissertation. Perhaps this diversity developed also from
1. Introduction

Figure 1.1: Falling fast. In the first few years after the end of the Human Genome Project, the cost of genome sequencing roughly followed Moore’s law, which predicts exponential declines in computing costs. After 2007, sequencing costs dropped precipitously. Reprinted by permission from Macmillan Publishers Ltd: Nature News. E. Hayden. Technology: The $1,000 genome, copyright 2014.

the time I spent at two additional universities. I visited University of Oxford two times, attending the six weeks Oxford Summer School on Computational Biology. There, I started a collaboration that I resumed upon my return two years later. During the second half of my studies, I stayed at University of California, Berkeley for five months.

This dissertation is divided in two parts. The first part consists of three chapters (Chapters 2 to 4) that introduce the research areas I have worked in. These chapters open with a brief history, background and related work, followed by an overview of my contributions. They conclude with suggested directions for future research. The seven publications and manuscripts resulting from these contributions are presented as distinct chapters (Chapters 5 to 11) in the second part of the dissertation.

In the varied collaborations I was involved in, two models prevailed: discrete time Markov chains and, their extension, hidden Markov models. Due to their versatility, I chose to sketch them briefly in Chapter 2. For the initiated reader, this chapter is by no means necessary for the understanding of the dissertation. The remainder of the first part is divided in two chapters, broadly entitled Inference of population history (Chapter 3) and Inference of patterns (Chapter 4).

In Chapter 3, I present my contributions that fall under the area of population genetics. The aim is to study the dynamics of genetic variation within a population and how this is influenced by a series of factors, such as mutation and selection, but also changes in population size. By analyzing the observed
genetic variation, the main scope is to infer the population history. This is achieved by identifying which forces shaped the genetic variation, how populations are related to each other and, at the individual level, how much genetic material individuals share from a common ancestor. My research in this area resulted in two publications and one manuscript [195, 197, 198], reprinted in Chapters 5 to 7.

Biological processes result in data containing intricate patterns. The analysis of these patterns can elucidate the underlying processes. I address the study of patterns in bioinformatics in Chapter 4, where I focus on two problems. The first is detection of patterns, such as words, in sequence data. Statistically significant patterns can pinpoint to important biological functions and they can aid in downstream analysis. My contributions resulted in one publication and one manuscript [154, 196], reprinted in Chapters 8 and 9. The second problem is the prediction of the pattern of basepairs that produces the RNA secondary structure. The function of an RNA molecule is grounded in its secondary structure. Therefore, accurate computational prediction is needed. Emerging from my visits at Oxford University, I have worked with RNA secondary structure prediction, producing two publications [6, 7], reprinted in Chapters 10 and 11.

The seven publications and manuscripts included in this dissertation are given below in chronological order.


Part I

Overview
Discrete Markov models

The multitude of models and software used in bioinformatics, in particular for analyzing molecular data, rely on just a handful of basic stochastic processes. Most of the work I have been involved in during my PhD in one way or another uses a stochastic process that is characterized by the memoryless Markov assumption. Throughout this dissertation, two models are used often: a discrete time Markov chain and one of its extensions, a hidden Markov model. In this chapter, I introduce the terminology and notation for these two models and recapitulate some of the major results and algorithms for them. This brief background information is given here with the purpose of building a more self contained dissertation. Throughout Chapters 3 and 4 I refer back to this chapter.

Markov chains were introduced by Andrey Markov in 1907. A stochastic process \( X(t) \) defined over a set of states \( S \), which, over the time \( t \), moves from one state to another, is Markovian if it is characterized by the memoryless assumption: the next state depends only on the current state and not on the events that preceded it. Most often, as considered here, the Markov processes are finite and time homogeneous, i.e. the set of states is finite, \( S = \{s_1, s_2, \ldots, s_n\} \), and the way the process moves through \( S \) is independent of the time \( t \). In this chapter I consider processes \( X(t) \), or chains, that move through time in discrete steps, \( t \in \{1, 2, \ldots\} \). The transitions between the states are governed by the transition probability matrix, \( P = (p_{ij})_{1 \leq i,j \leq n} \), and the probabilities of the chain starting in either of the states are given by the initial probability distribution, \( \Pi = (\pi_i)_{1 \leq i \leq n} \).

The concept of the process moving through time is appropriate if, for example, we consider the evolution of the nucleotide at a given site in some population, such as the model presented in Section 3.1 on page 16. Sometimes, as in the model presented in Section 4.2 on page 46 or when using hidden Markov models, the concept of time is replaced by that of space.

2.1 Discrete time Markov chains

The theory of discrete time Markov chains is very well developed [73], covering both general chains but also special cases, arising from properties of the tran-
2. Discrete Markov models

sition probability matrix. However, more often than not, approximations are used instead of the classical Markov chain theory, due to the computational cost arising from calculating the powers of a matrix.

A Markov chain can be, for example, absorbing or ergodic. An absorbing chain is one where

• at least one of the states \( s_i \) is absorbing, i.e. \( p_{ii} = 1 \);

• it is possible, from every state, to go to an absorbing state.

Such a chain is given, for example, by a random walk, considered in Section 4.2 on page 46, where I illustrate how both the Markov chain theory but also an asymptotic approximation can be used.

At the other end, an ergodic chain has positive probability of going from every state to every state, though not necessarily in one move. For such chains, there exists a probability vector \( \Phi \), called the stationary distribution, such that the powers \( P^t \) approach a limiting matrix with all rows begin equal to the same vector \( \Phi \). In Section 3.1 on page 16, I introduce the Wright-Fisher model, a Markov chain that can either be absorbing or ergodic. This chain has, typically, a very large state space and its stationary distribution for the ergodic version and absorbing probability for the absorbing version have been calculated using approximations.

For a general Markov chain, the most widely used result is the probability of being in state \( s_j \) after \( t \) steps

\[
\mathbb{P}(X(t+1) = s_j) = \left( \Pi P^t \right)_j.
\]  

(2.1)

2.2 Hidden Markov models

A hidden Markov model (HMM) is a probabilistic model that can be used to describe observable events that depend on hidden states which are not directly observable. Because of their computational and analytic tractability, they are used in a many fields, and their wide application in bioinformatics [217] ranges from modeling of proteins to coalescent theory. In this dissertation I consider HMMs both from an application point of view but also from an algorithmic perspective. In Section 3.3 on page 25 and Section 3.4 on page 30 I use the standard HMM algorithms for two related HMMs. I obtain information about the population that a sample of individuals originated from and detect relationships between the sampled individuals, respectively. In Section 4.3 on page 52 I consider the extension of some of the standard HMM algorithms with the final aim of improving the quality of the prediction, which can potentially be used in the wide range of applications in bioinformatics.

An HMM describes a joint probability distribution over the observed sequence \( y_{1:T} = y_1y_2\ldots y_T \in \mathcal{O}^* \) and a hidden sequence \( x_{1:T} = x_1x_2\ldots x_T \in \mathcal{H}^* \), where \( \mathcal{O} \) and \( \mathcal{H} \) are finite alphabets of observables and hidden states,
2.2. Hidden Markov models

respectively. The hidden sequence is a realization of a discrete time Markov process \( X(t) \). We can formally define an HMM \cite{167} as consisting of

- two finite alphabets of hidden and observable states, \( H = \{ h_1, \ldots, h_N \} \) and \( O = \{ o_1, \ldots, o_M \} \), respectively;
- a probability vector \( \Pi = (\pi_i)_{1 \leq i \leq N} \) where
  \[
  \pi_i = \pi_{h_i} = \mathbb{P}(X(1) = h_i),
  \]
  is the probability of the hidden process starting in \( h_i \);
- a probability matrix \( A = \{a_{ij}\}_{1 \leq i,j \leq N} \) where
  \[
  a_{ij} = a_{h_i,h_j} = \mathbb{P}(X(t) = h_j | X(t-1) = h_i),
  \]
  is the probability of a transition from \( h_i \) to \( h_j \);
- a probability matrix \( B = \{b_{ij}\}_{1 \leq i \leq N, 1 \leq j \leq M} \) where
  \[
  b_{ij} = b_{h_i,o_j} = \mathbb{P}(Y(t) = o_j | X(t) = h_i),
  \]
  is the probability of \( h_i \) emitting \( o_j \).

The standard algorithms for HMMs, summarized below, are used to

- calculate the likelihood of observed sequences;
- for an observed sequence, determine a hidden path \( x_{1:T}^* \), called decoding (or prediction), that explains the observation;
- determine the parameters \( \Pi, A \) and \( B \) that best explain the observed sequences.

The forward algorithm

The forward algorithm \cite{167} finds the likelihood of the data \( y_{1:T} \) by summing the joint probability of the observed and hidden sequences for all possible sequences \( x_{1:T} \)

\[
\mathbb{P}(y_{1:T}) = \sum_{x_{1:T}} \mathbb{P}(y_{1:T}, x_{1:T}),
\]

\[
\mathbb{P}(y_{1:T}, x_{1:T}) = \pi_{x_1} b_{x_1,y_1} \prod_{t=2}^{T} a_{x_{t-1},x_t} b_{x_t,y_t},
\]

where Eq. (2.3) is the multiplication of the probabilities of transitions and emissions, which explain observing \( y_{1:T} \) with \( x_{1:T} \) as the hidden sequence.
The forward algorithm finds Eq. (2.2) by recursively filling up a table, $\alpha$, with values $\alpha_t(h_i) = P(y_1:t, x_t)$, being the probability of observing $y_{1:t}$ and being in state $x_t$ at time $t$. The recursion is given by

$$\alpha_1(h_i) = \pi_i \cdot b_{i,y_1}, \quad \alpha_t(h_i) = \sum_{j=1}^{N} \alpha_{t-1}(h_j) \cdot a_{ji} \cdot b_{i,y_t},$$

and, finally,

$$P(y_{1:T}) = \sum_{i=1}^{N} \alpha_T(h_i).$$

The probability of observing the sequence $y_{1:T}$ can also be found as the sum of the entries in the following vector-matrix product

$$P(y_{1:T}) = \sum_{i=1}^{N} \left( \Pi B_1 A B_2 \ldots A B_T \right)_i,$$

(2.4)

where $B_t$ is a diagonal matrix with the emission probabilities of $y_t$ on the diagonal

$$B_t = \begin{bmatrix} b_{1,y_t} \\ b_{2,y_t} \\ \vdots \\ b_{N,y_t} \end{bmatrix}.$$

If we were to not consider the emission probabilities in Eq. (2.4), corresponding to only running the hidden Markov chain, the product becomes $\Pi A^{T-1}$, and we recover Eq. (2.1) for $T - 1$ steps and $P = A$.

The backward algorithm

The backward algorithm [167] is very similar to the forward algorithm, but instead of moving forwards through the sequence $y_{1:T}$, it moves backwards, recursively filling up a table $\beta$ with values $\beta_t(x_t) = P(y_{t+1:T} \mid x_t)$, the probability of observing $y_{t+1:T}$, given that the state at time $t$ is $x_t$. Then

$$\beta_T(h_i) = 1, \quad \beta_t(h_i) = \sum_{j=1}^{N} a_i \cdot b_{j,y_{t+1}} \cdot \beta_{t+1}(h_j).$$

The backward probabilities stored in $\beta$, together with the forward probabilities stored in $\alpha$, can be used to calculate the posterior probabilities and, consequently, a decoding, as follows.
The posterior decoding algorithm

The posterior decoding algorithm [167] relies on the posterior probabilities of the hidden states conditional on the data, \( \gamma_t(h_i) = P(X(t) = h_i \mid y_{1:T}) \), i.e. the probability of being in state \( h_i \) at time \( t \), conditional on the observed sequence. It finds the sequence of hidden states \( x^*_t \) that maximizes locally the posterior probabilities,

\[
x^*_t = \arg \max_{x_t} \{ \gamma_t(x_t) \}, \quad \text{using} \quad \gamma_t(h_i) = \frac{\alpha_t(h_i) \beta_t(h_i)}{\sum_{j=1}^{N} \alpha_T(h_j)}.
\]

Thus, to compute the posterior decoding, \( \alpha_t(h_i) \) and \( \beta_t(h_i) \) are first filled out for all \( t \) and \( i \) and then the decoding is computed by \( x^*_t = \arg \max_{h_i} \{ \gamma_t(h_i) \} \).

The Viterbi algorithm

The Viterbi algorithm [167] is an alternative to the prediction given by the posterior decoding algorithm. It finds the sequence of hidden states, \( x^*_{1:T} \), that maximizes the joint probability of the observed and hidden sequences given in Eq. (2.3)

\[
x^*_{1:T} = \arg \max_{x_{1:T}} \{ P(y_{1:T}, x_{1:T}) \}, \quad \text{using} \quad \omega_t(x_t) = \max_{x_{1:t-1}} \{ P(y_{1:t}, x_{1:t}) \},
\]

a new table containing the probability of a most likely decoding ending in \( x_t \) at time \( t \), having observed \( y_{1:T} \). This can be obtained recursively as follows

\[
\omega_1(h_i) = \pi_i b_{i,y_1}, \quad \omega_t(h_i) = \max_{1 \leq j \leq N} \{ \omega_{t-1}(h_j) a_{ji} b_{i,y_t} \}.
\]

After computing \( \omega \), a most likely sequence of hidden states is retrieved by backtracking through the table, starting in entry \( \arg \max_{h_i} \{ \omega_T(h_i) \} \).

The Baum-Welch algorithm

For the algorithms presented so far, it is implicitly assumed that all the probability distributions that parameterize the HMM are fully known. In data analysis the model has to be trained first: the probability distributions contained in \( \Pi, A \) and \( B \) have to be inferred, given some set of observed sequences. The Baum-Welch algorithm is an Expectation-Maximization (EM) training algorithm. From a set of starting parameters, the Baum-Welch algorithm finds \( \Pi^*, A^* \) and \( B^* \) such that the likelihood \( P(y_{1:T} \mid \Pi^*, A^*, B^*) \) is a local maximum. This is done in an iterative procedure where the forward, backward and posterior probabilities, given the current parameters, are calculated and then the parameters are updated, until the likelihood converges.
Inference of population history

Population genetics studies the dynamics of genetic variation within species. Classical models focus on the evolution of the distribution of allele frequencies in populations. Within this area, “evolution” is typically defined as any change in a population’s genetic composition over time. This is caused by random genetic drift and three main evolutionary processes: natural selection, mutation and migration. The evolutionary processes can be thought of and analyzed as acting in an infinite population, leading to deterministic equations for the evolution of allele frequency. Random genetic drift describes the stochastic effects of reproduction in a finite population. Within classical population genetics, a topic of interest is the determination of whether gene frequency changes are just the result of drift, or whether (and which) evolutionary forces were at play. With the emergence of molecular data, coalescent studies drove the field towards analyzing the genealogical history of a sample from one (or more) population(s). Coalescent models aim to infer the history of the population, such as mutation, migration and selection rates and past changes in population size.

The first law of population genetics describes the relation between the allelic and genotypic frequencies in a randomly mating infinite population with no evolutionary forces. The law, known as the Hardy-Weinberg principle [79, 207], states that the allele frequency is preserved from generation to generation. It also provides a way to directly deduce the genotype frequencies from the allele frequencies and vice versa. This consequently permits the great simplification of regarding allele frequencies as the fundamental quantities for evolutionary change.

The theoretical foundations of the field were established in the 1920’s and 1930’s by R.A. Fisher, J.B.S. Haldane and S. Wright [62, 77, 212]. Their achievement was to show that the Darwinian theory of natural selection not only integrates with Mendelism, but that together they describe the evolution of a population. The three theorists developed formal models to explore how the evolutionary forces modify the genetic composition of a Mendelian population over time.

The field was joined in the 1950’s by two newcomers that mainly extended the work of Wright. G. Malécot formulated the concept of identity-by-descent
3. Inference of population history

[134] and brought in the tools of modern probability analysis by introducing the idea of population transformations as a Markov chain [135]. M. Kimura was very creative in using the diffusion approximation and obtained the distribution of allele frequency as a function of time [101]. He also formulated the neutral theory of molecular evolution [103] to explain some then new findings in evolution and variation at the molecular level: most changes at molecular level are neutral with respect to natural selection. Coalescent models rely on the neutral theory he developed. More details on the beginnings and development of population genetics can be found in [40].

Classical population genetics models, among which the Wright-Fisher is most used, are prospective – given a set of values for selection, mutation and migration rates, equations can be solved predicting, for instance, the allele frequency in the next generation. Retrospective coalescent models have been developed at the beginning of 1980’s. The first retrospective view was embedded in Ewens sampling formula [57], which can be easily proved in the coalescent framework. The foundations of coalescent models were laid by J. Kingman [104]. R.R. Hudson [91] further developed the model to include recombination. The coalescent models are natural extensions of classical population genetics models and arise from the diffusion approximation. They are based on the idea that all genes in a population derive from a single common ancestor. This can be regarded as a generalization of Malécot’s identity-by-descent to more than two genes. The essence of the coalescent models is to trace the genealogy of the sampled genes backward in time until the common ancestor.

Even though the Wright-Fisher model was built as a prospective model, it can also be used for retrospective inference. Different (simulation-free) inference problems, as addressed in this chapter, are better suited for one model or the other. For example, inference based on the Wright-Fisher model relies, so far, on a set of independent loci. Free recombination is assumed between them and linkage disequilibrium in the data can interfere with accurate inference. No full solution is known to efficiently incorporate recombination within the Wright-Fisher in a simulation-free inference framework, but some approaches exist [96]. On the other hand, the coalescent model, even though intractable when considering recombination, can be used within a Markovian approximation framework. This allows for modeling of recombination events, which are essential for identification of identical-by-descent (IBD) tracts, and inference of recombination rates.

When considering detection of selection, the Wright-Fisher seems the more appropriate model. The genealogical structure behind the coalescent with selection is very similar to the coalescent with recombination. However, the same Markovian approximation cannot be applied for the coalescent with selection. Therefore, detection of selection, as in [169], is often performed by assuming a neutral coalescent and subsequent identification of regions which depart from the neutral model. Direct use of the coalescent with selection is mathemat-
ically and computationally challenging, even though some theoretical results exist [106, 121, 162].

Before the molecular revolution, the rich stochastic theory of population genetics suffered from the fact that it dealt with allele frequencies, whereas observations were on phenotypes, often complex ones. For simple Mendelian traits, relying on the Hardy-Weinberg principle, the underlying allele frequencies are easily determined from the observed phenotypes. However, complex traits are typically the cumulative result of many Mendelian factors. This limited the applicability of the theory on the then available data. The arrival of molecular data and recent developments in biotechnology have changed this field from one rich in theory but poor on data, to one where the development of theory and methodology are trailing behind the collection of data.

In this chapter I address the issue of developing new methodology to analyze molecular data and infer population history. My work has been pertained to both classical population genetics, namely the Wright-Fisher model, but also the extension of the coalescent theory to include recombination and variable population size, and its application to inferring IBD tracts in individuals. Additionally, I investigated efficient ways of calculating expectations of certain summary statistics for continuous time Markov chains (CTMCs) which, as shown later, can be used in the setting of inferring population history. This work resulted in one manuscript and two publications

- Section 3.2 (Chapter 5): *Modeling allele frequency data under a Wright-Fisher model of drift, mutation and selection: the Beta distribution approach* [197];
- Section 3.4 (Chapter 6): *diCal-IBD: demography-aware inference of identity-by-descent tracts in unrelated individuals* [198];
- Section 3.5 (Chapter 7): *Comparison of methods for calculating conditional expectations of sufficient statistics for continuous time Markov chains* [195].

The remainder of this chapter is structured as follows. I first introduce the Wright-Fisher model and some of its approximations in Section 3.1. This leads to my contribution presented in Chapter 5 and outlined in Section 3.2, on approximating the allele frequency distribution using the beta distribution.

In Section 3.3 I introduce the coalescent, discuss recombination and the Markovian approximation, the sequentially Markov coalescent (SMC) [143]. I also briefly present theoretical and simulation results that I obtained using the SMC. I have, however, discontinued this work due to recent development in the field [169] which, in my opinion, rendered my work superfluous. During my visit at the University of California, Berkeley, I was involved in developing the first coalescent-based IBD detection method by applying an existing SMC-based framework [182]. This work is presented in Chapter 6 and summarized in Section 3.4.
3. **Inference of population history**

I discuss briefly in Section 3.5 how the amount of available data requires joint models of population genetics and phylogenetics. Phylogenetic models are used to study the evolutionary relationships between species or populations, which is modeled as a tree. Typically they rely on CTMCs to describe the evolution of sequences on the tree. Within the framework of CTMCs, efficient calculation of certain expectations of summary statistics or matrix exponentials are often required, which leads to my contribution summarized in the same Section 3.5 and presented in Chapter 7.

I conclude the chapter with Section 3.6, where I draw the main conclusions and present potential future work.

3.1 **The Wright-Fisher model**

The mathematical model resulting from the work of R.A. Fisher and S. Wright is called the Wright-Fisher model and describes how a locus in a population evolves in time. Here, I consider only bi-allelic loci. The model assumes that the generations are discrete and non-overlapping. This simplifying assumption is not very realistic, but the model proved nevertheless very useful in practice. Additionally, the model assumes haploid individuals, but it can efficiently be used for diploid populations of $N$ individuals by considering a Wright-Fisher model with $2N$ individuals. The theory presented in this section is discussed in more details in [58, 199].

In the simplest form, the population size is constant (of size $2N$) and it undergoes random mating. Under pure drift, a new generation is created from the current one by random sampling with replacement: each individual and its corresponding allele is chosen, with equal probability, to be copied in the next generation. Let $A$ and $a$ be the two alleles at the locus. The Wright-Fisher model follows the count $Z_t$ of one of the alleles, $A$, as a function of the generation $t$. This is in fact a random variable which, due to the random sampling that governs the model, follows a binomial distribution.

Let us define a new random variable $X_t = Z_t / 2N$, the frequency of the allele $A$, and a function $g(x)$ over allele frequencies, with $0 \leq x \leq 1$ and $0 \leq g(x) \leq 1$. I call this function here the evolutionary pressure. It encapsulates the deterministic change in the allele frequency resulting from the various evolutionary forces. Under pure drift, the allele frequency does not change and $g(x) = x$. If the count of the allele is $z_t$ at generation $t$, $Z_{t+1}$ is given by

$$Z_{t+1} \mid Z_t = z_t \sim \text{Bin}(2N, g(x_t)) . \tag{3.1}$$

Modeling mutation and/or selection is solely done through changing the definition of the evolutionary pressure $g(x)$. Mutations are included by considering the probabilities per generation of each of the allele mutating to the other, $u : A \rightarrow a$ and $v : a \rightarrow A$. The evolutionary pressure takes the form

$$g(x) = (1 - u) x + v (1 - x) = (1 - u - v) x + v .$$

16
3.1. The Wright-Fisher model

![Figure 3.1: Example of a Wright-Fisher population.](image)

Generations are evolving down the figure. Lines join individuals in two generations if one is the offspring of the other. The population contains two alleles, A (red) and a (black). (A) Pure drift. (B) Mutation with probabilities $u : A \rightarrow a$ and $v : a \rightarrow A$. (C) Selection (on the A allele, indicated by larger circles) and mutation.

The random sampling happens exactly as before. After all the individuals are chosen for the next generation, they might mutate to the other allele type. Then an allele $A$ is descended from an individual from the previous generation that was $A$ and did not mutate ($(1 - u)x$) or $a$ and mutated $(v(1 - x))$.

To incorporate selection, the random sampling has to be adjusted to indicate that some genotypes (in a diploid model) are more likely to survive or be chosen due to an increased fitness. This changes the previous uniform sampling probability into one that depends on fitness coefficients. If the allele $A$ has selection coefficient $s$, dominance parameter $h$ and has frequency $x$, the three possible genotypes have the following frequencies and relative fitness

<table>
<thead>
<tr>
<th>Genotype</th>
<th>$AA$</th>
<th>$Aa$</th>
<th>$aa$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>$x^2$</td>
<td>$2x(1-x)$</td>
<td>$(1-x)^2$</td>
</tr>
<tr>
<td>Relative fitness</td>
<td>$1+s$</td>
<td>$1+sh$</td>
<td>$1$</td>
</tr>
</tbody>
</table>

The fitness describes how likely it is for that particular genotype to be transmitted to the next generation, relative to the other genotypes. Then the probability of each genotype to be part of the next generation is given by the product of the fitness and the frequency of the genotype. This frequency is obtained from the alleles frequencies using the Hardy-Weinberg principle, resulting from random mating. From this, we can calculate the probability of picking an allele $A$ in the next generation by choosing either the $AA$ ($(1+s)x^2$) or $Aa$ ($(1+sh)x(1-x)$) genotype. The evolutionary pressure function becomes

$$g(x) = \frac{(1+s)x^2 + (1+sh)x(1-x)}{(1+s)x^2 + 2(1+sh)x(1-x) + (1-x)^2}$$

$$= x + \frac{s x(1-x) [h + (1-2h)x]}{(1+s)x^2 + 2(1+sh)x(1-x) + (1-x)^2},$$

where the denominator is a normalizing factor. We normally assume that, except in extreme cases, the fitness coefficients $s$ and $sh$ are small. This allows
3. Inference of population history

for a reformulation of the evolutionary pressure [58] by ignoring small-order terms in the above,

\[ g(x) = x + s x (1 - x) [h + (1 - 2h)x ] . \]

The addition of mutations on top of selection follows as previously described. If we let \( S(x) \) be the above defined function, containing the selection dependent evolutionary pressure, the total evolutionary pressure can be described as

\[ g(x) = (1 - u - v) S(x) + v . \]

Figure 3.1 shows a Wright-Fisher population with \( 2N = 8 \) individuals evolving for 6 generations under pure drift (A), mutation (B) and selection with mutation (C).

Approximations to the Wright-Fisher model

For inference purposes, given a Wright-Fisher model, the main interest lies in calculating the distribution of \( Z_t \), or more commonly, \( X_t \), as a function of the generation \( t \), conditional on a starting frequency \( x_0 \) in the initial population. Even though the mathematics involved in setting-up the model are very simple and the model simplifies greatly the behavior of natural populations, no analytic closed form result is known for this distribution.

A simple and straightforward solution is given by the Markov chain theory (Section 2.1 on page 7). The random variable \( Z_t \) is a discrete time Markov chain on the state space \( \{0, 1, \ldots, 2N\} \), with transition probabilities given by Eq. (3.1). The allele frequency distribution is given by the \( t^{th} \) power of the transition probability matrix. However, the population size in practical applications is typically very large and this approach is therefore not computationally feasible. I note here that the nature of the Markov chain induced by the Wright-Fisher is determined by the presence or absence of mutation (and/or migration). If mutation is absent, the chain is absorbing and will eventually reach 0 or 1. If mutation is present, the process is ergodic and can always move away from 0 and 1 through mutations.

For inference purposes, further simplifications need to be made to the Wright-Fisher model. Typically two different classes of approximations are used. They both assume that the population size \( 2N \) is large enough that the frequency \( X_t \) can effectively be treated as a continuous variable. The first one, which both Wright [212] and Fisher [62] used, but with which Kimura [102] obtained more general results than his predecessors, is the diffusion limit. The second type of approximation relies on moments and was first used by Balding and Nichols [13].
The diffusion limit to the Wright-Fisher

Diffusion models are based on the idea that variables disperse (or diffuse) away from their original location or state in a Markovian way. In diffusion analysis, the random variable and time are both treated as continuous variables, yet diffusion models are often used to approximate stochastic models with discrete variables and discrete time steps. The key lies in choosing an appropriate transformation of the original model.

The main idea behind the diffusion limit to discrete time Markov models, including the Wright-Fisher, is to scale the states and time by the size of the state space \( N \), i.e. \( 2N \) for the Wright-Fisher, obtaining \( X_{\tau} = Z_{t}/2N \), with \( \tau = t/2N \). The re-scaled process is essentially identical to the original process, but as the size of the state space tends to get larger and larger, the process converges to a continuous-time and continuous-space diffusion process. A diffusion process is characterized by two quantities, the mean \( a(x) \) and the variance \( b(x) \) of the infinitesimal displacement. Various quantities of interest, such as the stationary distribution of the ergodic process or fixation probabilities and absorption times when the chain is absorbing, are found using the forward and backward Kolmogorov equations [58]. Sometimes, explicit solutions to these equations can be found, giving the full time-dependent distribution of the random variable described by the Markov chain.

For the Wright-Fisher model, under selection and mutation, the infinitesimal mean and variance are given by [58]

\[
\begin{align*}
a(x) &= 2Ns x (1 - x) [h + (1 - 2h) x] - 2Nu x + 2Nv (1 - x), \\
b(x) &= x (1 - x).
\end{align*}
\]

The other cases, such as pure drift or only mutation, can be obtained from the above by setting the right parameters to 0. There is no known closed form solution for the diffusion process with selection and mutation. Kimura [101] was the first to solve the diffusion equation for pure drift, while in [102], he discusses various results for more complicated scenarios. Most of the known results were obtained by Kimura and little has been done in that regard since then. Some new approaches exist to efficiently calculate numerical approximations to the diffusion [221] or the spectral representation of the allele frequency density function [187].

I note here that the diffusion limit to the Wright-Fisher requires that the various parameters involved in the evolutionary pressure, given by \( u, v, s \) and \( sh \), are all in the order of \( 1/N \), such that the resulting scaled parameters, \( 2Nu, 2Nv, 2Ns \) and \( 2Nsh \), are in the order of 1. This is the source of the common practice of simplifying expressions by throwing out “small” terms like \( u^2 \) or \( u/(2N) \) [206]. It also indicates that in the diffusion limit, a re-scaling of the parameters and time by a constant factor will not affect the allele frequency distribution. This result is responsible for the notion that it is impossible to estimate, for example, the mutation rate and population
3. Inference of population history

size separately. However, while it may be be true that there is low power in doing so, this is simply a consequence of the assumptions of the model. These might be expected to break down in cases in which the diffusion is not appropriate [206].

Moment-based approximations

The main idea behind the moment-based approximations is to use mathematically convenient distributions and fit those to the true allele frequency distribution by relying, typically, on the first two moments of the distributions. These approximation are not principled and are less ambitious than the diffusion. They do not aim at accurately describing the whole true distribution, but provide easy mathematical formulas to work with. They have been proved, nevertheless, useful in inference problems [13, 38, 152, 160, 186].

For the bi-allelic Wright-Fisher, the two distributions used so far are the normal and beta distributions. For the multi-allelic case, their extensions, multinomial and Dirichlet distributions are typically used. The rationale for the two different distributions came form the diffusion approximation. The normal distribution is used in terms of a transient model, being the result of the diffusion when drift is small [160]. On the other hand, the beta distribution was used in terms of an equilibrium model [12]. It has been shown, using the diffusion limit, that the stationary distribution under mutation and/or migration is a beta distribution [102].

The normal and beta distributions are similar when $x$ is close to 0.5, but qualitatively different otherwise. The beta distribution is defined on $[0, 1]$, while the normal distribution has to be truncated by replacing the densities in the intervals $(-\infty, 0]$ and $[1, \infty)$ with atoms at 0 and 1, respectively [152]. The differences between the two distributions have been used as a way to distinguish populations that are either in a transient phase or in equilibrium [12].

3.2 The Beta with spikes approximation

The beta distribution misfits the true distribution at the boundaries (Fig. 5.1 on page 71). Due to its continuous nature, the probabilities of the allele frequency being either 0 or 1 will always be 0. However, under the Wright-Fisher, positive probabilities can accumulate at the boundaries.

So far, all moment-based approximations exclude selection. I address the issue of adjusting the beta distribution to more accurately model the boundaries and extend it to include selection in

3.2. The Beta with spikes approximation

Here, I will present the main idea behind Chapter 5 and summarize the results. In this work, I contributed to all stages, from the development of the methodology, designing and implementing the code, performing experiments and writing the manuscript. It is still in progress and I will resume it upon the completion of my PhD.

Approach

The beta distribution can easily be fitted when only mutation and/or migration are at play. This is a consequence of the linearity in $x$ of $g(x)$, which gives analytic expressions for the mean $E[X_t]$ and variance $\text{Var}(X_t)$ [185]. To include selection, we need to rely on a recursive approach to calculate the two moments. The laws of total expectation and variance provide a way to do so and they rely on $E[g(X_t)]$ and $E[g(X_t)^2]$ (Eq. (5.1) on page 72). These two quantities, as indicated in Eq. (5.3) on page 72, can be calculated by relying on the assumption that $X_t$ follows a beta distribution.

The issue of the probabilities at the boundaries can be addressed by adjusting the beta distribution with the two missing probabilities. The resulting distribution, which we called the beta with spikes, requires two more quantities: the loss $P(X_t = 0)$ and fixation $P(X_t = 1)$ probabilities. These two can be calculated in a recursive manner using the law of total probability (Eq. (5.6) on page 73). When the two spikes are considered, they need to be included in the calculation of $E[g(X_t)]$ and $E[g(X_t)^2]$, as indicated in Eq. (5.5) on page 73.

The two recursive algorithms are summarized in Table 5.1 on page 73.

Results

As preliminary results, we obtained that the beta with spikes matches the true distribution much better than the beta distribution. We investigated the fit under pure drift, three different values of mutation rates and three different values of selection coefficients. Figure 5.3 on page 76 shows the quality of the approximations as a function of the initial frequency and the time, while Fig. 5.4 on page 77 shows the full true and approximated distributions for a few scenarios.

We also inspected briefly how parameter scaling effects the resulting distribution, as discussed previously on page 19. We calculated the loss probability in two equivalent (up to scaling) populations of size $2N = 200$ and $2N = 400$, under pure drift, mutation and selection. As shown in Fig. 5.5 on page 78, both the Wright-Fisher and beta with spikes produce loss probabilities that are not equal in the two equivalent populations.

To investigate the beta with spikes from an inference perspective, we reproduced experiments from [67], where a new software, Kim Tree, is introduced. Kim Tree uses Kimura’s diffusion solution to pure drift and estimates split
3. Inference of population history

times between populations. In our experiments, we obtained that the beta with spikes was able to accurately infer the split times (Fig. 5.6 on page 79). It showed an average performance close to Kim Tree, but with some subtle differences between the estimations of the different split times considered, as discussed in Section 5.3 on page 78.

3.3 The coalescent process

The coalescent process, introduced by Kingman [104], is dual to the Wright-Fisher model and relies on the same assumptions. Instead of following the evolution of the allele frequency forward in time, the coalescent process traces the genealogical history of a sample from the population backward in time. The basic idea behind the coalescent is that the sampled individuals can be viewed as randomly picking their parents. When two individuals pick the same parent, they coalesce. The process terminates when all individuals coalesced into one common ancestor, called the most recent common ancestor (MRCA). Figure 3.2 (B and C) illustrates the genealogy of 5 individuals from the population in Fig. 3.2 A. The present is now found at time 0.

The coalescent process can be depicted as a bifurcating rooted tree, where each internal node represents the coalescence of two individuals, or lineages (Fig. 3.2 C). The root of the tree corresponds to the MRCA. The coalescent process moves in continuous time and makes the same assumption as the diffusion limit: the population size is large enough such that the scaled time \( \tau = t/2N \) can be regarded as continuous. An additional assumption is that the sample size \( n \) is small relative to the population size \( 2N \), such that several coalescence events cannot occur simultaneously [16].

It can then be shown that \( T_i \), the (random) time during which there are exactly \( i \) individuals in the sample, follows an exponential distribution with mean \( 1/(i^2) \). Using the distribution of the \( T_i \)'s, various distributions and expectations of interest can be calculated, such as the MRCA, the total length of the tree, etc. These quantities can also be calculated for variable population size [199], but as I will discuss later, other approaches are used in practice. I note here that the coalescent process is a continuous time Markov chain, where state \( i \) indicates that there are \( i \) lineages present in the sample and \( T_i \) is the waiting time in state \( i \). The chain is absorbed in state 1.

Once the coalescence tree is built, mutations are simply randomly distributed on the tree, where the number of mutations follows a Poisson distribution with mean \( L 2Nu \), where \( L \) is the (random) total length of the tree. So far, I assumed that only two alleles are present in the population and that they can mutate between each other. This is useful for modeling SNP sites, but when analyzing longer genomic sequences, other models of mutations are needed. Such models are the infinite alleles and infinite sites models [199]. The former assumes that each mutation creates a new allele type, while the
The coalescent process

Figure 3.2: Example of genealogy in the Wright-Fisher. The population contains two alleles, A (red) and a (black). (A) A Wright-Fisher population where mutations occur with probabilities $u: A \rightarrow a$ and $v: a \rightarrow A$. (B) Genealogy of a sample of 5 individuals. They find their MRCA in the initial generation. (C) Coalescent tree of the 5 sampled individuals. $T_i$ represents the time during which there are exactly $i$ individuals in the sample.

later assumes that each mutation arises at a site (genomic position) that has not mutated before. The infinite site models implicitly makes the same assumption as the infinite allele model. For both models, only one mutation parameter is needed, $u$, giving the probability of a mutation per site per generation. Within the coalescent framework, the infinite sites model is typically used. Using the Poisson distribution and the distribution of the total length of the tree, various quantities, such as the expected number of observed mutations in the sample, can be calculated [199]. For inference purposes, coalescent methods are sometimes paired with continuous time Markov chains (CTMCs) which describe the evolution of a sequence. Such models are discussed in Section 3.5.

The model presented so far assumes one non-recombining locus. Hudson [91] was the first to describe the coalescent process with recombination going backwards in time. Wiuf and Hein [210] reformulated the original process as going along the genomic sequence. This sequential view is very useful for inference problems but comes at a cost: unlike the backwards in time process, the sequential one is no longer Markovian. To alleviate this, McVean and Cardin [143] introduced the following approximation to the coalescent with recombination.

The sequentially Markov coalescent process

The genealogy under the coalescent with recombination process is a more complex structure, called the ancestral recombination graph (ARG). For each site in the sampled sequences, a path can be followed through the ARG, giving a marginal coalescent tree that can be described by the simple coalescent process. Building the ARG as a process along the sequence, the coalescent tree at position $t+1$ is determined by all marginal coalescent trees at positions
3. Inference of population history

1, 2, ..., l. McVean and Cardin [143] investigated the properties of a Markovian process where the tree at position \( l + 1 \) is determined solely by the tree at position \( l \). They found that, for large recombination rates, the Markovian approximation, entitled the sequentially Markov coalescent (SMC), produces patterns that are consistent in most respects with those resulting from the full coalescent with recombination process.

The way SMC becomes a Markovian process is by disallowing certain classes of recombination and coalescent events. Marjoram and Wall [137] introduced a slight modification to SMC, known as the SMC', by allowing for an additional class of recombinations. They show that the patterns produced by SMC' are closer than the ones from SMC to those originating from the full coalescent with recombination. In the past years, the SMC approximation has been used extensively for inference problems. These new approaches typically add one more layer of approximation: the time is discretized in non-overlapping intervals such that the infinite set of possible marginal coalescent trees can be represented by a finite set of trees. This is done by representing the infinite possible coalescence times from one time interval as just one time point. The discretization allows for the model to be expressed as an HMM (Section 2.2 on page 8), where the hidden states are the discretized marginal coalescence trees and the observed data is given by the genomic sequences.

Among the first inference approaches to simplify the coalescent with recombination as a Markovian process was led by a group of researchers from the Bioinformatics Research Centre at Aarhus University [50, 83, 84, 131, 132]. Their class of models, termed coalescent hidden Markov models (CoalHMMs) have been initially applied in isolation (no migration) models for three species, for example humans, chimp and gorilla, where only one genome per species was available. The relative short evolutionary time between the consecutive speciations of the considered species created patterns of incomplete lineage sorting [50, 83, 84]. Only in [132] did the authors recognize the relation between their CoalHMM models and SMC.

The first inference method to explicitly build on SMC was the pairwise sequentially Markov coalescent (PSMC) [122], published in 2011. PSMC considers only the case of two sampled sequences (or one diploid individual). This allows for the representation of the coalescent tree solely through the time of the MRCA. PSMC includes variable population size by using a piece-wise constant function: the size is constrained to be constant per time interval. PSMC has been shown, using simulation studies, to efficiently recover the variable population size and has, since its publication, been used in a wide range of data analysis.

At around the same time, Paul et al. [159] proposed a different approach, the sequentially Markov conditional sampling distribution (SMCSD), which allows for more than two sequences to be analyzed jointly. To obtain an efficient algorithm, SMCSD simplifies parts of the coalescent trees. The model, implemented in diCal [182], has been shown to provide better estimates, com-
pared to PSMC, for the population sizes that are closer to the present. This is a direct result of the increased sample size, that broadens the time period for which population sizes can be efficiently recovered.

In 2014, two new approaches have been published. PSMC has been extended to MSMC, to handle multiple individuals [181]. MSMC keeps track of just the first coalescent and can be used to analyze migration. ARGweaver [169] was built using a similar approach to [159] and can be used for the analysis of tens of individuals. Unlike SMCSD, it aims at reconstructing the full coalescent trees. It is the first Bayesian method to build on a discretized SMC, and the aim is to obtain a posterior distribution over the ARGs.

The count of possible discretized marginal coalescent trees grows exponentially with the number of sequences. CoalHMM explicitly builds the full state space and even though it can theoretically analyze any number of sequences, it is limited in practice to only a couple. diCal and MSMC avoid constructing this large space. diCal simplifies the coalescent trees, while MSMC only considers the first coalescent event. ARGweaver bypasses this issue by using the conditional sampling distribution in a Gibbs sampler for ARGs.

Starting at the end of 2011 and continuing through 2012, I have been working on extending PSMC to multiple individuals by explicitly considering the underlying discretized marginal trees. I interrupted this work during my visit in 2013 to University of California, Berkeley. As this visit was coming to an end, MSMC and ARGweaver were emerging in the scientific community. Upon my return to Aarhus University, in light of the new development, I decided to discontinue my work on extending PSMC. I will present in the following the main idea of my approach and how the transition probabilities between different marginal trees can be obtained. I additionally show some preliminary results I obtained.

The SMC with multiple sequences

The approach presented here is framed as an HMM and extends PSMC [122], with a few differences. PSMC relies on SMC to calculate the transition probabilities between the coalescent trees, but I chose to use SMC’, due to its increased accuracy. The time is discretized in \( K + 1 \) intervals: \([0, t_1], [t_1, t_2), \ldots, [t_K, \infty)\). Let interval \( k \) be \([t_k, t_{k+1})\), with \( 0 \leq k \leq K \), \( t_0 = 0 \) and \( t_{K+1} = \infty \). Each coalescent event that occurs in interval \( k \) is moved to time \( t_k \leq t < t_{k+1} \) (Fig. 3.3 A and B). This adjustment of the coalescent times can lead to a non-bifurcating tree (Fig. 3.3 B). Each interval \( k \) has a coalescence rate \( c_k = 1/2N_k \), where \( 2N_k \) is the population size in interval \( k \). This effectively allows for a piece-wise constant population size.

According to SMC’, a transition from a coalescent tree \( T \), conditional on one recombination event occurring, proceeds as follows (Fig. 3.3 C)

- a recombination point is chosen uniformly on \( T \);
- a floating lineage is added from the recombination point;
3. Inference of Population History

- the floating lineage coalesces back to \( T \);
- the old branch between the recombination point and the closest internal node (corresponding to a coalescent event) is deleted.

The order of the above operations is different in SMC, where the old branch is deleted before the floating lineage coalesces. SMC', by keeping the old branch a bit longer, adds a probability that the recombination event results in a new tree that is identical to \( T \). This happens when the floating lineage coalesces back into the old branch.

To specify the calculation of the transition probabilities, as given by the above procedure, let \( l_i^x, R_i^x, C_{ij}^x \), with \( x \in \{n,b,a\} \), be

- \( l_i^x = \) the number of lineages in \( T \) in interval \( i \),
- \( R_i^x = \) the probability of having the recombination event in interval \( i \),
- \( C_{ij}^x = \) the probability of the floating lineage that emerged in interval \( i \) to coalesce back in interval \( j \).

The superscript \( x \) can be \( n \) (none), indicating that there is no coalescent event in interval \( i \); \( b \) (before), indicating that the variables encode values before the nodes in \( i \); and \( a \) (after), indicating that the variables encode values after the nodes in \( i \). In Fig. 3.3 B, \( l_k^b = 5 \) and \( l_k^a = 2 \) as there are 5 lineages in the interval before the coalescent events, after which only two lineages remain.

To transition from \( T \), two events must occur: a recombination event in some time interval \( i \), followed by a coalescence event in time interval \( j \geq i \), with probability

\[
P(\text{rec} \mid T) \cdot R_i^T \cdot C_{ij}^N \cdot \frac{1}{l_j^T},
\]

Figure 3.3: (A) Part of a bifurcating coalescence tree corresponding to time interval \( k \). (B) Discretized tree corresponding to the tree in A. All nodes in interval \( k \) are placed at the same time \( t \). (C) Illustration of the SMC' procedure. The recombination point (x) is chosen uniformly on the tree. A floating lineage (dashed) emerges, coalesces back to the tree (circle) and the old branch (dotted) is deleted.
3.3. The coalescent process

where $x$ and $y$ are set correspondingly for intervals $i$ and $j$, depending on the exact position of the recombination and coalescence event relative to any existing nodes in the intervals. The transition probability $P(T^* | T)$ from $T$ to $T^*$ is given as a sum over all events (one recombination and one coalescence) that transform $T$ into $T^*$, of probabilities of the above form. The probability of a recombination event on $T$ follows from the exponential waiting time between recombination events, with rate $2NrL$, where $r$ is the recombination rate per site per generation and $L$ is the total length of the tree

$$P(\text{rec} | T) = 1 - e^{-2NrL}.$$ 

Let $X_j$ be the random variable indicating the time $t_j \leq t \leq t_{j+1}$ where the coalescence nodes in interval $j$ are placed. This variable is distributed as $l^b_j - l^a_j$ independent truncated exponential variables, conditional on them being equal. These variables are the $l^b_j - l^a_j$ coalescence events from interval $j$ which follow an exponential distribution with rate $c_j$, truncated to the interval $j$. It can be shown that $X_j$ has a truncated exponential distribution with rate $(l^b_j - l^a_j)c_j$. Let $f_j(t)$ be this distribution. The probabilities $R^n_i$ and $C^n_{ij}$ for a given tree $T$ are calculated as follows.

**Probability of a recombination event**

As the recombination is placed uniformly on the tree, $R^n_i$ are given by the ratio between the time period where the recombination can occur and $L$, the total length of the tree. As the nodes are not fixed to a specific time, I use the expectations of the tree length and coalescence time. Then

$$R^n_i = \frac{t_{i+1} - t_i}{E[L]}, \quad R^b_i = \frac{E[X_i] - t_i}{E[L]}, \quad R^a_i = \frac{t_{i+1} - E[X_i]}{E[L]}.$$ 

where

$$E[L] = \sum_{k=0}^{K} E[\text{length in } k],$$

$$E[\text{length in } k] = \begin{cases} l^a_k (t_{k+1} - t_k) & \text{if no nodes in } i, \\ l^b_k (E[X_k] - t_k) + l^a_k (t_{k+1} - E[X_k]) & \text{otherwise}. \end{cases}$$

**Probability of coalescing back**

To calculate the probability of the floating lineage coalescing back in interval $j$, the cases $i = j$ and $j > i$ are treated separately. When $i = j$, the recombination and coalescence time are integrated jointly. The density of the recombination time $u$ in interval $i$ is $g(u|t_i, t_{i+1}) = 1/(t_{i+1} - t_i)$. Then

$$C^n_{ii} = \int_{t_i}^{t_{i+1}} g(u|t_i, t_{i+1}) P(\text{coal in } (u, t_{i+1})) \, du,$$
3. Inference of population history

\[
\begin{align*}
A & \quad \text{coal} \quad t_{i+1} \quad \text{rec} \quad t_i \quad \text{coal} \\
B & \quad \text{coal} \quad t_{i+1} \quad \text{rec} \quad t_i \\
C & \quad \text{coal} \quad t_{i+1} \quad \text{rec} \quad t_i 
\end{align*}
\]

Figure 3.4: Relative position of recombination and coalescence events in interval \( i \).

\[
C_{ii}^b = \int_{t_i}^{t_{i+1}} \int_{t_i}^{t} f_i(t) g(u|t_i, t) \mathbb{P} \text{ (coal in } (u, t)) \, du \, dt \quad \text{(Fig. 3.4 A)},
\]

\[
C_{ii}^a = \begin{cases} 
\int_{t_i}^{t_{i+1}} \int_{t_i}^{t} f_i(t) g(u|t, t_{i+1}) \mathbb{P} \text{ (coal in } (u, t_{i+1})) \, du \, dt & \text{if the recombination is placed after the nodes (Fig. 3.4 B),} \\
\int_{t_i}^{t_{i+1}} \int_{t_i}^{t} f_i(t) g(u|t, t) \mathbb{P} \text{ (survive in } (u, t)) \mathbb{P} \text{ (coal in } (t, t_{i+1})) \, du \, dt & \text{if the recombination is placed before the nodes (Fig. 3.4 C).}
\end{cases}
\]

The probability of coalescing or surviving (not coalescing) in a time period follow from the coalescent theory. For example, if \( k \) is an interval where there are no nodes present,

\[
\mathbb{P} \text{ (coal in } (t_k, t_{k+1})) = \int_{t_k}^{t_{k+1}} f_k(t) c_k e^{-\int_k^t c_k \, dt} \, dt = 1 - e^{-t_{k+1} c_k} (t_{k+1} - t_k),
\]

\[
\mathbb{P} \text{ (survive in } (t_k, t_{k+1})) = 1 - \mathbb{P} \text{ (coal in } (t_k, t_{k+1})) = e^{-t_{k+1} c_k} (t_{k+1} - t_k).
\]

When \( j > i \), I use \( p_k^x \), the probability of a coalescence in interval \( k \), where \( x \in \{ n, b, a \} \). Then the total probability of a coalescence in interval \( k \) is either \( p_k = p_k^n \) if no nodes are present, or \( p_k = p_k^b + p_k^a \), otherwise, with

\[
p_k^n = \mathbb{P} \text{ (coal in } (t_k, t_{k+1})) ,
\]

\[
p_k^b = \int_{t_k}^{t_{k+1}} f_k(t) \mathbb{P} \text{ (coal in } (t_k, t)) \, dt ,
\]

\[
p_k^a = \int_{t_k}^{t_{k+1}} f_k(t) \mathbb{P} \text{ (survive in } (t_k, t)) \mathbb{P} \text{ (coal in } (t, t_{k+1})) \, dt .
\]

Then \( C_{ij}^y \), with \( y \in \{ n, b, a \} \), is given by

\[
C_{ij}^y = \left( 1 - C_{ii}^b - C_{ii}^a \right) \prod_{k=i+1}^{j-1} (1 - p_k) p_j^y .
\]

Accuracy of transition probabilities

To assess the accuracy of the above transition probabilities, I used the Kullback-Leibler (KL) divergence [116], defined as, for two discrete probability distri-
3.3. The coalescent process

Figure 3.5: KL divergence for trees with (A) two and (B) three sequences. White points, $f_1$, $f_2$ and $f_3$, correspond to constant population size, and gray points, $v_1$ and $v_2$, correspond to varying population size. In all set-ups, $t_K = 3$ and the intervals are uniformly distributed with $t_{k+1} - t_k = 0.2$ for $f_1$ and $t_{k+1} - t_k = 0.3$ for the rest. The coalescence rates were set to $c_k = 1$ for $f_1$, $c_k = 2$ for $f_2$ and $f_3$, $c_k \in \{2, 1, 4, 2\}$ for $v_1$ and $c_k \in \{2, 1.5, 2, 0.75\}$ for $v_2$, where the changes in the coalescence rates happen at times 0.6, 1.2 and 2.7, respectively. The distributions corresponding to the worst KL divergence from A and B, outlined in red, are given in (C) for two and (D) three sequences, respectively. The plots do not contain the transitions with probability 0.
3. Inference of population history

Distributions $P$ and $Q$

$$D_{KL}(P \parallel Q) = \sum_i P(i) \ln \frac{P(i)}{Q(i)}.$$ 

I calculated the KL divergence for the transition distribution for all possible trees with two or three leaves, using different population sizes. Figure 3.5 shows the KL divergence together with the true and approximated distributions for the trees for which the KL divergence was highest. The true distribution is calculated from SMC$'$ either analytically, when using two sequences with fixed population size, or from simulations performed with MaCS [33], which implements SMC$'$. As Fig. 3.5 shows, the calculated transition probabilities approximate well the true. As expected, the approximation is worse when the true distribution is obtained from simulations, especially when using three sequences. I believe that the main reason for this is the unreliability of the true distribution calculated from simulations, as the possible number of trees is large and it is difficult to obtain good estimates for rare events.

To complete the presented approach, the emission probabilities need to be calculated. They are given by the likelihood of observing each site from the marginal coalescent trees. This is calculated using a continuous time Markov chain (CTMC) which describes the evolution of the sequences on the tree, using Felsenstein’s peeling algorithm [61]. Similar sequence evolution models are also used in phylogenetics and more details are given in Section 3.5.

3.4 Inference of identity-by-descent

The concept of identity-by-descent (IBD) in a population undergoing random genetic drift was first formulated by Malécot. In his description, two individuals are IBD at a locus if they descended from the same ancestor. Under pure drift, eventually all individuals in the population will be IBD. When analyzing genomic data, homologous segments from two individuals are considered IBD if they have been inherited from the same ancestor without recombination.

As the coalescent can be regarded as a generalization of Malécot’s identity-by-descent, it is the natural framework for IBD identification. In fact, when disregarding mutations, every single base within the genome of two individuals is IBD. The problem of IBD tract detection reduces therefore to determining maximally contiguous genomic segments that have been inherited from the same ancestor without recombination. These can be retrieved by predicting recombination breakpoints and filtering tracts based on some measure, typically a minimum tract length. This procedure allows for the tract to contain mutations, but this is not universally agreed upon and most IBD detection methods do not allow mutations to be present.

IBD is playing an increasing role in many different types of data analysis, ranging from association mapping to detection of natural selection [3, 27, 168].
Accurate IBD prediction is therefore needed for downstream analysis. Existing IBD detection methods [24–26, 76] typically rely on characterizing similar haplotypes or considering patterns of linkage disequilibrium, and most of them [24, 26, 76] analyze SNP data. With the current advent of sequencing technologies, methods that take advantage of the full sequence data are needed. I present a new sequence-based demography-aware IBD detection software in


diCal-IBD builds on the existing software diCal [182], which implements the SMCSD [159] approach to the coalescent with recombination. diCal-IBD is the first coalescent-based method, enabling it to incorporate prior knowledge on demographic history, such as variable population size. Additionally, diCal-IBD allows for the presence of mutations within tracts, which are likely to occur in humans due to comparable mutation and recombination rates.

Here, I introduce the main idea and results of Chapter 6. This publication was the result of my visit to University of California, Berkeley where we developed diCal-IBD. In this work, I contributed to all stages, from the development of the methodology, designing and implementing the code, performing experiments and writing the manuscript.

**Approach**

diCal is similar in flavor to PSMC [122] and was the first inference method based on the coalescent with recombination to analyze jointly multiple individuals. It builds on the SMCSD framework that provides an accurate estimation of the probability of observing a newly sampled haplotype given a set of previously sampled haplotypes. diCal analyzes multiple individuals by starting with one haplotype and gradually adding more haplotyes. When only two haplotypes are analyzed, the resulting model is equivalent to PSMC. However, due to additional simplifications used, the model does not incorporate full details about the underlying ARG when more than two haplotypes are used. diCal was originally developed for inference of variable population size as a piece-wise constant population size, but it is currently being extended to handle multiple populations, migration and admixture [192].

diCal is formulated as an HMM (Section 2.2 on page 8) and, when analyzing only a pair of sequences, a decoding (Section 2.2 on page 11) returns the TMRCA for each site. Changes in TMRCA are direct results of recombination events. diCal-IBD infers pairwise IBD tracts as maximally contiguous segments with the same TMRCA. diCal-IBD can incorporate prior on variable population size by approximating it with a piece-wise constant function. As in PSMC and other methods, diCal requires the time to be discretized. diCal-IBD implements a variety of discretizations which use the prior information on
3. Inference of population history

A

Figure 3.6: (A) European demographic history (black) together with the constant size (purple) and piece-wise constant (red) approximation. The vertical dotted lines indicate the discretized time intervals for the two approximations. (B) IBD tracts for one pair of sequences simulated using the European demographic history.

the population size to better partition the time. It includes a newly developed discretization which aims at concentrating the intervals in the time period containing the TMRCA of detectable IBD tracts. Here, an IBD tract is detectable if it is longer than a specified length threshold. diCal-IBD additionally implements data pre-processing, for speed purposes, and tracts post-processing relying on the posterior probabilities obtained from the decoding of diCal.

We implemented in diCal-IBD a framework for one of the applications of IBD tracts, detection of natural selection [3]. diCal-IBD analyzes the predicted tracts to identify regions on which natural selection might have acted. Such regions typically show an increased average of IBD sharing relative to the background average.

Results

We compared the performance of diCal-IBD with state-of-the-art IBD detection methods on simulated data. We first generated ARGs using ms [92] and then simulated sequences on the ARGs. The true pairwise IBD tracts were recovered from the simulated ARGs as maximally consecutive sites with the same TMRCA. When running ms, we used two different demographic histories, the African and European histories (Fig. 6.3 on page 87), respectively, as inferred by Tennessen et al. [200]. Figure 3.6 A shows the European demographic history, its piece-wise constant size approximation and two of the discretizations implemented in diCal-IBD based on a constant and variable population size. The figure illustrates that the demographic prior has
3.5 Inference on a tree

Population genetics is typically concerned with the analysis of data coming from several individuals of the same species. Variable sites, termed polymorphisms, are due to mutations which still segregate in the population. On the other hand, phylogenetics approaches use only one sequence per species to analyze the evolutionary relationship between different species. The variable sites, termed substitutions, arise as ancestral mutations that became fixed. The advances in sequencing technologies have increased drastically the amount of genome data from a wide range of species, including closely related ones. Additionally, genomes from different individuals belonging to the same species are now available. This large collection of data enables the joint analysis of closely related species from which several genomes have been sequenced. For this, new models are needed that reconcile phylogenetics with population genetics and consider both polymorphisms and substitutions.

Some methods do exist that aim at analyzing such data and perhaps one of the most recent ones is PoMo [42]. PoMo, phylogenetic models and some population genetics models use continuous time Markov chains (CTMCs) to model the evolution of a sequence on a tree. CTMCs are very similar to discrete time Markov chains (Section 2.1 on page 7). The difference is that CTMCs move in continuous time. They are typically described using a rate
3. **Inference of Population History**

matrix $Q$ containing the rates at which the process transitions between the different states. For modeling sequence evolution, the most simple CTMC is the Jukes-Cantor model, where the rates in $Q$ are all equal. More complex CTMCs for sequence evolution have been developed, which aim at incorporating specific substitution patterns observed in real data. PoMo uses a CTMC whose states encode the count of the number of alleles in the sample and are therefore dependent on the sample size.

Given a CTMC for sequence evolution and a tree describing the relation between the sequences in the leaves, the likelihood of the data can be efficiently calculated using Felsenstein’s peeling algorithm [61]. This likelihood calculation requires the transition probability matrix $P(t)$ which gives the probability of the CTMC moving between the different states over the time $t$. The transition probabilities are given by a matrix exponential, $P(t) = e^{Qt}$. Typically, the parameters of the CTMC and the tree are not known and they are inferred using, for example, a maximum likelihood approach. PoMo relies on numerical maximization of the likelihood, but other methods [89, 105, 111] use an Expectation-Maximization (EM) algorithm. The EM algorithm requires the calculation of the expectation of certain sufficient statistics, conditional on the starting and ending states of the CTMC.

Various methods exist for calculating the matrix exponential and the sufficient statistics. I address the issue extending three of these methods for the calculation of sufficient statistics and comparing their performance in


Here, I will summarize the three methods and main results from Chapter 7. In this work, I contributed to all stages, from the development of the methodology, designing and implementing the code, performing experiments and writing the manuscript.

**The three methods**

Let us consider a CTMC with $n$ states. The statistics required by the EM algorithm are $T_c$, the total time spent in state $c$, and $N_{cd}$, the number of jumps from $c$ to $d$, given in Eq. (7.2) on page 99. As discussed in Chapter 7, many applications require linear combinations of these statistics. I have extended the methods summarized here to efficiently calculate such linear combinations, defined in Eq. (7.4) on page 99.

**EVD:** The first method considered relies on the eigenvalue decomposition of the rate matrix $Q$. This method is the most widely used one and was initially applied for calculating the transition probability matrix. Employing the decomposition, the linear combination of the statistics can be obtained using matrix algebra, as shown in Eq. (7.10) on page 101.
3.6 Conclusions and future work

**UNI:** The uniformization method, just as the eigenvalue decomposition, was first used for the calculation of $P(t)$. The uniformization method arose as an alternative description of the CTMC as a discrete time Markov chain that is subordinated to a Poisson process. This reformulation resulted in different mathematics that enabled the calculation of the transition probability matrix and sufficient statistics as an infinite sum. In Eq. (7.13) on page 102 I show how this sum can be rewritten to obtain the linear combinations. I also discuss the truncation of the sum in a way such that the total error is bounded, illustrated in Fig. 7.1 on page 103.

**EXPM:** The last method considered, termed exponentiation, calculates the sufficient statistics from the matrix exponential of a matrix of size $2n \times 2n$. This matrix contains in the top right corner a matrix that can be manipulated to obtain the linear combinations of the statistics rather than just one statistic. Given efficient matrix exponentiation, the EXPM method is the most simple and easy to implement of the three.

**Results**

We tested the three methods with regard to their accuracy by using a CTMC of variable size inspired by the Jukes-Cantor model, where all off-diagonal rates are equal. Additionally, we used a second CTMC for DNA sequence evolution. For both models, the sufficient statistics can be calculated analytically. The accuracy of the three methods is given in Fig. 7.2 on page 105, indicating that all methods have good accuracy.

In inference problems, the speed of the methods is often a concern. I discuss in Section 7.2 on page 104 different properties of the three methods that affect the speed. In particular, the computations required by EVD and UNI can be divided in two parts, out of which the first part can be reused for a set of different time points $t$. This partition of computation is illustrated in Table 7.1 on page 105. We also inspected the speed of the three methods in a series of eight experiments, with results given in Fig. 7.3 on page 106. EXPM was the slowest method, while EVD and UNI competed for being the fastest. Different applications are certainly better suited for one or the other.

The main conclusion of the paper was that, although the standard method based on eigenvalue decomposition performs well, the uniformization method is often overlooked despite the fact, as we showed, it is comparable in performance to EVD and can sometimes outperform it. Our message seems to have been received by the scientific community and Bio++ currently implements the UNI method [51, 74].

3.6 Conclusions and future work

In this chapter I have introduced work from my PhD studies that falls under the area of population genetics. I have addressed the issue of developing
3. **Inference of population history**

new methodology to analyze molecular data and infer population history using both classical population genetics models, such as the Wright-Fisher, but also relying on the newer approaches based on the coalescent theory. Despite the simplicity of the mathematical formulation of the Wright-Fisher model, analytic results are difficult, if not impossible, to obtain. I presented how the beta distribution can be used for approximating the allele frequency distribution arising from the Wright-Fisher. I extended the approximation to include selection and explicitly model the probabilities of loosing or fixing the allele. I then discussed the Markovian approximation to the coalescent with recombination and presented how this framework is the natural setting for detection of identical-by-descent (IBD) tracts in individuals. At the end of the chapter, I briefly considered the need of merging population genetics with phylogenetics models and inspected various quantities required for such analysis, arising from continuous time Markov chains (CTMCs).

With the exception of the work on CTMCs, the results presented here are from the second half of my PhD and represent my current focus and interest. These topics can be extended and improved in many ways, some of which are discussed below.

Upon the completion of my PhD, I plan to resume the work regarding the beta approximation to the Wright-Fisher. The coming future plans cover investigations of inference accuracy of selection rates, both for data described by a tree but also time-serial data for which observations are available at different time points. However, I would like to improve the method in a number of ways. Firstly, I think a more accurate approximation of the probabilities at the boundaries is needed. Efficient solutions as a function of time exist based on Kimura’s diffusion approximation, but, to my knowledge, these are limited to pure drift.

Additionally, I would like to investigate a different formulation of the beta approximation, in an almost analytic form, as a function of the initial frequency \(x_0\). This would require expressing the two moments and probabilities at the boundaries as an almost analytic function in \(x_0\), where the terms that are independent of \(x_0\) would be calculated recursively. Such a formulation would improve on the accuracy and speed calculations of the data likelihood on a tree. I am not sure though that this is possible. I also consider that incorporating variable population size could be relevant, especially given the multitude of results regarding inference of population size using coalescent theory, such as PSMC and diCal.

Another attractive perspective is the incorporation of recombination. It would be of interest to investigate whether an SMC-like approach can be implemented in the beta distribution approximation, where the allele frequency at one locus is distributed conditionally on the allele frequency at a previous locus. The strength of this dependency would be determined by the recombination rate between the two loci. Such an approach would make the model suited for analyzing larger data sets containing loci that are close enough such
3.6. Conclusions and future work

that they cannot be considered as evolving independently.

The Wright-Fisher model can be used to analyze data from multiple species. However, in this setting, limiting the analysis to bi-allelic loci is not reasonable. New approximations are required for modeling jointly the allele frequencies as a function of time for multi-allelic loci. The generalization of the beta distribution, the Dirichlet distribution, has been used for this purpose, but preliminary results obtained by Asger Hobolth indicate that this distribution is not flexible enough to accurately model the allele frequency distribution.

IBD detection is receiving much attention in the research community. One issue is the detection of multiple, rather than pairwise, IBD tracts. A multiple IBD tract is a tract shared by more than two individuals. Methods such as [165] aim at detecting clusters of IBD tracts by building on pairwise IBD tracts. The main idea is that the IBD relation is transitive: if A is IBD with B and B is IBD with C, than A should be IBD with C. However, pairwise detection methods can potentially invalidate such relationships. This issue would be solved if multiple IBD tracts would be detected in a coalescent framework and render methods like [165] redundant. diCal-IBD is limited to only pairwise IBD due to the nature of diCal and its simplification of the ARG. I believe that methods such as ARGWeaver can be used for this purpose. ARGWeaver produces, though, a posterior distribution over the ARGs, while for IBD inference, one predicted ARG is required.

For IBD detection, the issue of phased data is of central importance. Some methods, such as IBDseq, are able to handle unphased data. However, diCal-IBD and all coalescent-based inference methods require phased data. Nevertheless, the large amount of newly generated data is typically unphased. High quality phasing can be obtained by sequencing trios, where the genome of the parents aids in phasing the genome of the offspring. This procedure is expensive and not suited for large sequencing projects that aim at characterizing whole populations. Computational phasing methods do exist, some of which benefit from larger data sets, as it increases their accuracy. However, I think it would be of great interest if coalescent-based methods could directly consider unphased data rather then relying on a pre-processing phasing step. The computational phasing will most likely create new errors in the data set, leading to inaccurate inference, as investigated briefly in the MSMC paper. The authors of ARGWeaver mentioned that they believe it possible to directly integrate over all possible phasings. Such an approach would greatly improve the applicability of the coalescent-based methods.
Inference of patterns

Biological data, the object of study of bioinformatics, contains intricate patterns generated from the biological processes governing the data. The analysis and inference of these patterns can shed light on the underlying biological processes. Here, the definition of “pattern” is flexible and covers a wide range of meanings, from simple words or regular expressions, to RNA secondary structure. In this chapter, I present two independent problems of pattern inference within bioinformatics which I addressed during my PhD.

This work resulted in one manuscript and three publications

- Section 4.2 (Chapter 8): Motif discovery in ranked lists of sequences [154];
- Section 4.3 (Chapter 9): Algorithms for hidden Markov models restricted to occurrences of regular expressions [196];
- Section 4.5 (Chapter 10): Evolving stochastic context-free grammars for RNA secondary structure prediction [6];
- Section 4.6 (Chapter 11): Oxfold: kinetic folding of RNA using stochastic context-free grammars and evolutionary information [7].

The first problem I address is the statistical significance of patterns occurring in biological sequences. Here, a pattern is a regular expression over the alphabet of the sequences. The specific background for this problem, relying on embedded Markov chains and deterministic finite automata, is introduced in Section 4.1. My contribution within this topic was to incorporate the embedding technique in more complex settings. In Section 4.2 I present the additional background theory I used and summarize the results of Chapter 8. This was a collaboration on investigating whether occurrences of patterns within lists of ranked sequences are randomly distributed. Additionally, I outline in Section 4.3 the approach and results of Chapter 9, where I introduce new algorithms for hidden Markov models to account for pattern occurrences in the hidden sequence.

In the second part of this chapter, the pattern of interest is the secondary structure of RNA sequences. In Section 4.4 I introduce the background on how stochastic context-free grammars (SCFGs) have been used for RNA secondary structure prediction. The two publications which build on this are
the result of two projects initiated at the Oxford Summer School on Computational Biology. Because of the origin of the projects, these results are somewhat different than the rest of my work. I summarize in Section 4.5 the results of Chapter 10, a collaboration where we developed an evolutionary algorithm for searching the large space of SCFGs for a good RNA secondary structure predictor. Following that collaboration, I have been involved in the co-supervision of work to couple an SCFG with the kinetics of RNA folding and evolutionary information contained within an RNA alignment. This work, presented in Chapter 11, is outlined in Section 4.6.

This chapter concludes with Section 4.7, containing the main conclusions and potential future work.

4.1 Markov chain embedding and DFAs

Starting in the 1950s, there has been considerable interest in the statistical and probabilistic properties of words occurring in random strings, in fields ranging from reliability theory to analysis of biological sequences. The typical aspects of word occurrences are

- How many times does it occur?
- Where does it occur?

Once the above are determined, the focus is to discover the words for which the number of occurrences or their position is statistically significant, i.e. it cannot simply be explained by the background probabilistic model that describes the underlying sequence. If a word occurs more or less often than expected by pure chance, it is likely that the word is involved in some sort of biological function [172].

Such analysis does not need to be restricted to investigating just one word. I will therefore call it, generally, a pattern. Finding the probability of the observed number of counts of a pattern has been under great focus and a variety of methods exist, with different approaches to:

- What is a pattern? A word, a collection of words, a regular expression;
- Can the occurrences of the pattern overlap?
- What is the background probabilistic model describing the random sequence? A Bernoulli trial, a Markov chain, a hidden Markov model.

Fu and Koutras [63] were the first to calculate the distribution of the number of occurrences in a unified framework building on the Markov chain embedding technique. They used Bernoulli tries as a background model and investigated various types of consecutive success runs, providing formulas easy to evaluate. Since then, the Markov chain embedding technique has been extended to other types of background models, e.g. Markov chains [155].

Formally, a nonnegative integer random variable $Y_T$ can be embedded in a Markov chain $X(t)$ defined on a finite state space $S$ if
4.1. Markov chain embedding and DFAs

- there exists a finite partition of the state space, $S = S_0 \cup \cdots \cup S_k$;
- for every $0 \leq i \leq k$, $P (Y_T = i) = P (X(T) \in S_i)$.

For pattern occurrences, the nonnegative random variable $Y_T$ represents the count of the pattern and $P (Y_T = i)$ is the probability of observing $i$ pattern occurrences in a random sequence of length $T$. In data analysis, the statistical significance is reported as a $p$-value, $P (Y_T \geq k)$, the probability of a random sequence of length $T$ containing at least $k$ pattern occurrences, where $k$ is the number of patterns contained in the observed sequence.

Counting words

Let us consider the situation where the pattern is just a simple word. I exemplify here how to build the embedded Markov chain for DNA sequences. To start with, let us assume that the background model from which the DNA sequences are generated assumes that each position is independent and identically distributed, with a discrete probability distribution $p_A, p_C, p_G$ and $p_T$ over the four nucleotides. To calculate the $p$-value, the probability of the observed number of occurrences $k$ of a word $w$, let’s say $w = \text{ATG}$, we first build a directed labeled graph, $G$, that models $w$ (Fig. 4.1 A). This graph has states $0 \leq j \leq n$, where $n$ is the length of the word, and each state indicates that we are currently placed at the $j$th character in the word. The edges of $G$ are labeled with the four nucleotides. Given a DNA sequence, we start in state 0, and guided by the nucleotides in the observed sequence, we move through the graph using the labeled edges

$$
0 1 0 1 2 3 1 1 2 3 0
A C A T G A A T G C .
$$

Using $G$, we can build a Markov chain on the same state space with transition probability matrix $P$ (Fig. 4.1 B), by using the labels (nucleotides) on the edges to calculate the transition probabilities between the states. The embedded chain will be used to count the number of occurrences of $w$, where each occurrence $i$ will be composed of the states $j$ from $G$ (Fig. 4.1 C). The state space of the embedded chain is

$$
S = \{(0, j) : 0 \leq j < n\} \cup \{(i, j) : 0 < i < k, 1 \leq j \leq n\} \cup \{(k, n)\},
$$

where $i$ indicates the number of already observed words (Fig. 4.1 C). States $(0, n)$ and $(k, j)$ for $j \neq n$ are not part of $S$ because they would never be reached

- state $(i, n)$ implies we are placed at the last character of the word and therefore $i > 0$;
- once we reach the desired number of occurrences $k$, i.e. reach the state $(k, n)$ for the first time, we stop the chain.
4. Inference of patterns

Due to the last condition, state $(k, n)$ is an absorbing state. The transition probability matrix of the embedded chain, $P_k$ (Fig. 4.1 D), is sparse and the (potentially) positive entries are obtained from $P$ (Fig. 4.1 D)

\[
P_k((i, j_1); (i, j_2)) = P(j_1; j_2) \quad \text{if } j_2 \neq n ,
\]
\[
P_k((i, j); (i + 1, n)) = P(j; n) \quad \text{if } j \neq n ,
\]
\[
P_k((i, n); (i, n)) = P(n, n) .
\]

The previously considered DNA sequence with 2 occurrences of the word has the following corresponding path through the embedded chain

\[
\begin{array}{ccccccc}
0 & 1 & 2 & 3 & 1 & 2 & 3 \\
0 & 1 - P_A & P_A & 0 & 0 & 0 & 0 \\
1 & P_C + P_G & P_A & P_T & 0 & 0 & 0 \\
2 & P_C + P_T & P_A & 0 & P_G & 0 & 0 \\
3 & 1 - P_A & P_A & 0 & 0 & 0 & 0 \\
\end{array}
\]

The partition of $S$ is given by $S_i = \{(i, j) : (i, j) \in S\}$. The $p$-value is obtained from the discrete time Markov chain theory (Section 2.1 on page 7)

\[
P \left( \text{observe } w \ \text{at least } k \text{ times} \right) = P \left( Y_T \geq k \right) = \left( P_k^T \right)_{(0,0), (k, n)} .
\]

So far, we assumed that the positions in the DNA are independent. This assumption can be unrealistic and a more appropriate background probabilistic model could be a Markov chain, where the distribution at some position $j$ is determined by the previous position, $j - 1$. Such neighboring dependencies in DNA sequences have been shown by several empirical studies [11]. The presented construction can be easily extended by adjusting the starting graph.

Figure 4.1: Embedded chain. (A) A directed labeled graph that corresponds to observing the word ATG. Arcs are colored according to the labels: $A$ - purple, $C$ - red, $G$ - green, $T$ - blue. (B) Transition probability matrix for a Markov chain built based on (A). (C) Embedded Markov chain for observing the word ATG at least twice. (D) Transition probability matrix for the embedded chain from (C). Light gray entries have probability zero, while the dark gray entry has probability one, indicating that the state $(2, 3)$ is absorbing.
4.1. Markov chain embedding and DFAs

\( \mathcal{G} \). To this extent, each state \( j \) in the original graph \( \mathcal{G} \) needs to be copied at most 4 times, \((j, A), (j, C), (j, G) \) and \((j, T)\), such that it encodes the previous observed character. Some of these states, e.g. \((1, C)\), will not be reachable and therefore they can be removed from the graph. Let us assume that DNA sequences are described by a Markov chain with transition probabilities of the form \( p_{A,A}, p_{A,C}, \ldots \), where \( p_{A,C} = P(C \mid A) \). Then the transition probability matrix \( P \) for the word \( w = ATG \) becomes

\[
\begin{pmatrix}
(0, A) & (0, C) & (0, G) & (0, T) & (1, A) & (2, T) & (3, G) \\
(0, A) & 0 & p_{A,C} & p_{A,G} & p_{A,T} & p_{A,A} & 0 & 0 \\
(0, C) & 0 & p_{C,C} & p_{C,G} & p_{C,T} & p_{C,A} & 0 & 0 \\
(0, G) & 0 & p_{G,C} & p_{G,G} & p_{G,T} & p_{G,A} & 0 & 0 \\
(0, T) & 0 & p_{T,C} & p_{T,G} & p_{T,T} & p_{T,A} & 0 & 0 \\
(1, A) & 0 & p_{A,C} & p_{A,G} & 0 & p_{A,A} & p_{A,T} & 0 \\
(2, T) & 0 & p_{T,C} & 0 & p_{T,T} & p_{T,T} & p_{T,A} & 0 & p_{T,G} \\
(3, G) & 0 & p_{G,C} & p_{G,G} & p_{G,T} & p_{G,A} & 0 & 0 
\end{pmatrix}
\]

The structure of \( P \) is essentially the same as in Fig. 4.1 B, but with the transition probabilities split according to the different nucleotides. Once the graph \( \mathcal{G} \) and corresponding transition matrix \( P \) are calculated, we obtain the embedded chain and \( P_k \) exactly as before. The state of the embedded chain will be of the form \((i, j, N)\) where \( i \) is the number of already observed words and \((j, N)\) are the states from \( \mathcal{G} \). When we calculate the \( p \)-value, we have to ensure starting and ending in the correct states. Let \( N_f \) be the last nucleotide in the word, then the \( p \)-value becomes

\[
P \left( Y_T \geq k \right) = \sum_{N \in \{A, C, G, T\}} \left( P_k^T \right)_{(0,0,N),(k,n,N_f)} .
\]

**Counting regular expressions**

This framework for counting words can easily be extended to regular expressions. The graph \( \mathcal{G} \) (Fig. 4.1 A) is, in fact, a deterministic finite automaton which recognizes the language described by the regular expression \((A|C|G|T)^* ATG\), with states 0 and 3 being initial and accepting, respectively.

Automata theory is concerned with the study of abstract computing devices, or “machines”. The study of abstract machines has been pioneered by Turing in the 1930’s, having a great influence in the development of computer science [90]. Finite automata were introduced in the 1940’s and in the late 1950’s, the linguist Chomsky began the study of formal “grammars”. Even though grammars are not machines, they are in close relationship with automata. Generally speaking, a grammar generates a language, while an automaton recognizes it. The grammars, together with automata, are widely used in computer science and are applied in a wide range of areas. As I will
present in Section 4.4, grammars can be used in molecular biology for modeling RNA secondary structure, while finite automata can be used in almost any system which, at all times, is in one of a finite number of “states”. The purpose of a state is to remember the relevant part of the system’s history. The automaton begins in a starting state and moves between states using transitions, as response to inputs. Some of the states are marked as accepting or final, indicating that the sequence of inputs used so far is good in some way. Finite automata can be regarded as non-probabilistic discrete time Markov chains with labeled transitions.

One of the widely known application for finite automata is pattern matching, as they recognize regular languages, described by regular expressions. A regular expression is used to specify a set, potentially infinite, of strings, formally called a language. For example, the regular expression \( 0^* | 1^* \) denotes the language consisting of all strings that are either a single 0 followed by any number of 1’s, or a single 1 followed by any number of 0’s. In regular expressions parenthesis can be used for grouping and mark the precedence of the operators. Without parenthesis, the star \( ^* \) operator has the highest precedence. For example, the regular expression \( 0^* \) is different than \( (01)^* \). Even though other operators can be used in regular expressions, concatenation \( (01) \) is the concatenation of 0 and 1, vertical bar \( | \) (separates alternatives, replacing a boolean or) and star \( ^* \), are sufficient for specifying any regular expression.

Finite automata can be grouped in two equivalent types: deterministic and nondeterministic. Even though they can recognize the same regular languages, deterministic automata are easier to implement, but typically have a higher number of states compared to the nondeterministic ones. Given a regular expression, the pathway used to build a deterministic finite automaton (DFA) that recognizes the corresponding regular language is to first built a nondeterministic finite automaton guided by the regular expression, and then convert it to a deterministic one [90]. The automaton in Fig. 4.1 A recognizes all strings ending in \( ATG \), i.e. the regular expression \( (A | C | G | T)^* ATG \).

Using the well described theory of finite automata and regular expressions [90], we can now build a DFA to help us construct an embedded Markov chain for counting occurrences of any pattern that can be written as a regular expression. If the pattern of interest is \( r \) and the alphabet of the strings is \( \mathcal{H} \) (such as the four nucleotides), then building the embedded Markov chain for \( r \) is a three step process

- build the DFA that recognizes \( (h_1 | \cdots | h_n)^* r \);
- build the corresponding Markov chain transition probability, \( P \);
- build the embedded Markov chain transition probability \( P_k \) from \( P \).

The second and third steps follow exactly as previously described. When the pattern of interest was just a simple word, there was a natural meaning of each state: state \( j \) indicated that we were positioned on character \( j \) of the
4.2 Patterns in lists of ranked sequences

word, or, alternatively, that we have observed the prefix of the word of length $j$. When the pattern is a regular expression, the states cannot be given an interpretation any longer. An example of the DFA and embedded chain for matching the pattern $A C^* T (G | C)$ is shown in Fig. 8.2 on page 114.

The occurrences of the patterns could potentially overlap. To handle this situation, the DFA needs to be altered slightly. This is discussed in details in Section 9.1 on page 127.

Using DFAs to build efficient Markov chain embedding is not a novel approach [153, 155], but has been previously limited to only embedding patterns in random sequences with independent positions or described by Markov chains. In my PhD, I have been involved in developing this framework further, in two different settings, detailed in the following two sections.

4.2 Patterns in lists of ranked sequences

Biological experiments often generate sequences which are ranked in a list by some functional property. Such an example would be an RNA expression experiment, where the sequences are the various RNA sequences detected in the tissue, while their ranking is induced by the expression level. In this setup, patterns that are statistically significant that occur preponderantly at one of the ends of the list are probably

- part of a binding factor or enhancer, if the pattern is enriched in highly expressed RNAs;
- part of an inhibitor, if the pattern is underrepresented in the lowly expressed RNAs.

When investigating patterns in ranked sequences, it is not enough to calculate the $p$-value for each sequence within the list. A new question has to be addressed: are the $p$-values distributed randomly within the ranked list? I present a solution to this problem in


Chapter 8 introduces a new tool, Regmex (REGular expression Motif EXplorer). For a given pattern (regular expression), the evaluation in Regmex can be divided in two steps

- for each sequence, calculate the $p$-value using the embedding technique coupled with DFAs (Section 4.1);
- determine whether the $p$-values are distributed randomly within the list or they form clusters of low $p$-values.

Regmex can additionally explore the space of $k$-mers (words of length $k$), for a given $k$, and for each perform the above analysis, reporting only
4. Inference of Patterns

the significant findings. This is currently restricted to only k-mers and the approach is a simple exhaustive search, as in other previous methods [170].

The significance of the distribution of the p-values within the list is calculated using three different procedures

- a Brownian bridge, developed by Jacobsen et al. [95] and used in [170];
- a modified rank sum statistic, based on the Wilcoxon rank sum test [136];
- a random walk [59].

Various methods exist to address this problem [56, 119, 170, 191, 203]. To my knowledge, Regmex is the first tool to use

- regular expressions;
- the modified rank sum statistic;
- the random walk.

As show in Section 8.3 on page 117, regular expressions allow for more different types of patterns to be investigated and increase the applicability of the method. Additionally, the different procedures for calculating the significance of the distribution of the p-values perform differently in different scenarios and each of them is better suited than the others in specific cases.

Here, I will present how the random walk theory can be used to assess the randomness of the distribution of the p-values within the list, and summarize the main results of Chapter 8. My contribution covered the application of the Markov chain embedding and random walk theory, designing and writing code for building the DFA and calculating significance using random walks, and drafting of the manuscript.

Random walks

A random walk is a mathematical formalization of a path that consists of a succession of random steps. It has applicability in many fields, from economics to physics. Within bioinformatics, random walks have been used to infer risk-active pathways, as a step required for precise cancer classification [125], predict protein localization within cells [215] and predict microRNAs from genome sequence data [220]. Perhaps the most known algorithm in bioinformatics that relies on random walk theory is BLAST [59].

Suppose we have the p-values for a certain pattern for a list of ranked sequences (Fig. 4.2 A). These p-values are transformed into steps. Because low p-values are statistically significant, we assign a step of +1 to a p-value that is at most 0.05, and a step of −1 otherwise. As we go through the p-values in the order of the ranked list, we perform a random walk by starting at 0 and accumulating the corresponding steps. A random walk can be depicted graphically as in Fig. 4.2 B. Points in the walk that are lower than any previously reached point, called ladder points, are depicted as filled circles. The
4.2. Patterns in lists of ranked sequences

Figure 4.2: Random walk on $p$-values. (A) $p$-values for a list of 25 ranked sequences. The blue line indicates the 0.05 threshold used to transform the $p$-values into steps: a $p$-value below 0.05 will give a step of +1, and one above will correspond to a step of −1. (B) The random walk corresponding to the $p$-values in (A). Ladder points are marked as filled black circles and the filled gray circle represents the start of the random walk at 0. (C) Random walk equivalent to the one in (B). When the random walk reaches the boundary -1, it restarts (from 0). Starting points are filled with gray. The points where the boundary -1 is reached correspond to the ladder points in (B). (D) Excursions that have a positive maximum height. The maximum height of the excursion is depicted.

The blue line indicates the 0.05 threshold used to transform the $p$-values into steps: a $p$-value below 0.05 will give a step of +1, and one above will correspond to a step of −1. (B) The random walk corresponding to the $p$-values in (A). Ladder points are marked as filled black circles and the filled gray circle represents the start of the random walk at 0. (C) Random walk equivalent to the one in (B). When the random walk reaches the boundary -1, it restarts (from 0). Starting points are filled with gray. The points where the boundary -1 is reached correspond to the ladder points in (B). (D) Excursions that have a positive maximum height. The maximum height of the excursion is depicted.

Part of the walk from a ladder point until the highest point reached before the next ladder point is called an excursion. The maximum height of an excursion can be calculated as the difference between the highest point and the starting ladder point. In Fig. 4.2 D, the maximum excursion heights are 1, 1 and 4. When the walk moves from one ladder point immediately to the next, the corresponding height is 0.

Alternatively, the random walk can be restarted every time the boundary -1 is reached. The resulting random walk is equivalent to the original one (Fig. 4.2 C) and it preserves the maximum excursion heights. The random walk where we restart upon reaching the boundary is mathematically easier...
4. Inference of patterns

to evaluate and provides straightforward means to calculate the maximum excursion height: the maximum point reached before the walk is stopped.

The main focus in random walk theory is, given a probability distribution over the steps, to calculate the probability of an excursion having a maximum height that is larger than some observed. Within the problem of finding clusters of low \( p \)-values that are significant, an excursion corresponds to such a cluster. The probability of the maximum height of the excursion directly relates to the statistic significance of clusters of low \( p \)-values.

The random walk depicted in Fig. 4.2 is rather restrictive, in that only two types of steps are allowed (-1 and +1). More general random walks can be used [59], but here I limit the presentation to walks that have only one negative step, -1, and, potentially, several positive steps. This is motivated by the fact that high \( p \)-values are not significant, so they decrease the value of the walk with -1, but low \( p \)-values can be classified in several groups, each corresponding to a different step: the lower the \( p \)-value, the higher the step.

Let \( S \) be the (random) step which takes values in the set \(-1, 0, 1, \ldots, d-1, d\), with corresponding probabilities \( P(S = s) = p_s, -1 \leq s \leq d, \) some of which might be 0. The random walk theory requires, as in [59], that \( p_d > 0 \) and that the step size is of the “negative mean” type

\[
E[S] = \sum_{s=-1}^{d} s p_s < 0 .
\]

The random walk starts in 0 and stops when it reaches the boundary -1. The mean of the (random) step \( S \) is required to be negative such that the walk will eventually reach the boundary. We are interested in the distribution of \( Y \), the maximum height the walk reaches before stopping.

Random walks are special cases of discrete time Markov chains and can thus be analyzed using Markov chain theory, as shown below. However, due to special properties, specific methods can be used for analysis of random walks, particularly very efficient asymptotic approximations relying on a geometric-like distribution.

Using a geometric-like approximation

Ewens and Grant [59] discuss, under a more general random walk, that the distribution of \( Y \) can be asymptotically approximated by a geometric-like distribution

\[
P(Y \geq y) \approx C e^{-\theta^* y} , \quad \text{with} \quad \sum_{s=-1}^{d} p_s e^{s \theta^*} = 1 .
\]

By substituting \( e^{\theta^*} = x \) we obtain

\[
p_{-1} \cdot \frac{1}{x} + (p_0 - 1) + \sum_{s=1}^{d} p_s x^s = 0 .
\]
4.2. Patterns in lists of ranked sequences

Therefore, to find \( \theta^* \), we need to find the zeros of the above polynomial, problem for which efficient algorithms exist, such as RPOLY [164].

For the random walk considered here, the constant \( C \) is given by [59]

\[
C = -\frac{\mathbb{E}[S] (1 - e^{-\theta^*})}{\mathbb{E}[Se^{S\theta^*}]} = -\left( \sum_{s=-1}^{d} s p_s \right) (1 - e^{-\theta^*}) \frac{\sum_{s=-1}^{d} s e^{s\theta^*} p_s}{\sum_{s=-1}^{d} s e^{s\theta^*} p_s}.
\]

Using Markov chain theory

Using the geometric-like distribution in practice can be problematic if the observed random walk is not long enough to reach the boundary. This is where using the Markov chain theory (Section 2.1 on page 7) can be useful.

Let us define a new random variable, \( Y(T) \), to be the maximum height of a walk of length \( T \). The previously defined variable \( Y \) can be regarded as \( Y(\infty) \).

To calculate both \( P(Y(T) \geq y) \) and \( P(Y \geq y) \), we define a Markov chain \( X(t) \) with two absorbing states: -1 and \( y \). The state space of the Markov chain is given by the values of the random walk: \{-1, 0, 1, ..., y\}. The transition probability from state \( i \) to \( j \) for which \( j - i > d \) or \( j - i < -1 \), is 0. This is because there is no step large enough that can take the walk from \( i \) to \( j \) in one go. For the rest of the states, \( 0 \leq i < y \), the transition probabilities are

\[
p_{i,i+s} = p_s \quad \text{if} \quad -1 \leq s \leq \min(d, y - i), \quad (4.1)
p_{i,y} = \sum_{s=y-i+1}^{d} p_s \quad \text{if} \quad y - i < d. \quad (4.2)
\]

Equation (4.1) describes the transitions for states that are far away from the boundary \( y \). Once the chain approaches \( y \), we need to ensure, using Eq. (4.2), that when a step would transition the random walk above \( y \), the chain stops, in fact, at \( y \). From this, we obtain directly that

\[
P(Y(T) \geq y) = P(X(T) = y) = \left( P^T \right)_{0,y}.
\]

where \( P \) is the transition probability matrix of \( X(t) \).

The theory developed for absorbing Markov chains [73] can be used to calculate the probability of the chain eventually being absorbed in \( y \). For this, we use the fundamental matrix \( N \) defined as \( N = \left( I - P_{0:(y-1)} \right)^{-1} \), where \( I \) is the identity matrix and \( P_{0:(y-1)} \) is the matrix obtained from \( P \) by only considering the rows and columns from 0 to \( y - 1 \). As shown in [73], this matrix verifies that

\[
N = \sum_{t=0}^{\infty} \left( P_{0:(y-1)} \right)^t.
\]
4. Inference of patterns

![Figure 4.3: Maximum height distribution.](image)

The figure shows the distribution of the maximum height for a random walk with probability 0.2 for a +1 step and 0.8 for a −1 step. The distribution is calculated using: the Markov chain theory for chains of length 15 (red) and 20 (blue); the absorption probability from Markov chain theory (green); and the geometric-like approximation (black).

We finally obtain, as in [73],

$$P(Y \geq y) = P\left(\text{the chain is eventually absorbed in } y\right) = \sum_{t=0}^{\infty} \sum_{i=0}^{y-1} P(X(t+1) = i \mid X(1) = 0) \cdot p_{i,y}$$

$$= \sum_{i=0}^{y-1} n_{0,i} \cdot p_{i,y},$$

where $n_{0,i}$ are entries in the matrix $N$ and $P_{[y-1],y}$ is a column vector containing the transition probabilities from all states $0 \leq i < y$ to state $y$.

Figure 4.3 shows the distribution of the maximum height for a simple random walk with a positive step of +1 with probability 0.2. The distribution $P(Y \geq y)$ is calculated using both the Markov chain theory and the geometric-like approximation. Additionally, the figure shows the distribution of $Y(T)$ for $T$ equal to 15 and 20, respectively. In the shorter chain, the probabilities of the maximum height at the tail of the distribution are lower than the rest. This is due to the fact that, the chain being shorter, it is less probable for the random walk to have a maximum height that is so large. The same trend is observed when comparing the chain of length 20 with the probability of absorption, which essentially assumes an infinite length. The quality of the geometric-like approximation is increasing towards the tail of the distribution, as expected, due to its asymptotic nature.

**Maximum of $n$ excursions**

Typically, when calculating pattern occurrences for ranked lists and building the random walk, the restarting process described previously will generate a set of $n$ independent excursions. Each of these excursions will have a maximum height, and the interest will lie in the probability of observing a maximum
4.2. Patterns in lists of ranked sequences

height that is at least as large as the observed maximum height among the
*n* excursions. We calculate this using $Y_{\max} = \max\{Y_1, Y_2, \ldots, Y_n\}$, where $Y_i$ are the independent and identically distributed maximum heights of the *n* independent excursions, as follows

$$P(Y_{\max} \geq y) = 1 - P(Y_{\max} < y)$$

$$= 1 - \prod_{i=1}^{n} P(Y_i < y)$$

$$= 1 - \prod_{i=1}^{n} (1 - P(Y_i \geq y))$$

$$= 1 - (1 - P(Y \geq y))^n.$$  

Results

Regmex, a tool for evaluating correlation of patterns with lists of ranked sequences, is available as an R package. The novelty of this tool resides in both the definition of a pattern and the methods used for calculating pattern enrichment. As opposed to existing methods, which typically use words as patterns, Regmex allows for the pattern to be as complex as a regular expression, using the approach presented in Section 4.1. As illustrated in Section 8.3 on page 118, the regular expression allows for specific hypothesis testing.

Regmex provides three different approaches for calculating the pattern enrichment in a list of ranked sequences: a Brownian bridge, a modified rank sum statistic and a random walk. The random walk approach, presented here in detail, has the highest sensitivity among these methods (Fig. 8.4 on page 116). It can detect pattern enrichment as a significant cluster of low *p*-values, regardless of where in the sequence list it occurs, whereas the other two methods are more sensitive to pattern enrichment in the beginning and end of the sequence list.

The power of combining words into one regular expression and analyzing it jointly as one pattern is illustrated through both simulated (Section 8.3 on page 117) and real data (Section 8.3 on page 118). For example, if a simple word $w_1$ has a biological function when it is present jointly with a different word, $w_2$, both words would be enriched in the sequences, as both of them are required for the function to be performed. However, combining the words in one regular expression, as $w_1(A|C|G|T)^*w_2$ or $w_2(A|C|G|T)^*w_1$, increases the statistical significance, as the words would tend to be present in the sequences together. If these two words would have independent biological functions, the statistical significance of the combined regular expression would be lower. This is illustrated in Fig. 8.4 on page 116, but also with experimental data from microRNA over-expression data sets (Table 8.1 on page 118).
4. Inference of patterns

4.3 Patterns in the hidden structure of sequences

So far, the patterns I aimed at finding as being statistically significant were identified in a sequence or set of observed sequences, which are assumed to be generated from a background probabilistic model. Even though here I have been only concerned with a simple Markov chain as the background model, more complex models can be tackled, such as \( m \)-order Markov chains [155], hidden Markov models [219], and even probabilistic arithmetic automata (PAA), a model that generalizes both DFAs and HMMs [139]. However, when dealing with a model that describes some underlying hidden structure, like HMMs or PAAs, the patterns are looked for in the observed sequences.

Often, the hidden structure itself is more relevant to study than the observed sequence it generated and a natural question arises: what about patterns in the hidden structure? How can we identify them, count the number of occurrences and subsequently use it for downstream analysis? I address these questions in Chapter 9: [196] P. Tataru, A. Sand, A. Hobolth, T. Mailund, and C. N. Pedersen. Algorithms for hidden Markov models restricted to occurrences of regular expressions. Biology, 2(4):1282–1295, 2013

Within bioinformatics, HMMs have been used with a wide scope, ranging from inferring protein secondary structure to coalescent theory, and for a large portion of them, patterns in the hidden structure could provide more information

- protein secondary structure: patterns could entail neighboring secondary structures that are different;
- sequence alignments: the pattern could capture specific characteristics, such as long indels;
- coalescent theory: changes in the tree (indicating recombination events) along the sequence are often relevant (Section 3.4 on page 30);
- coding DNA: patterns corresponding to genes are the main focus.

Using the standard HMM algorithms (Section 2.2 on page 8), the number of occurrences of such patterns can be counted in a straightforward manner (as in the methods based on [114])

- run a decoding algorithm, e.g. the Viterbi algorithm;
- count the occurrences of the pattern in the prediction.

Such an approach is limited in that only a point estimate is obtained and, as shown in Section 9.2 on page 129, it can give consistently biased estimates of the true number of occurrences. A more realistic method is presented in [10], where the distribution of the number of pattern occurrences is computed using Markov chain embedding. The approach from [10] doesn’t consider the
more general patterns encoded by regular expressions. To our knowledge, [10] is the only study of patterns in the hidden sequence of HMMs.

We aimed at answering two questions related to identification of patterns in the hidden structure of a given (trained) HMM

• What is the distribution of the number of pattern occurrences in the hidden sequence for an observed sequence?

• How can we adapt the standard decoding algorithms to incorporate the number of pattern occurrences?

We addressed these questions using the embedding technique for regular expressions (Section 4.1). Combining it with HMMs, we introduced new versions of three of the standard algorithms

• the restricted forward algorithm, an extension of the forward algorithm (Section 2.2 on page 9); it computes the likelihood of the data restricted to those hidden sequences containing a specific number of pattern occurrences; this can be used to compute the distribution of the count of pattern occurrences;

• the restricted Viterbi and restricted posterior-Viterbi algorithms, which extend the Viterbi (Section 2.2 on page 11) and posterior-Viterbi algorithms, respectively; the prediction is restricted to containing a certain number of occurrences.

The posterior-Viterbi is a variant of the posterior decoding algorithm (Section 2.2 on page 11), described in Section 9.1 on page 126.

Here, I present the idea we used to build these algorithms and summarize the results of Chapter 9. In this work, I contributed to all stages, from the development of the methodology, designing and implementing the code, performing experiments and writing the manuscript.

The restricted algorithms

Let us consider an HMM as defined in Section 2.2 on page 8 and the DFA for a regular expression \( r \) over the hidden alphabet \( \mathcal{H} \), as described in Section 4.1 on page 43. Also, let \( O_r(x_{1:T}) \) be the number of matches of \( r \) in \( x_{1:T} \).

We aim at estimating \( O_r \) conditional on the observed data \( y_{1:T} \), by determining its probability distribution. Using the restricted decoding algorithms, we wish to obtain a sequence of hidden states, \( x^*_{1:T} \), for which \( O_r(x^*_{1:T}) \in [l, u] \). This interval can be set to, for example, the expected number of occurrences, which can be calculated from the distribution.

The main idea behind the restricted algorithms is to run the HMM and the DFA in parallel and keep track of the observed number of patterns at any position \( t \) in the sequence. To do so, the tables built during the forward, Viterbi and posterior-Viterbi are extended to contain the number of observed patterns and the current state in the DFA. The details on the restricted algorithms are given in Section 9.2 on page 127.
4. Inference of Patterns

Figure 4.4: Pattern occurrences. For the HMM and pattern considered in Fig. 9.1 on page 125 and Fig. 9.2 on page 126, respectively, and for a sequence of length 500 generated from the HMM, the plot shows the true (black) number of pattern occurrences, the counts contained within the Viterbi (red) and posterior-Viterbi (blue) decodings, together with the distribution calculated by the restricted forward algorithm, the interval covering 95% (gray) of the probability distribution and the expectation (green) of the distribution.

Results

As presented in detail in Section 9.2 on page 128, we performed experiments to investigate the accuracy of the distribution inferred by the restricted forward algorithm and evaluate the two restricted decoding algorithms. For this purpose, we used data generated from a hand-built HMM (Fig. 9.1 on page 125), aiming at predicting coding regions within a DNA sequence. We ran the restricted algorithms on the generated observed sequences and used the known generated hidden sequences for evaluation purposes. The pattern we chose for this simulation study was a regular expression encoding the start of a gene.

Figure 4.4 illustrates the distribution calculated from the restricted forward algorithm for one generated sequence of length 500. The figure indicates that, for this sequence, the distribution fits well with the true number of pattern occurrences and its expectation closely matches the true. The Viterbi and posterior-Viterbi provide a gross underestimate.

As shown in Fig. 9.3 on page 129, the expectation calculated from the distribution obtained from the restricted forward algorithm was a very good estimate to the true number of pattern occurrences. The Viterbi and posterior-Viterbi algorithms provided different estimates, but always underestimated the truth. From the two, the posterior-Viterbi algorithm approximated more accurately the correct number of pattern occurrences.

To assess the accuracy of the decoding algorithms, we evaluated the quality at both nucleotide and gene level, following the analysis in [28]

- nucleotide level: we compared the decoding and the true hidden sequence position by position;
4.4 SCFGs and the RNA secondary structure

- gene level: when using measures at nucleotide level, genes that are longer contribute more to the measures considered; however, it is interesting how well the genes are recovered, independent of how long they are; for this, we compared the decoding and true hidden sequence gene by gene.

Our experiments showed that the restricted Viterbi algorithm brings a great improvement to the prediction compared to the standard Viterbi algorithm (Figs. 9.5 and 9.6 on page 130), both at nucleotide and gene level. The difference in performance between the standard posterior-Viterbi and its restricted version is a bit more involved, but we did not observe the same kind of improvement. This is discussed in details in Section 9.2 on page 131.

4.4 SCFGs and the RNA secondary structure

In the following sections I present the background theory and the results of two publications. These resulted from projects developed at the Oxford Summer School on Computational Biology, and are somewhat different than the rest of my PhD work.

The central dogma of molecular biology, put simple is: DNA makes RNA, RNA makes proteins, proteins make us. During the process of creating proteins from DNA, three different types of RNA molecules are used

- the messenger RNA (mRNA), which is essentially a transcribed copy of the DNA and serves as a template for the final protein;
- the transfer RNA (tRNA), which reads the code written within the codons of the mRNA and pairs them with the corresponding amino acids to perform the translation to protein;
- the ribosomal RNA (rRNA), which is a component of the ribosom, a large and complex molecular machine that serves as the primary site of the mRNA translation to protein.

Before the discovery of other types of RNA molecules, it was thought that RNA was only involved in the process of creating proteins from DNA [54] and that all RNA molecules were contained within the mRNA, tRNA and rRNA classes. It was later discovered that some RNAs play an active role within cells. The function that an RNA molecule performs is known to depend on the three dimensional (tertiary) structure it forms. This is built upon the secondary structure, a scaffold of basepairs formed through the hydrogen bonds between the nucleotides. In turn, the secondary structure is determined by the sequence of nucleotides (primary structure) composing the RNA. Therefore, in order to determine the function of a molecule, a first necessary step is to infer the secondary structure. The advent of next-generation sequencing technologies and new methods in transcriptomics have increased the importance of RNA secondary structure prediction.
4. **Inference of Patterns**

The secondary structure is solely determined by the hydrogen bonds, which can be divided in two categories: the canonical Watson-Crick baseparing A-U and C-G, and non-canonial basepairing, among which G-U is the most common one. The computational process of predicting the RNA secondary structure starting from its sequence constitutes in determining which positions in the sequence form a basepair and which positions remain unpaired. Early attempts at secondary structure predictions simply evaluated all possible structures with respect to free energy functions [161]. Later, thermodynamic principles were used to advance free energy methods in algorithms such as UNAfold [138] and RNAfold [86]. They rely on a large number of experimentally determined parameters. Alternative approaches use stochastic context-free grammars (SCFGs) to find the most likely structure. Among them, the Pfold algorithm of Knudsen and Hein [108] and PPfold [193], its parallelized implementation, are the most successful ones.

Within an RNA secondary structure, two basepairs can be in one of three configurations: juxtaposed, nested or overlapping (Fig. 4.5). Basepairs are almost always stacked onto other basepairs, creating a series of nested basepairs, called stems (Fig. 4.6). At the end of the stems, unpaired bases form loops. Overlapping basepairs form pseudoknots, which are difficult to predict. Optimal structures with pseudoknots can be found provided that the pseudoknots satisfy a set of constraints, but finding structures with unconstrained pseudoknots is a hard problem [2]. Most RNA secondary structure prediction methods do not predict pseudoknots. Here, I only consider pseudoknot-free secondary structures, as it is well established that SCFGs cannot predict them [23]. While grammatical-based methods do exists to predict pseudoknots [30], they require more complex models than just simple SCFGs.

**Stochastic context-free grammars**

Stochastic context-free grammars (SCFGs) are probabilistic models which generalize HMMs (Section 2.2 on page 8) and the standard algorithms for SCFGs closely resemble the ones for HMMs. Protein secondary structure has been successfully predicted using HMMs [113], but due to the non-local inter-

---

**Figure 4.5:** **Configurations of basepairs.** Two basepairs can be in one of three configurations. Here, the RNA backbone is marked with a gray line and the nucleotides with red dots, while the blue arcs indicate basepairs. Each configuration is also shown above in the dot-parenthesis representation of RNA secondary structure.
4.4. SCFGs and the RNA secondary structure

![Diagram of RNA secondary structure]

Figure 4.6: **Stem and loop.** Two common elements of RNA secondary structure using dot-parenthesis and two graphical representations.

actions that occur in the RNA secondary structure, HMMs cannot be used for this purpose, and the more general SCFGs are required.

Chomsky, a computational linguist, in an attempt to understand the structure of natural languages, developed a general theory for modeling strings of symbols and laid the founding theory for grammars [34, 35]. In time, grammars became a fundamental model in theoretical computer science [90]. Context-free grammars are a special type of grammars, and in their stochastic version, they are coupled with a set of discrete probability distributions. Early use of stochastic context-free grammars for RNA secondary structure prediction includes [88, 107, 175, 177], with different underlying SCFGs designs. Rivas [174] notes that any architecture for RNA secondary structure prediction can be described in the form of a grammar.

Formally, a context-free grammar \( G \) (henceforth abbreviated to grammar) consists of

- a finite set \( N \) of non-terminal variables;
- a finite set \( V \), disjoint from \( N \), of terminal variables;
- a distinguished starting symbol \( S \in N \);
- a finite set \( P \) of production rules; each production rule replaces one non-terminal with a string of non-terminals and terminals.

A grammar is a generative model, producing strings over the alphabet \( V \). The generating process, called derivation, starts from \( S \) and sequentially applies rules from \( P \) until it reaches a string formed only of non-terminals. Within the RNA secondary structure prediction problem, \( V = \{ ., (, ) \} \), where the string produced at the end can be seen as an RNA secondary structure in the dot-parenthesis representation (Figs. 4.5 and 4.6): positions that are unpaired are marked with ‘.’ and basepairs are marked with matching parenthesis. As can be seen in Fig. 4.5, in this representation, a pseudoknot would just be confounded with nested basepairs.

One of the most well known context-free grammars for predicting RNA secondary structure has been introduced by Knudsen and Hein [107], denoted...
4. Inference of Patterns

Figure 4.7: Derivation using KH99. Left: sequence of rules (derivation) used to produce 
(\.). In the center and to the right, the corresponding parse trees of the derivation are illustrated. The tree to the right corresponds to the stochastic derivation, where each rule has a probability, denoted here in blue. Additionally, whenever a basepair or unpaired position is produced, their corresponding prior probabilities are included. Here, the RNA string is ACAUGU and for each produced terminal, the corresponding nucleotide is shown in gray.

Here KH99. The grammar has three non-terminals, \(S, L\) and \(F\), and 6 production rules, two for each non-terminal. The grammar can be represented in the following condensed form

\[
\text{KH99:} \quad S \rightarrow LS \mid L \quad L \rightarrow . \mid (F) \quad F \rightarrow LS \mid (F)
\]

Figure 4.7 shows the derivation of the string (\.\.) using KH99. This can also be represented as an ordered rooted tree, called a parse tree. A depth-first traversal of the tree will produce the generated string, (\.\.).

An SCFG is a grammar with an associated probability distribution over the production rules which start from each non-terminal. Thus, we can obtain a probability for generating a specific string by just multiplying the probabilities of the rules used in the derivation. Therefore, the SCFG describes a probability distribution over all the strings it can generate. In the case of RNA secondary structure prediction, we obtain a probability distribution over the secondary structures. To complete the model, we require prior probabilities for any of the four nucleotides being unpaired \((p_A, p_U, p_C, p_G)\) and probabilities for the sixteen corresponding possible basepairs \((p_{AU}, p_{UA}, p_{CU}, \ldots)\). For
example, let us assume that KH99 has the following distribution over rules

\[
\text{KH99: } S \rightarrow 0.5 \, LS \mid 0.5 \, L \rightarrow . \mid (F) \quad F \rightarrow 0.4 \, LS \mid 0.6 \, F
\]

Then the total probability for the derivation of (..) for the RNA string ACAUGU is

\[
0.5 \cdot 0.8 \cdot p_{AU} \cdot 0.6 \cdot p_{CG} \cdot 0.4 \cdot 0.5 \cdot 0.2 \cdot p_A \cdot 0.2 \cdot p_U,
\]

as indicated in Fig. 4.7.

For RNA secondary structure prediction, it is most convenient to write the production rules in a double-emission form, introduced in [6], where rules can only take one of the following forms

\[
U \rightarrow . \\
U \rightarrow VW \\
U \rightarrow (V)
\]

This normal form captures the fundamental features of RNA secondary structure (unpaired bases, branching and basepairs) and allows for the development of the structural motifs commonly found in RNA. This form is not restrictive, in that any context-free grammar is equivalent to one in this form. For example, KH99 can be rewritten as

\[
\text{KH99: } S \rightarrow LS \mid . \mid (F) \quad L \rightarrow . \mid (F) \quad F \rightarrow LS \mid (F)
\]

The standard algorithms for SCFGs are natural counterparts to the fundamental algorithms of HMMs: the Cocke-Younger-Kasami (CYK) algorithm [218] relates to the Viterbi algorithm (Section 2.2 on page 11), and the inside-outside (IO) algorithm [117] relates to the forward and backward algorithms (Section 2.2 on pages 9 and 10). The CYK algorithm is used to find the probability of the most likely derivation, and so backtracking can be used to find the most likely structure. The IO algorithm is used to calculate the posterior basepairing probabilities which can then be employed in a posterior decoding method, similar to how the forward and backward algorithms create the building blocks for the posterior decoding algorithm for HMMs (Section 2.2 on page 11). The secondary structure with the maximum expected number of correct positions can be calculated from the posterior probabilities via dynamic programming. Even though these algorithms were constructed for a different normal form, they can easily be adapted to the double-emission form and additional prior probabilities on unpaired and paired nucleotides.

Apart from predicting a structure given an SCFG, another question has to be dealt with: given the grammar, how do we find the best probability distribution for the rules? These are determined using the IO algorithm in an Expectation-Maximization algorithm [117], which corresponds to the Baum-Welch algorithm for HMMs (Section 2.2 on page 11).

Many different SCFG models for RNA secondary structure prediction have been implemented [44, 107, 108] and in the search for the best predictor, two different directions can be taken.
4. Inference of patterns

- build a complex grammatical model that captures the essential elements and dependencies in secondary structures;
- couple the grammatical model with other models that include additional biological and physical information.

Starting at the summer schools in Oxford, I have been involved in SCFG-based RNA secondary structure prediction using both directions mentioned above, detailed in the following two sections.

4.5 An evolutionary algorithm for SCFGs

The Pfoid algorithm [107, 108] is one of the most successful approaches using SCFGs and uses the grammar KH99, which was constructed by hand. While KH99 is effective, it seems to have been chosen relatively arbitrarily. This problem was addressed in [44], in which nine different handmade lightweight SCFGs were evaluated on a benchmark set of RNA secondary structures.

This suggested that a computational search through the large space of grammars might find stronger ones. We have performed such a computation search using an evolutionary algorithm


I will briefly present here the approach and main results of Chapter 10. This publication is the result of my first attendance at the Oxford Summer School in Computational Biology in 2010, where I contributed to the development of the methodology, designing and writing the code and drafting of the manuscript.

**Evolutionary algorithm design**

The way that the evolutionary algorithm searches the space is determined by the design of the initial population of grammars, mutation, breeding and selection procedure. The algorithm starts with an initial population and in each generation, it grows the current population by introducing a number of newly mutated or bred grammars, which are then pared back to a fixed population size by selection. During selection, the new grammars are first trained and then tested to evaluate their fitness (quality of prediction). This has been done both by means of CYK and IO.

The initial population we used was composed of sixteen small grammars. The small grammars subsequently became larger through the mutation and breeding rules, which grow the number of non-terminals and production rules.

Mutations constitute the majority of movement through the search space. The form of mutation used is very basic, but allows many structural features to develop over generations. More complex mutation is clearly possible, however
we have found that the model we used gave sufficient mobility in the search space. We allowed for changing, copying, deleting and inserting new non-terminals and production rules.

In our breeding model we formed a grammar which can produce all derivations of its parent grammars by merging the two start non-terminals into a single one. This breeding model was chosen to keep the size of the grammar relatively small, whilst allowing expression of both bred grammars to be present in derivations.

During selection, we first trained and then tested the grammars. We determined the probability of elimination of a grammar by the inverse of some fitness measure. Fitness functions we used included sensitivity and positive predictive value (PPV), following the definition in [44].

Results

Across all our experiments, over 300,000 grammars were searched, among which we found several strong ones using both CYK and IO. Our results showed that very different grammars perform equally well. We have shown that there exist many grammars which perform similarly to KH99. Tables 10.2 and 10.3 on page 140 show the performance of some of the grammars found and other well established prediction methods.

One question we addressed was whether the grammars we identified produced the same secondary structure which was a good proxy to the true structure. Our results (Fig. 10.3 on page 142) indicated that the grammars found produced different kinds of structures, which were good representations of the true RNA secondary structure.

Overall, the grammars found in the evolutionary search performed well. Determining which was best depended on the measure of strength of prediction, whether the size of the grammar was a concern, and so on. However, it was clear that a selection of effective grammars has been found, with performance similar to KH99, though we have not succeeded in evolving one that can clearly outperform KH99.

4.6 Oxfold: kinetics and evolutionary information

The inclusion in prediction methods of additional biological and physical information about RNA sequences and the process of forming the structure has been proved successful in improving the quality of the predicted structure. This has been pioneered by Knudsen and Hein [107, 108] in a comparative structure prediction approach, where given an alignment of homologous RNA sequences, the scope was to predict a consensus structure.

Homologous RNA which share a common secondary structure might have a low sequence similarity [46] but show, due to the functional constraints, signs of compensatory mutations which maintain the basepairing complementarity
4. Inference of patterns

[107]. This makes RNA sequence analysis more complicated and difficult than protein or DNA sequence analysis, but also enables inference of a common structure for homologous sequences where the compensatory mutations indicate a conserved basepair [46].

Knudsen and Hein have coupled an SCFG with an evolutionary model, where nucleotides and pairs of nucleotides evolve on a phylogenetic tree relating the sequences. Their grammar, KH99, as discussed before, was proved to be a very simple and efficient predictor. The addition of the evolutionary information greatly improved the accuracy. This information is used to calculate the prior probabilities (Fig. 4.7) of columns being unpaired or two columns forming a basepair. The prior probabilities are simply the likelihood of the column / pairs of columns, given the phylogenetic tree and an evolutionary model. In the models used, compensatory mutations which preserve the structure of the RNA can be regarded as a single mutation instead of two.

Another approach to coupling models with physical information is to consider the kinetics of RNA folding. Craig et al. [39] studied the kinetics of RNA folding and determined the speed at which helices form. Helices are generally the tertiary structure that a stem (Fig. 4.6) folds into. Models where the folding kinetics are incorporated do exist (e.g. [41, 214]).

One would expect that, by relying on complementary sources of biological and physical information, the prediction accuracy would increase. We addressed the issue of combining folding kinetics with an evolutionary model within an SCFG and consequently built Oxfold


a model for predicting a consensus structure for a given, fixed alignment of RNA sequences. I will briefly present here the approach and main results of Chapter 11. This publication is the result of my return to the Oxford Summer School in Computational Biology in 2012, where, as an instructor, I co-supervised three students. Due to the results summarized in Section 4.5, we have have used the KH99 grammar throughout.

Helix formation and a distance function

The results of Craig et al. [39] motivated emulating the kinetics of RNA folding in a simplified way by forming helices iteratively. During each iteration, a suitable candidate basepair is identified, and then a helix is formed which contains that basepair. The rest of the basepairs in the helix formation are chosen such that there is strong evidence that the columns form a basepair: the posterior probability of the basepair has to be higher than the posterior
4.7 Conclusions and future work

In this chapter I presented two topics that I have contributed to during my PhD. The first one covers pattern detection, such as simple words, in sequences, either observed or hidden. I have presented how the probability of observing a certain number of patterns in a sequence can be calculated, using the embedding Markov technique. Coupling it with deterministic finite automata allows for flexible patterns in the form of regular expressions. Building on this, I have shown how random walk theory can be used to determine whether these probabilities for a list of ranked sequences form clusters or are randomly distributed. This has been incorporated in a tool, Regmex. Additionally, I have also introduced new algorithms for hidden Markov models to compute the distribution and account for pattern occurrence within the hidden structure.
4. Inference of patterns

The second topic handled RNA secondary structure prediction by means of stochastic context-free grammars. After introducing SCFGs and their application to RNA secondary structure prediction, I presented solutions to two problems. I firstly addressed finding a good SCFG predictor by means of an evolutionary algorithm. Given an SCFG, I also presented how to improve its prediction using the kinetics of RNA folding coupled with evolutionary information contained within an RNA alignment.

The work introduced here originated mainly from the first half of my PhD, and most of it arose from collaborations. As such, the two topics covered in this chapter have not constituted the focus of the second half of my PhD and will probably not be part of my future work. Nevertheless, the presented tools are valuable in the field of bioinformatics and I consider it useful to have acquired this knowledge. The two problems addressed could potentially be extended further as follows.

First of all, within the problem of pattern finding in sequences, it is apparent that exhaustive searches over all possible \( k \)-mers is of particular importance. Therefore, I believe that the next improvement of Regmex should be in handling this problem by using the suffix tree approach presented by Leibovich et al. [119]. More interestingly, one could try investigating an efficient search that would be suitable for more flexible patterns.

With regard to the hidden Markov model algorithmic development, several directions can be taken. In the presented theory, the HMM was implicitly assumed to be trained. However, in practice this is often not the case, and HMM based analysis typically starts with the training of the HMM. It would therefore be of interest to build a restricted backward algorithm which, jointly with the restricted forward algorithm, could be used in a Baum-Welch training approach. This could potentially lead to a better training model resulting in an HMM that represents the observed data more accurately. Alternatively, one could further adjust the restricted decoding algorithms by using more information from the distribution of the pattern occurrences, rather than just the expectation. The restricted forward algorithm can be used to calculate the waiting time until the \( k \)th occurrence of the pattern. Therefore, the decoding algorithms could be altered to not only contain the correct number of pattern occurrences, but also their most likely placement within the sequence. This could be useful in problems, such as gene prediction, where the accurate location of specific patterns in the decoding is of importance.

Within the topics of RNA secondary structure prediction using SCFGs, pseudoknots are probably the most important issue. One could potentially investigate coupling grammatical models built for predicting pseudoknotted structures [30] with the kinetics of RNA folding and the evolutionary information from RNA alignments. More interestingly, one could bridge the two topics presented here, by adapting the SCFG algorithms in a similar manner as the HMM restricted algorithms were built. SCFGs are a generalization of HMMs and one could use regular expressions matching various elements
4.7. Conclusions and future work

within an RNA secondary structure. I note here, however, that describing a helix of variable length with a regular expression, and consequently, a DFA, is not possible. For this, a pushdown automaton would be required. In fact, pushdown automata are equivalent to context-free grammars, in that for every context-free grammar, there exists a pushdown automaton that recognizes the language produced by the grammar [90]. The presented approach for the restricted algorithms could be extended to the more general context-free languages. However, this would probably come at the cost of memory consumption and speed, due to the addition of the stack in the pushdown automata.
Part II

Publications and Manuscripts
Modeling allele frequency data under a Wright-Fisher model of drift, mutation and selection: the Beta distribution approach

Paula Tataru¹ Thomas Bataillon¹ Asger Hobolth¹

The manuscript presented in this chapter contains work in progress, which I presented at the workshops *Biological sequence analysis and probabilistic models*, Oxford, UK, July 2014 and *Mathematics and genetics of selection and adaptation*, Aarhus, Denmark, October 2014. I plan to resume this work after completing my PhD, during my first postdoctoral position.


Abstract

Advances in sequencing technologies have revolutionized the collection of genomic data. These data contain information about mutation rates and selection coefficients, but also demographic history of populations. Researchers have been successfully applying the coalescent process to infer these, but such approaches are limited to a rather small number of individuals. More recently, the Wright-Fisher model has been increasingly used for data analysis. However, exact results for the behavior of the allele frequency as a function of time are not available in closed analytic form. To handle this problem, researchers have either adopted the diffusion approximation or relied on a mathematically convenient distribution, such as the beta distribution, to approximate the distribution of allele frequency. The beta distribution has so far been limited to only modeling drift and mutation. Its main drawback is the behavior at the boundaries. Due to its continuous nature, probabilities at the boundaries will always be zero. Nonetheless, under the Wright-Fisher model, these probabilities can be positive, corresponding to the allele being either lost or fixed. Here we introduce the beta with spikes, an extension of the beta approach. We present how to incorporate general diploid selection in this framework. We additionally adjust the distribution by explicitly modeling the loss and fixation probabilities as spikes at the boundaries. We show that these spikes improve the quality of the approximation and compare the beta with spikes approach to a recently proposed numerical diffusion, both in terms of accuracy and inference. We illustrate how the beta with spikes can be used for inferring split times between populations, with comparable performance to existing diffusion-based methods.

¹Bioinformatics Research Centre, Aarhus University, Aarhus, Denmark
5. The Beta approach to a Wright-Fisher model

5.1 Introduction

Advances in sequencing technologies have revolutionized the collection of genomic data, increasing both the volume and quality of available sequenced individuals from a large variety of populations and species. These data, which may involve tens of thousands of single nucleotide polymorphisms (SNPs), contain information about both the mutation and selection pressures acting on the DNA, but also the demographic history of the populations, such as split times, admixtures and migration events. There has been great focus in the past few years on inferring such information, and two broad categories of inference methods can be identified: coalescent-based backwards in time models or forwards in time Wright-Fisher models. These two models are in a dual relationship, in that the Wright-Fisher model describes, in discrete non-overlapping generations, how a population evolves forward in time, while the coalescent process is a backwards in time approximation to the Wright-Fisher model, assuming a large population size and continuous time.

The coalescent process explicitly models the genealogical relationships between the sampled individuals in the present and going back to the most recent common ancestor. The full coalescent with recombination process is intractable and therefore various approximations have been developed, mainly relying on the so called sequentially Markov coalescent [143]. These are generally restricted to a low number of individuals, but have been successfully applied to inferring population sizes [122, 182] and migration rates [180, 192]. A recent approach [169], which aims at producing the posterior distribution of the full ancestral recombination graph, can scale up to dozens of individuals. This approach provides means for, among others, characterizing the influence of natural selection. However, Rasmussen et al. [169] do not explicitly incorporate selection in the model, but rather rely on the inferred patterns of ancestry. Even though theoretical results exist based on the coalescent with selection [106, 121, 162], it can be computationally very challenging to use such models for inference. In spite of the progress achieved in using the coalescent process, such approaches are currently only applicable to a rather low number of individuals due to the exponentially growing state space of possible coalescence trees.

Without keeping track of the relationship between the individuals, methods relying on the Wright-Fisher model describe, at each independent locus, the population allele frequency as it evolves over time. One advantage of tracking solely the allele frequencies is that a larger sample size will not pose a high computational burden, but merely provide a better estimate of the true allele frequency in the studied population. This allows for the use of much larger data sets than the coalescent-based methods can typically handle. The main difficulty in using the Wright-Fisher model is that the allele frequency distribution as a function of time, conditional on a starting frequency, is not available in closed analytic form. So far, this has been mainly handled by either relying on the diffusion limit of the Wright-Fisher model, or by using a moment-based approximation.

The diffusion limit has been receiving increasing attention throughout the last years. It has been applied successfully to inferring split times between populations [67] and selection coefficients from temporal allele frequency data [19, 133]. Even though numerical [221] and analytic [187] approaches to calculating the allele frequency distribution do exist, no closed analytic form is known for the Kolmogorov diffusion equation. In practice, the challenge of applying diffusion to inference problems lies in choosing accurate, yet not too slow, numerical solutions to the Kolmogorov equation.

The moment based approximations are less ambitious in that, instead of aiming to characterize the full true distribution, they rely on fitting a mathematically convenient distribution by equating the first two moments. Some distributions are fully described by their mean and variance. Therefore, to use such distributions as an approximation, it is sufficient to only evaluate the first two moments of the true distribution. This type of approaches are much simpler than the diffusion and provide, typically, analytic results. Even though they are not principled, they have been proved, nevertheless, powerful in assessing population differentiation and isolation [152], identifying local adaptation [38] and inferring population splits [186] and mixtures [160]. Within the moment based approaches, the distributions used so far are the normal and beta distributions.

The normal distribution was motivated as the result of the diffusion limit when drift is small (at most on a timescale of the same order as the population size) [160]. Its main downside is that its support is on the whole real line, while allele frequencies lie between 0 and 1. The distribution has to be truncated [38, 152], an operation which leads
5.2 Methods

We consider a diploid random mating population of size $2N$ and a biallelic locus. We wish to model the allele frequency of one of the alleles, $A$. Let $Z_t$ be the random variable giving the number of $A$ alleles at generation $t$ and $X_t = Z_t/2N$. The evolution of $Z_t$ is shaped by drift, together with a deterministic evolutionary pressure, denoted here as $g(x)$, a polynomial in the allele frequency $x$. The evolutionary pressure can take the form of mutation, migration and selection. From the Wright-Fisher model, conditional on $Z_t = z_t (= 2Nx_t)$, $Z_{t+1}$ follows a binomial distribution

$$Z_{t+1} \mid Z_t = z_t \sim \text{Bin}(2N, g(x_t)) .$$

We are interested in the distribution of $X_t$ conditional on $X_0 = x_0$, as a function of the gener-
5. The Beta approach to a Wright-Fisher model

We characterize this distribution using its mean $\mathbb{E}[X_t]$, variance $\text{Var}(X_t)$, loss $\mathbb{P}(X_t = 0)$ and fixation $\mathbb{P}(X_t = 1)$ probabilities. For simpler notation, we leave out the explicit conditioning on $X_0 = x_0$ and implicit condition on population size and evolutionary pressure.

If mutations happen with probabilities $u$ and $v$ per generation

$$A \xrightarrow{u,v} a,$$

the evolutionary pressure takes the form

$$g(x) = (1 - u - v) x + v.$$

Pure drift can be viewed as a special case of mutation, where $u = v = 0$. Due to the linearity in $x$, the mean and variance are known in closed form [185]. If however, selection is present, the mean and variance can no longer be obtained analytically, as the polynomial $g(x)$ has a higher degree.

If the $A$ allele has selection coefficient $s$ and dominance parameter $h$, with the following relative fitness of the three genotypes

<table>
<thead>
<tr>
<th>Genotype</th>
<th>AA</th>
<th>Aa</th>
<th>aa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative fitness</td>
<td>$1 + s$</td>
<td>$1 + sh$</td>
<td>1</td>
</tr>
</tbody>
</table>

$g(x)$ becomes a polynomial of degree at most three

$$g(x) = (1 - u - v) S(x) + v,$$

where

$$S(x) = x + s x(1 - x) [h + (1 - 2h)x],$$

describes the action of selection on the allele [58].

The beta approach

Under the beta approach, we approximate the true distribution of $X_t$ with a beta distribution

$$f(x; t) = \frac{x^{\alpha_t-1}(1-x)^{\beta_t-1}}{B(\alpha_t, \beta_t)},$$

where $B(\alpha, \beta)$ is the beta function. To fit this distribution, i.e. determine $\alpha_t$ and $\beta_t$, we need to calculate the mean and variance of $X_t$. We rely on the laws of total expectation and variance. We first note that, from the binomial distribution of $Z_{t+1}$ conditional on $Z_t$, we obtain the mean and variance of $X_{t+1}$ conditional on $X_t$ to be

$$\mathbb{E}[X_{t+1} | X_t] = g(X_t),$$

$$\text{Var}(X_{t+1} | X_t) = \frac{1}{2N} g(X_t) (1 - g(X_t)).$$

Using this, we have that

$$\begin{align*}
\mathbb{E}[X_{t+1}] &= \mathbb{E}[\mathbb{E}[X_{t+1} | X_t]] \\
&= \mathbb{E}[g(X_t)], \\
\text{Var}(X_{t+1}) &= \mathbb{E}[\text{Var}(X_{t+1} | X_t)] \\
&= \mathbb{E} \left[ \frac{1}{2N} g(X_t) (1 - g(X_t)) \right] + \text{Var}(g(X_t)).
\end{align*}$$

When $g(x)$ is linear in $x$, such as in the mutation case, this recursion can be solved analytically [185], and the mean and variance are, for $u + v \neq 0$,

$$\begin{align*}
\mathbb{E}[X_t] &= (1 - u - v)^t \left( x_0 - \frac{v}{u + v} \right) + \frac{v}{u + v}, \\
\text{Var}(X_t) &= \frac{u v}{(u + v)^2} \left( 1 - (1 - u - v)^t \left( 1 - \frac{1}{2N} \right)^t \right) \\
&\quad \cdot \left( \frac{1}{2N} - (1 - u - v)^t \left( 1 - \frac{1}{2N} \right)^t \right) \\
&\quad + (1 - u - v)^t \left( x_0 - \frac{v}{u + v} \right) \frac{u - v}{u + v}.
\end{align*}$$

When the degree of $g(x)$ is higher, we use the approximated $f(x; t)$ together with the definition of expectation to determine $\mathbb{E}[g(X_t)^i]$ for $i = 1, 2$ and complete the recursion

$$\begin{align*}
\mathbb{E}[g(X_t)^i] &= \int_0^1 g(x)^i \cdot \frac{x^{\alpha_t-1}(1-x)^{\beta_t-1}}{B(\alpha_t, \beta_t)} \, dx. \quad (5.3)
\end{align*}$$

This integral can be calculated analytically due to the polynomial nature of $g(x)$. The resulting algorithm is summarized in Table 5.1.
The beta with spikes approach

When considering the spikes at the boundaries, we approximate the true distribution of $X_t$ with a beta distribution for the interval $(0,1)$, adjusted with the loss and fixation probabilities. The full density of $X_t$ then takes the form

$$f^*(x; t) = P(X_t = 0) \cdot \delta(x) + P(X_t = 1) \cdot \delta(1-x) + P(X_t \not\in \{0,1\}) \cdot \frac{x^{\alpha_t-1} \cdot (1-x)^{\beta_t-1}}{B(\alpha_t, \beta_t)},$$

where $\delta(x)$ is the Dirac delta function. To determine $\alpha_t^*$ and $\beta_t^*$, we now need the mean and variance conditional on polymorphism ($X_t \not\in \{0,1\}$), as given by

$$E[X_t \mid X_t \not\in \{0,1\}] = \frac{E[X_t] - P(X_t = 1)}{P(X_t \not\in \{0,1\})},$$

$$\text{Var}(X_t \mid X_t \not\in \{0,1\}) = \frac{\text{Var}(X_t) + E[X_t]^2 - P(X_t = 1)}{P(X_t \not\in \{0,1\})} - E[X_t \mid X_t \not\in \{0,1\}]^2.\ (5.4)$$

These depend on the mean, variance, loss and fixation probabilities. We have noted previously that the mean and variance are available in analytic form when $g(x)$ is linear in $x$. When $g(x)$ has a higher degree, we need to include the two spikes in the calculation of $E[g(X_t)^2]$ for $i = 1, 2$, as follows

$$E[g(X_t)^2] = P(X_t = 0) \cdot v^2 + P(X_t = 1) \cdot (1-u)^2 + P(X_t \not\in \{0,1\}) \cdot \int_0^1 g(x)^2 \cdot \frac{x^{\alpha_t-1} \cdot (1-x)^{\beta_t-1}}{B(\alpha_t, \beta_t)} \, dx,$$

where we used that $g(0) = v$ and $g(1) = 1-u$.

To calculate the loss and fixation probabilities, we rely on the law of total probability. We use the following approximation, which holds when mutation/selection are weak

$$1 - g(x) \approx (1-u-v)(1-x),$$

$$g(x) \approx (1-u) \cdot x.$$

We then obtain

$$P(X_{t+1} = 0) \approx P(X_t = 0) \cdot (1-v)^2 + P(X_t = 1) \cdot u^2 + P(X_t \not\in \{0,1\}) \cdot (1-u-v)^2 \cdot \frac{B(\alpha_t^*, \beta_t^*)}{B(\alpha_t^* + 2N, \beta_t^*)},\ (5.5)$$

$$P(X_{t+1} = 1) \approx P(X_t = 0) \cdot v^2 + P(X_t = 1) \cdot (1-u)^2 + P(X_t \not\in \{0,1\}) \cdot (1-u-v)^2 \cdot \frac{B(\alpha_t^* + 2N, \beta_t^*)}{B(\alpha_t^*, \beta_t^*)},\ (5.6)$$

The algorithm is summarized in Table 5.1.

---

### Table 5.1: The beta and beta with spikes approaches.

<table>
<thead>
<tr>
<th>The beta approach</th>
<th>The beta with spikes approach</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> $x_0, t, N, g(x)$</td>
<td><strong>Input:</strong> $x_0, t, N, g(x)$</td>
</tr>
<tr>
<td><strong>Output:</strong> $\alpha_t, \beta_t$</td>
<td><strong>Output:</strong> $P(X_t = 0), P(X_t = 1), \alpha_t^<em>, \beta_t^</em>$</td>
</tr>
<tr>
<td><strong>Reursion:</strong> for $i = 1 \ldots t$</td>
<td><strong>Reursion:</strong> for $i = 1 \ldots t$</td>
</tr>
<tr>
<td>1. calculate $E[X_t], \text{Var}(X_t)$ using Eq. (5.1) and Eq. (5.3)</td>
<td>1. calculate $E[X_t], \text{Var}(X_t)$ using Eq. (5.1) and Eq. (5.5)</td>
</tr>
<tr>
<td>2. calculate $\alpha_t, \beta_t$ from $E[X_t]$ and $\text{Var}(X_t)$.</td>
<td>2. calculate $P(X_t = 0), P(X_t = 1)$ using Eq. (5.6)</td>
</tr>
<tr>
<td>3. calculate $E[X_t \mid X_t \not\in {0,1}]$ using Eq. (5.4)</td>
<td>3. calculate $E[X_t \mid X_t \not\in {0,1}]$ using Eq. (5.4)</td>
</tr>
<tr>
<td>4. calculate $\alpha_t^<em>, \beta_t^</em>$ from $E[X_t \mid X_t \not\in {0,1}]$ and $\text{Var}(X_t \mid X_t \not\in {0,1})$.</td>
<td>4. calculate $\alpha_t^<em>, \beta_t^</em>$ from $E[X_t \mid X_t \not\in {0,1}]$ and $\text{Var}(X_t \mid X_t \not\in {0,1})$.</td>
</tr>
</tbody>
</table>
5. The Beta approach to a Wright-Fisher model

\[ X_{ia} \]

\[ t_{ia} = 0.2 \]

\[ t_{ab} = 0.133 \]

\[ X_{ib} \]

\[ t_{ib} = 0.133 \]

\[ t_{a3} = 0.2 \]

\[ t_{b1} = 0.1 \]

\[ t_{b2} = 0.133 \]

Figure 5.2: History of three populations. The ancestral population \( a \) splits in population 3 and population \( b \), which further splits in populations 2 and 1. The data, for each SNP \( i \), consists of the sample size, \( n_{ij} \), and allele count \( z_{ij} \). The likelihood of the data is an integral over the true population frequencies \( X_{ij} \) and ancestral frequencies \( X_{ia}, X_{ib} \). We assume that the frequency \( X_{ia} \) at the root follows a uniform distribution. The branch lengths are given as \( t_{ij} \).

Scaling of parameters

The diffusion limit to the Wright-Fisher relies on the assumptions that

- the population size \( N \) is large;
- the mutation probabilities and selection coefficients are small (\( \ll 1 \)).

Under these assumptions, the time, mutation probabilities and selection coefficients can be scaled without affecting the final resulting distribution. The important factors are the ratio between the time and the population size \( N \), and the product between the mutation probabilities and selection rate, \( 4Nu, 4Nv \) and \( 4Ns \). That is, the allele frequency distribution, as calculated under the diffusion approximation, is the same for \( N, t, u, v \) and \( s \) as for \( cN, ct, u/c, v/c \) and \( s/c \), for any constant \( c \). The same property holds for the beta distribution when the mean and variance can be obtained in analytic form, i.e. \( g(x) \) is linear in \( x \). To illustrate this, let us consider the case where \( u = v \). We set \( \tau = t/2N \) and \( \theta = 4Nu \). As \( N \) is assumed to be large and \( u \) is assumed to be small, we have that

\[
\left(1 - \frac{1}{2N}\right)^{t} \approx e^{-\tau},
\]

\[
(1 - u - v)^{t} \approx e^{-i\theta \tau},
\]

\[
2N - (1 - u - v)^{t}(2N - 1) \approx 1 + 2\theta .
\]

Using these approximations in Eq. (5.2), we obtain

\[
E [X_{\tau}] = e^{-\theta \tau} \left( x_{0} - \frac{1}{2} \right) + \frac{1}{2} ,
\]

\[
\text{Var} (X_{\tau}) = \frac{1}{4(1 + 2\theta)} \left( 1 - e^{-\left(1+2\theta\right)\tau} \right)
\]

\[
- e^{-2\theta \tau} \left( x_{0} - \frac{1}{2} \right) \cdot \left( 1 - e^{-\tau} \right) .
\]

In the case of beta with spike and/or when \( g(x) \) is no longer linear, no analytic predictions could be made regarding whether the same property will hold.

Inference of split times

The beta with spikes approximation provides means for calculating the distribution of the allele frequency at time \( t \), conditional on a starting frequency at time \( 0 \). This can be used for inferring split times of populations. Populations are represented as descendants of a single common ancestral population, which has encountered a number of splits. We assume that after each split, the new populations evolve in isolation (no migration) under pure drift. A rooted tree (Fig. 5.2) can be used to describe the history, where the common ancestral population is represented as the root and the populations in the present are in the leaves. The data \( D \) for \( M \) populations consists of \( k \) independent SNPs: the (arbitrarily defined) reference \( (A) \) allele count \( z_{ij} \) for a sample size \( n_{ij} \) \((0 \leq z_{ij} \leq n_{ij})\) for each locus \( 1 \leq i \leq k \) and population \( 1 \leq j \leq M \).

Figure 5.2 depicts the history of three populations. The ancestral population \( a \) splits in population 3 and population \( b \), which further splits in populations 2 and 1.

Assuming Hardy-Weinberg equilibrium, given the allele frequency at population level, \( X_{ij} \), the probability of observing \( z_{ij} \) \( A \) alleles in a sample of size \( n_{ij} \) is given by the binomial distribution

\[
P (z_{ij} \mid n_{ij}, X_{ij}) = \binom{n_{ij}}{z_{ij}} X_{ij}^{z_{ij}} (1 - X_{ij})^{n_{ij} - z_{ij}} .
\]

However, the \( X_{ij} \)’s are unknown, and therefore the likelihood is an integral over the joint distribution
We computed the Hellinger distance between the true distribution and the approximated distribution. The Hellinger distance is defined, for two discrete probability distributions $P = (p_1, \ldots, p_k)$ and $Q = (q_1, \ldots, q_k)$, as

$$h(P, Q) = \frac{1}{\sqrt{2}} \sum_{i=1}^{k} (\sqrt{p_i} - \sqrt{q_i})^2 .$$

The Hellinger distance lies between 0 and 1, with 0 indicating perfect match between the two distributions, while the value of 1 is achieved when $P$ assigns probability zero to every set to which $Q$ assigns a positive probability, and vice versa.

Figure 5.3 shows the Hellinger distance for the three approximations. The first row corresponds to pure drift, for scaled times up to 9.0, so that the allele frequency reaches the limiting distribution. The remaining rows show the distance for mutation/selection set-ups with scaled times up to 0.2, which roughly corresponds to the time since the split of the human sub-populations. Additionally, Fig. 5.3 contains in the last column the Hellinger distance between the true distributions for the considered set-ups and pure drift. This shows, as expected, that the distributions under the Wright-Fisher model are increasingly different from pure drift, for increasing values of mutation rates and selection coefficients. This comparison further illustrates that, for the considered time scale, the allele frequency distribution when mutation is low ($4Nu = 10^{-3}$, on the order of the human mutation rate), is nearly indistinguishable from the resulting distribution under pure drift. This is yet another argument that supports the approaches, such as [67], where human data is analyzed without modeling the occurrence of new mutations, but instead assuming that all observed polymorphism originates from the ancestral population.

### 5.3 Results

We have implemented in Python the beta and beta with spikes approximations, along with (for comparison purposes) the numerical solution to the diffusion equation as given by Zhao et al. [221]. The method of Zhao et al. [221] requires two user-defined parameters, which affect its performance. In the following, we have set these parameters such that the method has a good accuracy.

#### Approximation quality

For a population of size $2N = 200$ and a range of starting frequencies $x_0$, times $t$, mutation rates $u = v$ and selection coefficients $s$ (with $h = 0.5$), we computed

- the true underlying allele frequency distribution from the Wright-Fisher model;
- the beta approximation;
- the beta with spikes approximation;
- the numerical solution to diffusion [221].

As the true distribution is discrete, while the beta-based approximations yield continuous distributions, we integrated frequencies in bins centered around $i/2N$, of the form $[(i - 0.5)/2N, (i + 0.5)/(2N)]$, where $0 < i < 2N$, and $[0, 0.5/2N]$ and $[1 - 0.5/2N, 1]$ at the boundaries.

To characterize the quality of these approximations, we calculated the Hellinger distance between the true and approximated distributions. The Hellinger distance is defined, for two discrete probability distributions $P = (p_1, \ldots, p_k)$ and $Q = (q_1, \ldots, q_k)$, as

$$h(P, Q) = \frac{1}{\sqrt{2}} \sum_{i=1}^{k} (\sqrt{p_i} - \sqrt{q_i})^2 .$$

The Hellinger distance lies between 0 and 1, with 0 indicating perfect match between the two distributions, while the value of 1 is achieved when $P$ assigns probability zero to every set to which $Q$ assigns a positive probability, and vice versa.

Figure 5.3 shows the Hellinger distance for the three approximations. The first row corresponds to pure drift, for scaled times up to 9.0, so that the allele frequency reaches the limiting distribution. The remaining rows show the distance for mutation/selection set-ups with scaled times up to 0.2, which roughly corresponds to the time since the split of the human sub-populations. Additionally, Fig. 5.3 contains in the last column the Hellinger distance between the true distributions for the considered set-ups and pure drift. This shows, as expected, that the distributions under the Wright-Fisher model are increasingly different from pure drift, for increasing values of mutation rates and selection coefficients. This comparison further illustrates that, for the considered time scale, the allele frequency distribution when mutation is low ($4Nu = 10^{-3}$, on the order of the human mutation rate), is nearly indistinguishable from the resulting distribution under pure drift. This is yet another argument that supports the approaches, such as [67], where human data is analyzed without modeling the occurrence of new mutations, but instead assuming that all observed polymorphism originates from the ancestral population.
5. The Beta approach to a Wright-Fisher model

Figure 5.3: Approximation quality. The heatmaps show the Hellinger distance between various approximations and the true allele frequency distribution, for a range of starting frequencies $x_0$ and scaled time in generations, $t/N$. Each row corresponds to a different set-up: pure drift (first row), mutation (with $u = v$, rows 2–4) and selection (with $u = v = 0$ and $h = 0.5$, rows 5–7). The last column shows the Hellinger distance between the true allele frequency distribution under pure drift and the set-up for the corresponding row.
5.3. Results

Figure 5.4: Fit of the beta with spikes. The figure shows the beta (solid red), beta with spikes (dashed green) and the true allele frequency distribution (gray) for nine different set-ups represented in Fig. 5.3. In all cases, $x_0 = 0.2$. The plot in the bottom of the middle column, with $t/2N = 0.115$, $4Ns = 10$, gives the largest Hellinger distance for the beta with spikes, as shown in Fig. 5.3.

Figure 5.3 shows that, under pure drift, all three approximations converge to a distance of 0 when approaching the stationary distribution of the allele frequency. Diffusion converges the fastest, followed by the beta with spikes and finally by the beta approach. Overall, it is apparent from Fig. 5.3 that the beta with spikes, even though it has larger distances compared to diffusion, provides a better approximation than beta, for the whole range of considered set-ups. Another clear trend is that, for a given method, the accuracy is comparable across the different set-ups. Under the high mutation set-up ($4Nu = 1$), the beta approach approximates the true distribution much better than otherwise. This is a consequence of the large symmetric mutation, when the majority of the probability mass is concentrated around 0.5. The resulting distribution is easily modeled by the beta, since the loss and fixation probabilities are effectively zero.

Figure 5.4 illustrates the fit of the beta and beta with spikes for nine different set-ups. The beta with spikes has the largest Hellinger distance in Fig. 5.3 when $x_0 = 0.2$, $t/2N = 0.115$, $u = v = 0$, $4Ns = 10$ and $h = 0.5$, which is plotted in the bottom of the middle column in Fig. 5.4. When the true distribution concentrates on intermediary frequencies, with zero probability at the boundaries, both beta and beta with spikes provide a good approximation. However, as $t$ increases and the boundaries accumulate non-zero probability, as can be seen, particularly, in the right column, the beta cannot approximate accurately the true distribution. The beta with spikes provides a much better approximation, with a tendency here to underestimate the loss probability. A better estimate of the probabilities at the boundaries might lead to an overall better quality of the beta with spikes.

Scaling of parameters

To investigate the effect of parameters scaling for the allele frequency distribution, we calculated the probability of the allele being lost under pure drift, $4Nu = 5 \times 10^{-2}$ and $4Ns = 10$, corresponding to the distributions illustrated in Fig. 5.4. We used two different population sizes, $N = 100$, as in Fig. 5.4, and $N = 200$. We plotted the probability as a function of the scaled time in Fig. 5.5.
5. The Beta approach to a Wright-Fisher model

It is clear that for the population sizes we investigated, the loss probability is dependent on the scaling. The same holds for the beta with spikes.

Inference of split times

Obtaining a reasonable approximation to the true allele frequency distribution opens multiple possibilities for inference problems. We illustrate here the power of the beta with spikes to infer split times between populations. To do so, we have simulated data from the Wright-Fisher model with pure drift on the tree from Fig. 5.2, with $X_{ia}$ sampled from a uniform distribution. We replicated 40 datasets consisting of $n_{ij} = 100$ samples (50 diploid individuals) in each population, with $k = 5000$ loci. For each dataset we estimated the scaled branch lengths by numerically optimizing the likelihood of the data on the tree. The likelihood was calculated using Felsenstein’s peeling algorithm [61], as described in the Methods section. For comparison, we have inferred the branch lengths using both the beta with spikes and our implementation of [221]. We additionally ran the Kim Tree software [67], which infers split times between populations under pure drift, relying on diffusion and using a Bayesian approach.

The inference results are shown in Fig. 5.6. For each of the four branches, we have plotted the true branch length and the estimates obtained from the three methods. The beta with spikes provides a reasonable estimation of all four parameters, showing a tendency of overestimation. In comparison, Kim Tree performs very well for two of the branches, but underestimates the value of the remaining two. We were unable to recover the true branch lengths using our implementation of [221]. This is contrary to our expectation, given the high quality of the diffusion (Fig. 5.3).

To obtain an overall average performance of the methods, we have calculated $|1 - \hat{t}_{lj}/t_{lj}|$, the absolute value of the normalized difference between the inferred lengths $\hat{t}_{lj}$ and the true values $t_{lj}$, for all four branches and all 40 replicates. These are plotted in Fig. 5.6 and indicate that the beta with spikes has an average performance very close to Kim Tree, with a somewhat smaller variance and mean.

5.4 Discussion

We have presented a new approximation, the beta with spikes, to the allele frequency distribution as a function of time, conditional on a starting frequency, as described by the Wright-Fisher model. Our work extends the previously proposed approximation based on the beta distribution in two ways

- compared to existing results, which only considered drift and mutation [185, 186], we included general diploid selection;
- we additionally adjusted the beta distribution with two spikes at the boundaries, to account for the potentially positive probabilities of loosing or fixing the allele.

We showed how this adjustment increases the quality of the beta with spikes relative to the standard beta approach. Even though the beta with spikes is slightly inferior in accuracy compared to the numerical solution of the diffusion limit of the Wright-Fisher model [221], we showed here that it can be efficiently used in recovering the split times between populations under pure drift, with a comparable performance to Kim Tree [67], which
5.5. Future work

Figure 5.6: **Inference of split times.** The figure on the top shows the inferred lengths for the four different branches of the tree, using three methods: beta with spikes (green), diffusion (blue) and Kim Tree [67] (purple). The true value of each of the branch lengths is plotted as a black horizontal line. The figure on the bottom shows the absolute value of the normalized difference between the inferred lengths and the true values, for all four branches and all 40 replicates.

is diffusion-based. Our implementation of the numerical approximation to the diffusion equation by Zhao et al. [221] failed at inferring the split times, contrary to our expectations. This issue remains to be investigated further.

This new approximation opens a variety of possibilities for inference problems, ranging from modeling the history of multiple populations to the evolution of just one allele where time serial data is available, including mutation and selection. The presented theoretical background can also be easily extended to include migration between populations and variable population size.

The former is a natural addition without further complications of the mathematics, due to the linearity of migration in the evolutionary pressure function $g(x)$. For example, assuming no mutation/selection, if individuals migrate away from the population under study with probability $m_1$, and migrate from another population with constant allele frequency $x_c$ with probability $m_2$, we have $g(x) = (1 - m_1)x + m_2 x_c$.

Including a variable population size in the beta with spikes is straightforward due to the recursive nature of the approach. The calculations required for every generation $1 \leq i \leq t$, as illustrated in Table 5.1, rely on the population size $N$ through the variance, loss and fixation probabilities. For each step, $N$ can be replaced with a population size that is generation-dependent, $N_t$, without affecting any of the presented theory.

5.5 Future work

This chapter presents work in progress, which I will resume upon the completion of my PhD. The focus will be on inference of mutation rates and selection coefficients on a tree. This will be approached using a two step process. We will first perform an estimate of split times and mutation rates by averaging over all loci and use it as a background model. We will then use a locus-by-locus scan on top of the background model, to identify sites upon which selection could have acted.

We also wish to investigate whether we can
5. The Beta approach to a Wright-Fisher model

detect the correct underlying phylogeny (which is currently assumed to be known) by using a likelihood-based measure (such as the AIC) to compare the maximum likelihood for different phylogenies. Additionally, we consider moving towards a Bayesian framework for parameter inference and DIC for testing different phylogenies.

As illustrated in the Results section, the issue of parameters scaling is not resolved in this framework. Further investigation is required on the behavior of the distribution and its dependency on the scaling.
diCal-IBD: demography-aware inference of identity-by-descent tracts in unrelated individuals

Paula Tataru$^1$  Jasmine A. Nirody$^2$  Yun S. Song$^{3,4,5}$

The paper presented in this chapter was published in 2014 in *Bioinformatics*.


Apart from minor typographical and formatting changes, the content of this chapter is identical to the journal paper.

Abstract

**Summary:** We present a tool, diCal-IBD, for detecting identity-by-descent (IBD) tracts between pairs of genomic sequences. Our method builds on a recent demographic inference method based on the coalescent with recombination, and is able to incorporate demographic information as a prior. Simulation study shows that diCal-IBD has significantly higher recall and precision than that of existing SNP-based IBD detection methods, while retaining reasonable accuracy for IBD tracts as small as 0.1 cM.

**Availability:** [http://sourceforge.net/projects/dical-ibd](http://sourceforge.net/projects/dical-ibd)

---

$^1$Bioinformatics Research Centre, Aarhus University, Aarhus, Denmark

$^2$Biophysics Graduate Group, University of California, Berkeley, USA

$^3$Computer Science Division, University of California, Berkeley, USA

$^4$Department of Statistics, University of California, Berkeley, USA

$^5$Department of Integrative Biology, University of California, Berkeley, USA
The notion of identity-by-descent (IBD) between distantly related individuals is playing an increasing role in a variety of genetic analyses, including association mapping [27], inferring past demographic history [157, 168], and detecting signals of natural selection [3]. Currently there exist several useful methods for detecting IBD tracts. These methods are based on characterizing similar haplotypes [e.g., GERMLINE [76]] or considering patterns of linkage disequilibrium [e.g., fastIBD, Refined IBD, and IBDseq [24–26]], but they do not explicitly model genealogical relationships between genomic sequences. Here, we present a new IBD detection tool, diCal-IBD, which is based on a well-used genealogical process in population genetics, namely the coalescent with recombination. Another feature that distinguishes our method is that we can incorporate demographic information as a prior.

There seems to be no universally accepted definition of IBD. The definition we adopt is the same as that in Palamara et al. [157] and Ralph and Coop [168]. Specifically, an IBD tract is defined as a maximally contiguous genomic region that is wholly descended from a common ancestor without any recombination occurring within the region. In contrast to other methods, we allow IBD tracts to contain point mutations, which are likely to occur in humans due to comparable mutation and recombination rates.

diCal-IBD is able to detect IBD tracts with high accuracy in unrelated individuals, between whom the vast majority of shared tracts are below 1 cM. SNP-based methods are successful in detecting tracts longer than 2 cM, but have low power for shorter tracts, whereas sequence-based methods, such as diCal-IBD and IBDseq, maintain reasonable accuracy for tracts as small as 0.1 cM.

diCal-IBD utilizes a recently developed demographic inference method called diCal [182]. diCal is formulated as a hidden Markov model, a decoding of which returns the time to the most recent common ancestor (TMRCA) for each site when analyzing only a pair of sequences. A change in TMRCA requires a recombination event and diCal-IBD uses the posterior decoding of TMRCA to call IBD tracts above a user-specified length, optionally trimming the ends of the tracts that have low posterior probabilities.

diCal requires discretizing time by partitioning it into non-overlapping intervals. The user has the option of specifying any discretization scheme. The default setting implemented in diCal-IBD distributes the pair-wise coalescence probability uniformly over the intervals, similarly as in PSMC [122], under a constant population size model.

An alternative scheme concentrates the intervals in the time period that is most likely to give rise to tracts that are long enough to be detected accurately. These schemes are detailed in Supplementary Information. Given a variable population size history, we approximate it with a piecewise constant population size.

As an application of IBD prediction, we provide a framework for detecting natural selection. Using the average IBD sharing and posterior probability along the sequence, diCal-IBD identifies regions which exhibit high sharing relative to the background average, indicating possible influence of positive selection.

We refer the reader to the online Supplementary Information for details on data processing, options used in calling diCal, post-processing of posterior decoding, and identification of selection.

Figure 6.1: diCal-IBD was run with bin size 100, constant population size, time discretized in 10 intervals according to unconditional coalescence probability, minimum tract length of 0.1 cM, and trimming of tracts with a threshold of 0.2. Error bars show the variance.
6.3 Implementation

diCal-IBD is written in Python 2.7, is platform independent, and has a command line interface which allows the user to completely specify its behaviour. The implementation allows for parallel runs of diCal on different sequence pairs. diCal-IBD provides a visualization of the predicted tracts, their posterior probabilities and the corresponding TMRCAs, and sequence-wide average IBD sharing and posterior probability. Accuracy information is also provided if the true IBD tracts are known.

6.4 Performance

We carried out a simulation study to compare diCal-IBD with the state-of-the-art IBD detection methods. We used ms [92] to simulate full ancestral recombination graphs (ARGs) for 50 sequences of 10 Mb each. We used a constant recombination rate of $10^{-8}$, and the African and European demographic histories inferred by Tennesen et al. [200]. We simulated perfectly phased sequence data on the ARGs with a constant mutation rate of $1.25 \times 10^{-8}$ per base per generation. From the simulated ARGs, we reconstructed the true pairwise IBD tracts by finding maximally consecutive sites that have the same TMRCA for the pair in question. We only considered tracts of length $> 0.1$ cM. To run SNP-based methods, we generated SNP data with approximately 1 marker per 0.2 kb. For further details on running existing tools, see Supplementary Information.

Figure 6.1 shows the recall (percentage of true tracts which were correctly recovered), precision (percentage of predicted tracts which were correctly predicted), and F-score (harmonic mean of recall and precision) for each method. See Supplementary Information for other measures of accuracy as a function of the true tract length, as well as the effects of errors in the data, demography, discretization, and trimming based on posterior probabilities. As the figure shows, diCal-IBD was able to recall significantly more tracts with greater precision than could SNP-based methods, leading to a much higher F-score. diCal-IBD was run assuming a constant population size, but its accuracy performance for the examples considered did not seem to be affected much by using this incorrect prior. This suggests that the posterior distribution inferred by diCal is robust to mis-specification of population sizes; whether this trend persists for more complex demographics deserves further investigation. The precision and recall performance of diCal-IBD was comparable to that of IBDseq, with neither one strictly dominating the other on all population size histories. See Fig. 6.1. A strength of diCal-IBD is its ability to explicitly incorporate demographic information. diCal, on which diCal-IBD is based, was originally developed for inferring variable effective population sizes, but it is being extended to handle more complex demographic models, incorporating multiple populations, population splits, migration, and admixture. diCal-IBD will be updated in parallel with diCal and hence will be able to use a complex demographic model as a prior.

Figure 6.2 illustrates the potential of applying diCal-IBD to identify regions under selection [3]. We refer the reader to Supplementary Information for further details.

Acknowledgments and funding

We thank Sara Sheehan, Jack Kamm, Matthias Steinrücken, and other members of the Song group for helpful discussions.

This research is supported in part by an NSF IGERT grant (J.A.N.) from CiBER at UC Berkeley, an NIH grant R01-GM094402 (Y.S.S.), and a Packard Fellowship for Science and Engineering (Y.S.S.)
6. diCal-IBD

6.5 Supplementary information

Running diCal

Time discretization

diCal [182] is formulated as a hidden Markov model based on the coalescent with recombination. For a sample of size 2, hidden states represent the time to the most recent common ancestor (TMRCA) scaled relative to a given haploid reference population size $2N_{ref}$. To apply the standard HMM algorithms, we discretize time into $d$ non-overlapping intervals $t_0 = 0 < t_1 < \cdots < t_d = \infty$, with $t_i = g_i/(2N_{ref})$, where $g_i$ is the number of generations back in time. In diCal-IBD, the user has the option of specifying any discretization scheme. Additionally, we offer two discretization procedures.

Consider the coalescent process in discrete time, with a known population size history, where $N(j)$ is the size at generation $j$. The probability of two sequences coalescing at generation $g$—that is, the probability that the TMRCA in generations $g$—is given by

$$P(g_{MRCA} = g) = \frac{1}{N(g)} \prod_{j=1}^{g-1} \left(1 - \frac{1}{N(j)}\right).$$

The probability $P_i$ of the TMRCA in generations being placed in a specific interval $[g_i, g_{i+1})$ is

$$P_i = \sum_{g=g_i}^{g_{i+1}-1} P(g_{MRCA} = g) = \prod_{j=1}^{g_{i+1}-1} \frac{1}{N(g)} \prod_{j=g_i}^{g-1} \left(1 - \frac{1}{N(j)}\right),$$

where

$$P_i = \prod_{j=1}^{g_i-1} \left(1 - \frac{1}{N(j)}\right),$$

denotes the probability of the TMRCA in generations being older than $g_i - 1$.

Unconditional discretization

This discretization procedure distributes the coalescence probability uniformly over the time intervals by setting $P_i = \frac{1}{d}$ for all $i = 0, \ldots, d-1$. The corresponding values $g_i$ can be determined numerically for increasing values of $i$.

When assuming a constant population size, this discretization is similar to the one used in PSMC [122]. The main difference arises from the last time interval. In PSMC, $t_d$ is chosen manually, while here $t_d = \infty$.

Conditional discretization

The goal of this discretization procedure is to have more time intervals in the period where the TMRCA in generations of tracts longer than $m$ cM are expected to be found. Such a discretization scheme should be useful when one is interested in tracts above a certain length while very short tracts (tracts with large TMRCA) are disregarded.

To implement this procedure, we need to calculate the coalescence probability at a random locus conditioned on being spanned by a tract longer than $m$ cM. Define the following quantities:

- $q(g, l)$, the joint density that a random locus has TMRCA in generations $g$ and is spanned by a tract of length $l$;
- $p(g, m)$, the joint probability that a random locus has TMRCA in generations $g$ and is spanned by a tract of length at least $m$;
- $p(m)$, the probability of a random locus being spanned by a tract of length at least $m$.

These quantities are related by

$$p(g, m) = \int_m^\infty q(g, l) \, dl,$$

$$p(m) = \sum_{g=1}^\infty p(g, m).$$

Palamara et al. [157] showed

$$q(g, l) = P(g_{MRCA} = g) \cdot \text{Erl}_2 \left( l; \frac{g}{50} \right),$$

denoting the probability of the TMRCA in generations being older than $g_i - 1$. From which we obtain

$$p(g, m) = P(g_{MRCA} = g) \cdot \int_m^\infty \text{Erl}_2 \left( l; \frac{g}{50} \right) \, dl = P(g_{MRCA} = g) \cdot e^{-\frac{g}{50}} \left(1 + \frac{g \cdot m}{50}\right).$$

The Erl2 (Erlang-2) distribution results as the sum of two i.i.d. exponentially distributed variables with equal rates.

We discretize the time such that
\[
\sum_{g=g_i}^{g_{i+1}-1} p(g \mid m) = \frac{P_i}{p(m)} \\
\cdot \left[ \sum_{g=g_i}^{g_{i+1}-1} e^{-\frac{g \cdot m}{50}} \left( 1 + \frac{g \cdot m}{50} \right) \right] \\
\cdot \frac{1}{N(g)} \prod_{j=g_i}^{g-1} \left( 1 - \frac{1}{N(j)} \right) \\
= \frac{1}{d}.
\]

As before, we numerically solve for \( g_i \) for increasing values of \( i \).

### Population size approximation

If the user specifies an arbitrary population size history, diCal-IBD approximates it by a piecewise constant function and uses the approximation in running diCal. The approximation preserves the coalescence probability within each time interval. Let \( N_i \) be the approximation of the population size for the time interval \([g_i, g_{i+1})\). Then, the coalescence probability in interval \( i \) is given by

\[
P_i = \frac{1}{N_i} \sum_{g=g_i}^{g_{i+1}-1} \left( 1 - \frac{1}{N_i} \right)^{g-g_i} \\
= \frac{1}{N_i} \sum_{g=0}^{g_{i+1}-g_i-1} \left( 1 - \frac{1}{N_i} \right) \\
= \frac{1}{N_i} \left[ 1 - \left( 1 - \frac{1}{N_i} \right)^{g_{i+1}-g_i} \right],
\]

which yields

\[
\frac{1}{N_i} = \left[ 1 - \left( 1 - P_i \right)^{\frac{1}{g_{i+1}-g_i}} \right],
\]

where \( P_i \) and \( \overline{P}_i \) are calculated using the given true population size history.

A constant population size is assumed if no demographic information is provided. Specifically, if the per-generation per-site mutation rate \( \mu \) for a

6.5. Supplementary information

locus is known and the locus consists of \( l \) sites, diCal-IBD uses \( N(j) = N_w \) for \( j > 0 \), where

\[
N_w = \frac{\text{number of segregating sites}}{4 \mu \sum_{i=1}^{n-1} \frac{1}{l}}.
\]

**Data binning**

As in PSMC [122], for speed and memory considerations, we group consecutive sites using a user-specified bin size prior to calling diCal. We allow for presence of missing nucleotides (marked as ‘N’) in the data. A bin is marked as ‘missing’ if more than 90% of the nucleotides in the bin are missing, as heterozygous if the bin is not ‘missing’ and at least one base within the bin is heterozygous, and homozygous otherwise. For the analysis described here, we used a bin size of 100.

**Other options**

When running diCal, we assume that the mutation and recombination rates are given. Before passing on the rates provided by the user, diCal-IBD scales them by multiplying with \( 4N_{\text{ref}} \). We note that due to binning of the input sequences, the mutation and recombination rates are also adjusted by multiplying by the bin size.

Missing heterozygotes (false negatives) can occur due to insufficient read depth. If provided a false negative rate, diCal-IBD lowers the mutation rate to account for missing heterozygotes, as in PSMC [122].

In addition to mutation and recombination rates, diCal also requires a mutation matrix. Our binning method results in only two types of bins in the data, which we mark as ‘A’ for homozygous bins and ‘C’ for heterozygous bins. Therefore in the mutation matrix, all rates are set to 0, except \( A \leftrightarrow C \), which are set to 1.

Together with the required input to diCal (the fasta file, using command option -F and the parameters file, using command option -I), we provide the previously described discretization and population size history. As we do not require estimation of the history, we set the number of EM iterations to 0 (using the command option -N 0), the pattern of parameters spanning the time intervals to 1 + 1 + \cdots + 1, where the number of 1’s corresponds to a given \( d \) (using the command option -p), and request the posterior decoding (using the command option -d 5).

For the presented analysis we ran diCal v1.2.
Processing diCal output

Identification of tracts

We call IBD tracts using the posterior decoding from diCal, as maximally consecutive sites that have the same TMRCA for the pair in question. We disregard the tracts for which the TMRCA is placed in the last time interval, as this could be an artifact resulting from the lack of finer intervals. Provided that there are enough time intervals, this removal should not change the performance of diCal-IBD drastically, as tracts with old TMRCA are expected to be very short.

As we run diCal on binned data, we recalculate the position of the detected tracts to correspond to the original sequence length.

For reporting and accuracy calculation purposes, we only consider tracts above a user-defined length. For the presented analysis, we used a minimum length of 0.1 cM.

Trimming of tracts

In addition to the inferred TMRCA for each predicted tract, we also obtain the corresponding posterior probabilities, which can be used as a measure of confidence for our prediction along the tract. We optionally use this extra information by discarding bins and tracts whose posterior probabilities are too low. To trim a tract, we discount bins at both ends if the corresponding posterior probability is below a given threshold. After trimming, we keep the tract only if the average posterior probability of the remaining bins is above the threshold. Reporting of tracts is contingent only on the original length: that is, we still report trimmed tracts even if they are shorter than the specified minimum length. However, for our accuracy calculations we do not count the contribution of tracts which do not meet the minimum length requirement after trimming.

Simulations

Simulation of ARGs

We used the program ms [92] to simulate full ancestral recombination graphs (ARGs) with a per-base recombination rate of $10^{-8}$ for 50 sequences of 10 Mb each. We used two different population histories, corresponding to the African (AF) and European (EA) demographic histories inferred by [200]. As shown in Fig. 6.3a, both histories contain an ancient bottleneck and periods of rapid population expansion in the recent past. The European population contains the out-of-Africa bottleneck and a more recent bottleneck and, as well as two different epochs of rapid expansion.

To calculate the scaled recombination rate and the corresponding population sizes, we used an $N_{ref}$ of 1861. The ms commands used for the AF and EA demographies are

```
ms 50 1 -T -t 9.305e2
  -r 744.39992556 10000000
  -eN 0 227.8345
  -eG 0 122.63987
  -eN 2.7539e-2 7.77754
  -eN 7.9527e-1 3.927996
ms 50 1 -T -t 9.305e2
  -r 744.39992556 10000000
  -eN 0 275.1209
  -eG 0 145.55032
  -eG 2.7539e-2 22.889
  -eN 1.2359e-1 1
  -eN 2.7405e-1 7.77754
  -eN 7.9527e-1 3.927996
```

We note that the -t option is only used for later recovery of the mutation rate.

Figure 6.3b shows scatter plots of the simulated tracts. Due to the low number of simulated sequences, very few tracts are above 1 cM. The higher number of tracts above 0.1 cM in the EA dataset is a result of the reduced population size after the out-of-Africa bottleneck. This event left a strong signal in the data, creating a clear separation between the tracts that are older and the ones that are younger than the bottleneck.

Simulation of sequences

We generated sequence data on the ARGs using a per-base mutation rate of $1.25 \times 10^{-8}$ [109, 110]. In real data, errors in base calling can lead to nonexistent variants being called (false positives) or to true variants being overlooked (false negatives). To account for this, we superimposed sequencing errors, adding variants with rate $6 \times 10^{-6}$ and removing variants with rate 0.02, in accordance with reported false positive and false negative rates [94] and previous studies [26]. To remove variants, we paired the sequences (to mimic diploids) and, for each heterozygous position, we changed one of the two alleles so that the site became homozygous. Due to our pairing of the sequence, we added errors with probability twice the false negative rate. To add false positives, we chose the number of false SNPs from a Poisson distribution with mean equal
6.5. Supplementary information

Figure 6.3: Population histories used for simulating data. African (AF) and European (EA) as inferred by [200]. (a) Population size histories used. AF and EA share a common history until almost 2000 generations ago. (b) Simulated pairwise tracts as a function of their length and TMRCA in generations. The black dots correspond to the tracts longer than 0.1 cM, which are used in the presented analysis, out of which only 4 (AF) and 5 (EA) are above 1 cM.

Table 6.1: Summary of the simulated sequence data. The removed variants were not necessarily from different SNPs, so the total number of SNPs affected by the false negative errors is lower than the number of removed variants. Some of the SNPs could be present in very low frequency, so a removed variant could lead to the complete loss of the SNP. The EA dataset had an overall lower number of SNPs, most likely a consequence of the out-of-Africa bottleneck.

<table>
<thead>
<tr>
<th></th>
<th>Simulated SNPs</th>
<th>Removed variants</th>
<th>Affected SNPs</th>
<th>Removed SNPs</th>
<th>Added SNPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF</td>
<td>24370</td>
<td>4368</td>
<td>3809</td>
<td>164</td>
<td>3059</td>
</tr>
<tr>
<td>EA</td>
<td>12678</td>
<td>3088</td>
<td>2624</td>
<td>80</td>
<td>3077</td>
</tr>
</tbody>
</table>

To run other programs, we generated SNP data from the simulated sequence according to the SNP density (15 million SNPs genome wide) reported by the 1000 Genomes Project Consortium [1]. After inclusion of all segregating sites, we randomly selected additional sites such that the resulting marker density is approximately 1 marker per 0.2kb.

Table 6.1 shows the summary of the simulated sequence data. The removed variants were not necessarily from different SNPs, so the total number of SNPs affected by the false negative errors is lower than the number of removed variants. Some of the SNPs could be present in very low frequency, so a removed variant could lead to the complete loss of the SNP. The EA dataset had an overall lower number of SNPs, most likely a consequence of the out-of-Africa bottleneck.

Accuracy measures

From the simulated ARGs, we reconstructed the true pairwise IBD tracts by finding maximally consecutive sites that have the same TMRCA for the pair in question. We note that this excludes recombinations which do not affect the coalescence time, an event which occurs only rarely. We only considered the tracts of length > 0.1 cM.

To calculate the accuracy, we use the percentage overlap between true and predicted tracts, rather than their lengths. Doing so, all tracts, re-
follows. We note that true
define mathematically the accuracy measures as
rectly predicted), and F-score (the harmonic mean
tracts which were correctly recovered), precision
false positive, which is a function of the length
false negative: any part of true, that does
power, all overlaps between true and predicted
under-prediction: the part of true, that does
over-prediction: the part of pred, that does
All measures listed above are calculated as a
function of the length of the true tract, except the
false positive, which is a function of the length of
the predicted tract. In addition to the above
measures, we report recall (the percentage of true
tracts which were correctly recovered), precision
(the percentage of predicted tracts which were
correctly predicted), and F-score (the harmonic mean
of recall and precision). Together, these provide
an overall measure of accuracy.
Let \( I \) be the indicator function. Then we can
define mathematically the accuracy measures as
follows. We note that true, represents both the
tract and its length.

\[
\text{true positive}_i = \frac{\text{true}_i \cap \text{pred}_i^*}{\text{true}_i} \\
\text{false negative}_i = \frac{\text{true}_i - \sum_j \text{true}_i \cap \text{pred}_j}{\text{true}_i} \\
\text{false positive}_j = \frac{\text{pred}_j - \sum_i \text{pred}_j \cap \text{true}_i}{\text{pred}_j} \\
\text{power}_i = \frac{\sum_j \text{true}_i \cap \text{pred}_j}{\text{true}_i} \\
\text{under-prediction}_i = \frac{\text{true}_i - \text{true}_i \cap \text{pred}_i^*}{\text{true}_i} \\
\text{over-prediction}_i = \frac{\text{pred}_i^* - \text{true}_i \cap \text{pred}_i^*}{\text{true}_i}.
\]

true positive
\[
= \frac{1}{6} \left( \frac{\text{true}_1 \cap \text{pred}_1}{\text{true}_1} + \frac{\text{true}_3 \cap \text{pred}_2}{\text{true}_2} + \frac{\text{true}_6 \cap \text{pred}_6}{\text{true}_6} \right),
\]
false negative
\[
= \frac{1}{6} \left( \frac{\text{true}_1 - \text{true}_1 \cap \text{pred}_1}{\text{true}_1} + \frac{\text{true}_2}{\text{true}_2} + \frac{\text{true}_3 - \text{true}_3 \cap \text{pred}_2}{\text{true}_3} + \frac{\text{true}_5 - \text{true}_5 \cap \text{pred}_3}{\text{true}_5} + \frac{\text{true}_6 - \text{true}_6 \cap \text{pred}_4 - \text{true}_6 \cap \text{pred}_6}{\text{true}_6} \right),
\]
false positive
\[
= \frac{1}{6} \left( \frac{\text{pred}_1 - \text{pred}_1 \cap \text{true}_1}{\text{pred}_1} + \frac{\text{pred}_2 - \text{pred}_2 \cap \text{true}_3 - \text{pred}_2 \cap \text{true}_4}{\text{pred}_2} + \frac{\text{pred}_4 + \text{pred}_6 - \text{pred}_6 \cap \text{true}_6}{\text{pred}_6} \right),
\]

Figure 6.4 gives an overview of the different
types of overlaps. Both true and predicted tracts
are numbered, while for the predicted tract, the
corresponding \( i^* \) index is given after the equal
sign. A predicted tract can be the best predic-
tion for several true tracts. The overall average
accuracy for the example in the figure is given by

\[
\text{true positive}_i = \frac{\text{true}_i \cap \text{pred}_i^*}{\text{true}_i} \\
\text{false negative}_i = \frac{\text{true}_i - \sum_j \text{true}_i \cap \text{pred}_j}{\text{true}_i} \\
\text{false positive}_j = \frac{\text{pred}_j - \sum_i \text{pred}_j \cap \text{true}_i}{\text{pred}_j} \\
\text{power}_i = \frac{\sum_j \text{true}_i \cap \text{pred}_j}{\text{true}_i} \\
\text{under-prediction}_i = \frac{\text{true}_i - \text{true}_i \cap \text{pred}_i^*}{\text{true}_i} \\
\text{over-prediction}_i = \frac{\text{pred}_i^* - \text{true}_i \cap \text{pred}_i^*}{\text{true}_i}.
\]
6.5. Supplementary information

Figure 6.4: Types of overlaps. Regions in both true and predicted tracts are colored such that: green indicates overlap between true and predicted tracts and blue (red) indicate regions in true (predicted) tracts that do not overlap with other predicted (true) tracts. Horizontal lines (both gray and black) mark, for each true tract, the best overlap with a predicted tract. Black horizontal lines indicate that the overlap in percentage is above the required threshold $a$. For details, see main text.

under-prediction

$$\frac{1}{6} \left( \frac{\text{true}_1 - \text{true}_1 \cap \text{pred}_1}{\text{true}_1} + \frac{\text{true}_2}{\text{true}_2} + \frac{\text{true}_3 - \text{true}_3 \cap \text{pred}_2}{\text{true}_3} + \frac{\text{true}_5 - \text{true}_5 \cap \text{pred}_3}{\text{true}_5} + \frac{\text{true}_6 - \text{true}_6 \cap \text{pred}_6}{\text{true}_6} \right),$$

over-prediction

$$\frac{1}{6} \left( \frac{\text{pred}_1 - \text{pred}_1 \cap \text{true}_1}{\text{true}_1} + \frac{\text{pred}_2 - \text{pred}_2 \cap \text{true}_3}{\text{true}_3} + \frac{\text{pred}_6 - \text{pred}_6 \cap \text{true}_6}{\text{true}_6} \right).$$

**Detecting selection**

Positive selection has been shown to result in an increase in IBD sharing [3, 78]. We provide a framework within diCal-IBD to identify genomic regions with high sharing relative to the background average. We calculate the sharing for non-overlapping windows as the total length of the tracts spanning each window, divided by the number of pairwise comparisons. We consider a window to have increased sharing if it is greater than three standard deviations from the average sharing across all sites. Similarly, we calculate the average posterior probability by considering the average posterior of the tracts spanning the windows, divided by the number of tracts per window. In the presented analysis we used a window size of 0.1 cM.

**diCal-IBD implementation**


**Running other programs**

We compare the performance of diCal-IBD on the simulated data with four IBD detection softwares

- GERMLINE version 1.5.1 [76];
- fastIBD, packaged with BEAGLE version 3.3.1 [24];
- Refined IBD, packaged with BEAGLE version 4 [26];
- IBDseq version 1206 [25].

In the following we describe the options we used while running these programs. We note that this improved the performance compared to the default settings. Only GERMLINE and IBDseq allow for correcting errors in the data. To create diploid input for these programs, we replicated and paired together each sequence to generate “diploid” data which was homozygous at all sites.

**GERMLINE**

As we only considered IBD segments larger than 0.1 cM, we set $\text{min}_m = 0.1$. We also made use of GERMLINE’s $\text{hextend}$ option, which improves performance when data is well-phased. The $\text{bits}$ parameter determines the number of markers to be considered in an exact matching seed. Choosing a value which is too large may result in missing shorter IBD segments, while one that is too small affects computation time. We set this parameter so that a seed length corresponded to 0.02 cM, which was significantly shorter than our threshold.
for candidate IBD segments (0.1 cM). For example, for a density of 1 marker per 0.2kb, \( \text{bits} = \frac{20,000}{200} = 100 \). Note that the recombination rate we used in the simulated data, \( r = 10^{-8} \), implies that 0.02 cM corresponds to 20kb. When running on perfect data, we set the allowed number of homozygous mismatches per seed \( \text{err\_hom} \) to 0. Otherwise, we calculate the number of expected errors in a bin (here, 20kb). Using this method, we set \( \text{err\_hom} = 1 \). Because our data consisted of haploid sequences, we set \( \text{err\_het} = 0 \).

### fastIBD

We used the default \( \text{ibdscale} = 2 \). Because of no phase uncertainty in our data, we ran only one iteration (\( \text{ninterations} = 1 \)) and allowed for a very high score threshold of \( \text{fastibdthreshold} = 10^{-2} \).

### Refined IBD

Refined IBD uses the GERMLINE algorithm to find candidate IBD segments, and the \( \text{ibdwindow} \) parameter is equivalent to the GERMLINE \( \text{bits} \) parameter, and was set to the same value (\( \text{ibdwindow} = 100 \)). Likewise, we set \( \text{ibdcm} = 0.1 \), as this parameter is equivalent to GERMLINE’s \( \text{min\_m} \). We set \( \text{ibdtrim} = 75 \), which is significantly lower than the recommended value (the typical number of markers in a 0.15 cM segment \( \sim \frac{1.5 \times 10^5}{200} = 750 \)), as higher values reduce the power to detect shorter segments. Due to no phase uncertainty in our data, we chose \( \text{usephase} = \text{true} \) and \( \text{phase-its} = 1 \). The minimum LOD score for reported IBD was set to \( \text{ibdlod} = 1.0 \).

### IBDseq

Because IBDseq is sequence-based, variants that are in LD are filtered by using a user-specified threshold for the allowed values of \( r^2 \) given as \( \text{r2max} \). In our experiments, variant filtering reduced drastically the performance of IBDseq and we have therefore set \( \text{r2max} = 1 \), which means that all variants are used in the analysis. IBDseq closely resembles Refined IBD in that, for each variant, it calculates a LOD score that is a ratio between the likelihood of being IBD versus not being IBD. However, unlike Refined IBD, the value of the \( \text{ibdlod} \) didn’t affect the results, and therefore we used the default value of 3.0. IBDseq also allows for presence of errors in the data, behavior controlled through the \( \text{errormax} \) parameter, which we set to the default value of 0.001 for both perfect data and data with genotyping error.

### Results and discussion

#### IBD detection

Figures 6.5 to 6.7 show the performance of GERMLINE, fastIBD, Refined IBD, IBDseq and diCal-IBD for the two simulated datasets. From the figures, it is clear that the two sequence-based methods (IBDseq and diCal-IBD) have an overall better performance, with an increased recall, while the precision is at least as high as for the other programs. Overall, all programs have better accuracy for the AF dataset. We believe this is mainly due to the higher number of SNPs present in the AF dataset (Table 6.1), which increases the amount of available information.

In Figs. 6.5 to 6.7, we also plotted the accuracy as a function of tract length using windows of 0.1 cM (i.e., the first window contains tracts between 0.1 cM and 0.2 cM in length, the next 0.2 cM - 0.3 cM, and so on). We note that the sudden jumps in the figures for the longer tracts are due to the low number of tracts in those ranges (Fig. 6.3 b). IBDseq and diCal-IBD have very high true positive and power for tracts as low as 0.1 cM, while the other programs recover only a modest portion of the tracts. The small difference between the true positive and power for diCal-IBD indicates that most true tracts overlap at least 50% with a predicted tract. This, together with the similarity between false negative and under-prediction, suggests that most often there is a unique correspondence between true and predicted tracts, in that the situations where one true tract overlaps with several predicted tracts, or vice versa, are rare.

The main fault in diCal-IBD is its relatively high false positive rate. For the AF dataset, the discrepancy between the false positive and over-prediction rates implies that a significant portion of the false positives correspond to predicted tracts that do not overlap with any true tract. These tracts may correspond to true tracts too short to pass the length filter; this is supported by the fact that the false positive rate is considerably higher in the first window considered. In contrast, for the EA dataset, the false positive and over-prediction rates are more comparable. This indicates that false positives in this prediction are the result of extension at the boundaries of true tracts.
6.5. Supplementary information

(a) Average recall, precision and F-score.

(b) The six accuracy measures calculated as a function of tract length, for the perfect datasets.

Figure 6.5: Accuracy results for GERMLINE, fastIBD, Refined IBD and IBDseq for both AF and EA datasets.
6. diCal-IBD

(a) Average recall, precision and F-score.

(b) The six accuracy measures calculated as a function of tract length, for the perfect dataset.

Figure 6.6: Accuracy results for diCal-IBD, using different population sizes, discretizations and posterior probability thresholds for trimming of tracts, for the AF dataset.
6.5. Supplementary information

(a) Average recall, precision and F-score.

(b) The six accuracy measures calculated as a function of tract length, for the perfect dataset.

Figure 6.7: Accuracy results for diCal-IBD, using different population sizes, discretizations and posterior probability thresholds for trimming of tracts, for the EA dataset.
Comparison to IBDseq

When comparing IBDseq and diCal-IBD, it is difficult to say whether one method or the other is better.

We note that the performance of IBDseq reported here is much better than the one in [25]. This is due to the parameters we chose for IBDseq. IBDseq requires that variants in LD are filtered before calculating the likelihood of being IBD. However, we have found that not filtering the variants drastically improved its performance. Additionally, IBDseq controls for errors in the data through a user-defined parameter. The performance of IBDseq is highly dependent on this parameter as well. Even more so, for the perfect simulated data, we have used the default value of 0.001 rather than 0, as this affected the final results. This is counter-intuitive, as this parameter should in fact be set to 0 for error free data.

For the AF dataset, IBDseq has an F-score that is slightly higher than diCal-IBD, but when considering the variances, this difference does not seem significant. For the EA dataset, diCal-IBD shows an increased F-score, but in this case, the variances indicate that the difference is significant.

Both methods have a lower F-score for the EA dataset than for the AF dataset, but the discrepancy for IBDseq is larger. This could indicate that IBDseq is less robust to demography than diCal-IBD. To this extent, all existing simulation studies have been performed on one panmictic population and little is known about the effects of population structure and admixture on IBD detection. Unlike all other considered methods, including IBDseq, diCal-IBD has the potential of including such complex demographics and correct for it while detecting IBD.

Effect of errors in the data

Sequencing errors in the data reduce diCal-IBD’s accuracy slightly, with a more pronounced effect for the EA dataset. Correcting for the false negative errors doesn’t seem to affect the results.

In contrast, the other programs seem to be more drastically affected by errors. GERMLINE shows a better performance for the datasets with errors. We believe this is a consequence of GERMLINE’s option to accommodate mismatches in tracts (the err_hom parameter, which was set to 1), which effectively allows for presence of point mutations inside tracts. fastIBD and Refined IBD do not allow for mismatches.

While this option in GERMLINE could be used to account for point mutations even in perfect datasets, the number of such changes is highly dependent on the population history.

Bin size

In the presented analysis we binned the sequences using a bin size of 100. To investigate whether this influences the performance, we reran diCal using a bin size of 10. The performance showed no difference between the two runs (results not shown).

Population size and discretization

We ran diCal-IBD assuming a constant population size (‘const size’) or approximating the true population size (‘var size’). For the former, we used the two discretizations described previously (‘const disc’ and ‘const cond disc’, respectively), considering tracts longer than $m = 0.01$ cM for the conditional discretization. For the latter, we recalculated the two presented discretizations under the variable population size (‘var disc’ and ‘var cond disc’), but also used the two discretizations under a constant size. This enables us to investigate the effect of using different population sizes when the discretization is fixed.

diCal-IBD generally shows similar F-scores under the different assumed population sizes and discretizations, indicating the robustness of the posterior distribution. The discretization seems to have a higher impact on the performance, as the recall and precision vary with the discretization used, but not with the population size assumed.

Trimming of tracts

We investigated the effect of trimming of tracts based on the posterior probability, using thresholds 0.2 and 0.4. It is clear that in most cases the precision increases, while the false positive and over-prediction are reduced. Trimming can, however, have the opposite desired effect, as it potentially removes ends of the predicted tracts that overlap correctly with a true tract.

Most of the effects of trimming are rather minor, indicating that tracts have overall posterior probabilities above 0.4. In some cases, trimming using 0.4 as threshold decreases the recall more drastically, without a large enough increase in the precision, to balance the final F-score.
6.5. Supplementary information

<table>
<thead>
<tr>
<th>Coriell ID</th>
<th>Population</th>
<th>Coriell ID</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA06985</td>
<td>CEU</td>
<td>NA18501</td>
<td>YRI</td>
</tr>
<tr>
<td>NA06994</td>
<td>CEU</td>
<td>NA18502</td>
<td>YRI</td>
</tr>
<tr>
<td>NA07357</td>
<td>CEU</td>
<td>NA18504</td>
<td>YRI</td>
</tr>
<tr>
<td>NA10851</td>
<td>CEU</td>
<td>NA18505</td>
<td>YRI</td>
</tr>
<tr>
<td>NA12004</td>
<td>CEU</td>
<td>NA18508</td>
<td>YRI</td>
</tr>
</tbody>
</table>

Table 6.2: CEU and YRI samples from Complete Genomics data.

Figure 6.8: Detection of high sharing using diCal-IBD. Dotted lines indicate the thresholds for considering that a region exhibits high sharing.

Density of time intervals

From our experiments, we observed that using time intervals that are too dense decreases the performance of diCal-IBD. This is somewhat counterintuitive, as one would expect that using more time intervals would allow diCal to identify the TMRCA of each tract more accurately. However, we believe the decline in performance to be the result of the large variation in TMRCA for tracts of the same length (Fig. 6.3 b). On the other hand, discretization schemes that are too sparse can blur the boundaries between adjacent IBD tracts. An increased density of time intervals can arise from using a conditional discretization with a high tract length threshold tracts (for example, 0.1 cM), or from the discretizations based on the variable population size. Such an example is the discretization for the EA dataset based on the variable population size, where the resulting dense intervals are probably due to the out-of-Africa bottleneck. The low confidence in the TMRCA is reflected in the effect of trimming of tracts, as the performance is drastically affected when using a threshold of 0.4.

Detecting selection

For illustrating the visualization method for identifying regions under positive selection, we used a 4 Mb segment of chromosome 15 (46 Mb - 50 Mb) from the 69 Genomes public dataset by Complete Genomics [45]. The SLC24A5 gene is located in this region, at 48.41–48.43 Mb. This gene is responsible for light skin pigment and lack of dependence on sunlight for vitamin D production, and has been found to be under positive selection in northern Europeans [209].

We ran diCal-IBD on samples from both European (CEU) and African (YRI) populations (Table 6.2). We assumed the mutation and recombination rates to be $1.25 \times 10^{-8}$ and $10^{-8}$, respectively. We used constant population size, the unconditional discretization and untrimmed tracts that were at least 0.1 cM long. Figure 6.8 shows the resulting average sharing and posterior probability, as described in Section 6.5. As expected, only the CEU data contains one peak with high sharing at 48.5–48.6 Mb, which neighbors the SLC24A5 gene.
Comparison of methods for calculating conditional expectations of sufficient statistics for continuous time Markov chains

Paula Tataru¹  Asger Hobolth¹

The paper presented in this chapter was published in 2011 in *BMC Bioinformatics*.


Apart from minor typographical and formatting changes, the content of this chapter is identical to the journal paper.

Abstract

**Background:** Continuous time Markov chains (CTMCs) is a widely used model for describing the evolution of DNA sequences on the nucleotide, amino acid or codon level. The sufficient statistics for CTMCs are the time spent in a state and the number of changes between any two states. In applications past evolutionary events (exact times and types of changes) are unaccessible and the past must be inferred from DNA sequence data observed in the present.

**Results:** We describe and implement three algorithms for computing linear combinations of expected values of the sufficient statistics, conditioned on the end-points of the chain, and compare their performance with respect to accuracy and running time. The first algorithm is based on an eigenvalue decomposition of the rate matrix (EVD), the second on uniformization (UNI), and the third on integrals of matrix exponentials (EXPM). The implementation in R of the algorithms is available at www.birc.au.dk/∼paula/.

**Conclusions:** We use two different models to analyze the accuracy and eight experiments to investigate the speed of the three algorithms. We find that they have similar accuracy and that EXPM is the slowest method. Furthermore we find that UNI is usually faster than EVD.

¹Bioinformatics Research Center, Aarhus University, Aarhus, Denmark
7. Conditional expectations for CTMCs

7.1 Background

In this paper we consider the problem of calculating the expected time spent in a state and the expected number of jumps between any two states in discretely observed continuous time Markov chains (CTMCs). The case where the CTMC is only recorded at discretely observed time points arises in molecular evolution where DNA sequence data is extracted at present day and past evolutionary events are missing. In this situation, efficient methods for calculating these types of expectations are needed. In particular, two classes of applications can be identified.

The first class of applications is concerned with rate matrix estimation. Holmes and Rubin [89] describe how the expectation-maximization (EM) algorithm can be applied to estimate the rate matrix from DNA sequence data observed in the leaves of an evolutionary tree. The EM algorithm is implemented in the software XRate [105] and has been applied in [111] for estimating empirical codon rate matrices. Holmes and Rubin [89] use the eigenvalue decomposition of the rate matrix to calculate the expected time spent in a state and the expected number of jumps between states.

The second class of applications is concerned with understanding and testing various aspects of evolutionary trajectories. In [145] it is emphasized that analytical results for jump numbers are superior to simulation approaches and various applications of jump number statistics are provided, including a test for the hypothesis that a trait changed its state no more than once in its evolutionary history and a diagnostic tool to measure discrepancies between the data and the model. Minin and Suchard [145] assume that the rate matrix is diagonalizable and that the eigenvalues are real, and applies a spectral representation of the transition probability matrix to obtain the expected number of state changes.

Dutheil et al. [49] and Dutheil and Galtier [48] describe a method, termed substitution mapping, for detecting co-evolution of evolutionary traits, and a similar method is described in [146]. The substitution mapping is achieved using the eigenvalue decomposition method suggested by Holmes and Rubin [89]. In [49], [48] and [47] the substitution mapping is achieved using a more direct formula for calculating the number of state changes. In this direct approach an infinite sum must be truncated and it is difficult to control the error associated with the truncation. An alternative is described in [184] where uniformization is applied to obtain the expected number of jumps. Siepel et al. [184] use the expected number of jumps on a branch to detect lineages in a phylogenetic tree that are under selection.

A third algorithm for obtaining the number of changes or time spent in a state is outlined in [85]. The algorithm is based on [204] where a method for calculating integrals of matrix exponentials is described.

A natural question arises: which of the three methods (eigenvalue decomposition, uniformization or matrix exponentiation) for calculating conditional expectations of summary statistics for a discretely observed CTMC should be preferred? The aim of this paper is to provide an answer to this question. We describe and compare the three methods. Our implementations in R [166] are available at www.birc.au.dk/~paula/. Furthermore the eigenvalue decomposition and uniformization methods are also available as a C++ class in the bio++ library at http://biopp.univ-montp2.fr/.) The performance and discussion of the algorithms are centered around two applications. The first application is concerned with rate matrix estimation; we estimate the Goldman-Yang codon model [68] using the expectation-maximization algorithm. The second application is based on the labeled distance estimation presented in [156].

Consider a stochastic process \( \{X(s) : 0 \leq s \leq t \} \) which can be described by a CTMC with \( n \) states and an \( n \times n \) rate matrix \( Q = (q_{cd}) \). The off-diagonal entries in \( Q \) are non-negative and rows sum to zero, i.e. \( q_{cc} = -\sum_{d \neq c} q_{cd} = -q_c \). Maximum likelihood estimation of the rate matrix from a complete observation of the process is straight...
forward. The likelihood of the process, conditional on the beginning state $X(0)$, is given by (e.g. [82])

$$L(Q; \{X(s) : 0 \leq s \leq t\}) = \exp \left( -\sum_c q_c T_c \left( \prod_{c=1}^{n} \prod_{d \neq c} q_{cd}^{n_{cd}} \right) \right), \quad (7.1)$$

where $T_c$ is the total time spent in state $c$ and $N_{cd}$ is the number of jumps from $c$ to $d$. The necessary sufficient statistics for a CTMC are thus the time spent in each state and the number of jumps between any two states. In applications, however, access is limited to DNA data from extant species. The CTMC is discretely observed and we must estimate the mean values of $T_c$ and $N_{cd}$ conditional on the end-points $X(0) = a$ and $X(t) = b$. From [82] we have that

$$E[T_c \mid X(0) = a, X(t) = b] = E[T_c \mid t, a, b] = \frac{I_{ab}(t)}{p_{ab}(t)} \quad \text{and} \quad E[N_{cd} \mid X(0) = a, X(t) = b] = E[N_{cd} \mid t, a, b] = \frac{q_{cd} I_{cd}^{ab}(t)}{p_{cd}(t)}, \quad (7.2)$$

where $P(t) = (p_{ij}(t)) = e^{Qt}$ is the transition probability matrix and

$$I_{cd}^{ab}(t) = \int_0^t p_{ac}(u) p_{db}(t-u) \, du. \quad (7.3)$$

Many applications require a linear combination of certain substitutions or times. Examples include the number of transitions, transversions, synonymous and non-synonymous substitutions. In the two applications described below the statistics of interest is a linear combination of certain substitutions and times. Let therefore $C$ be an $n \times n$ matrix and denote by $\Sigma(C; t)$ the matrix with entries

$$\Sigma(C; a, b, t) = \sum_{c,d} C_{cd} I_{cd}^{ab}(t). \quad (7.4)$$

We describe, compare and discuss three methods for calculating $\Sigma(C; t)$. The evaluation of Eq. (7.3) takes $O(n^3)$ time and therefore a naive calculation, assuming that $C$ contains just one entry different from zero has a $O(n^3)$ running time. Even worse, if $C$ contains $O(n^2)$ entries different from zero, then the naive implementation has a $O(n^7)$ running time. For all three methods our implementations of $\Sigma(C; t)$ run in $O(n^3)$ time.

### 7.2 Results

#### Applications

**Application 1: Rate matrix estimation**

Our first application is the problem of estimating the parameters in a CTMC for evolution of coding DNA sequences which we describe using the $61 \times 61$ rate matrix (excluding stop codons) given by Goldman and Yang [68]

$$q_{ij} = \begin{cases} 
\alpha \kappa \pi_j & \text{if } i \xrightarrow{s,ts} j , \\
\alpha \pi_j & \text{if } i \xrightarrow{s,tv} j , \\
\alpha \omega \kappa \pi_j & \text{if } i \xrightarrow{ns,ts} j , \\
\alpha \omega \pi_j & \text{if } i \xrightarrow{ns,tv} j , \\
0 & \text{otherwise,}
\end{cases} \quad (7.5)$$

where $\pi$ is the stationary distribution, $\kappa$ is the transition/transversion rate ratio, $\omega$ is the non-synonymous/synonymous ratio, $\alpha$ is a scaling factor and $s$, $ns$, $ts$ and $tv$ stand for synonymous, non-synonymous, transition and transversion. The stationary distribution $\pi$ is determined directly from the data using the codon frequencies. We estimate the remaining parameters $\theta = (\alpha, \kappa, \omega)$ using the expectation-maximization (EM) algorithm [43] as described below.

Suppose the complete data $x$ is available, consisting of times and types of substitutions in all sites and in all branches of the tree. The complete data log likelihood is, using Eqs. (7.1) and (7.5),

$$\ell(\alpha, \kappa, \omega; x) = -\alpha \cdot (L_{s,tv} + \omega L_{ns,tv} + \kappa L_{s,ts} + \kappa \omega L_{ns,ts}) + N \log \alpha + N_{ts} \log \kappa + N_{ns} \log \omega,$$

where we use the notation

$$L_{s,ts} = \sum_i T_i \sum_j \pi_j I((i, j) \in \mathcal{L}_{s,ts}), \quad (7.6)$$

$$N_{ts} = \sum_{i,j} N_{ij} I((i, j) \in \mathcal{L}_{ts}),$$

where e.g.

$$\mathcal{L}_{s,ts} = \{(i, j) : \begin{align*}
i \text{ and } j & \text{ differ at one position} \\
\text{and the substitution of } i \text{ with } j & \text{ is a synonymous transition}\end{align*}\}.$$

A similar notation applies for $L_{s,tv}$, $L_{ns,ts}$, $L_{ns,tv}$, $N_{ns}$ and $N$, where the last statistic is the sum of substitutions between all states $(i, j)$ that differ at
one position and s, ns, ts and tv subscripts stand for synonymous, non-synonymous, transition and transversion.

The complete data log likelihood can be maximized easily by making the re-parametrization \( \beta = \alpha \kappa \). We find that

\[
\hat{\alpha} = \frac{N_{tv}}{L_{sv, tv} + \tilde{\omega}L_{ns, tv}}, \quad \hat{\beta} = \frac{N_{ts}}{\tilde{\omega}L_{ns, ts} + \alpha L_{ns, ts}},
\]

\[
\tilde{\omega} = -\frac{b + \sqrt{b^2 - 4ac}}{2a},
\]

where

\[
a = -L_{ns, tv}L_{ns, ts}N_s, \quad c = L_{ns, tv}L_{ts, tv}N_{ns}
\]

\[
b = L_{ns, tv}L_{ts, tv}(N_{ns} - N_{tv})
\]

\[
+ L_{ns, ts}L_{tv, tv}(N_{ns} - N_{ts})
\]

In reality the data \( y \) is only available in the leaves and the times and types of substitutions in all sites and all branches of the tree are inaccessible. The EM algorithm is an efficient tool for maximum likelihood estimation in problems where the complete data log likelihood is analytically tractable but full information about the data is missing.

The EM algorithm is an iterative procedure consisting of two steps. In the E-step the expected complete log likelihood

\[
G(\theta; \hat{\theta}_0, y) = E_{\hat{\theta}_0}[\ell(\theta; x) \mid y],
\]

conditional on the data \( y \) and the current estimate of the parameters \( \hat{\theta}_0 \) is calculated. In the M-step the parameters are updated by maximizing \( G(\theta; \hat{\theta}_0, y) \). The parameters converge to a local maximum of the likelihood for the observed data.

The expected log likelihood conditional on the data \( y \) and under the three parameters \( \alpha, \kappa \) and \( \omega \) is

\[
E[\ell(\alpha, \kappa, \omega; x) \mid y]
= -\alpha \cdot (E[L_{s, t} \mid y] + \omega E[L_{ns, t} \mid y])
+ \kappa E[L_{ts, t} \mid y] + \omega \kappa E[L_{ns, ts} \mid y]
+ E[N_{ns} \mid y] \log \kappa + E[N_{ts} \mid y] \log \omega
+ E[N \mid y] \log \alpha.
\]

Therefore the E-step requires expectations of linear combinations of waiting times in a set of states and number of jumps between certain states. Because of the Markov property this calculation can be divided in two parts. First we use the peeling algorithm \([61, 216]\) to obtain the probability \( P(\gamma_k = a, \beta_k = b \mid y, t_k) \) that a branch \( k \) of length \( t_k \) with nodes \( \gamma_k \) and \( \beta_k \) above and below the branch, respectively, has end-points \( a \) and \( b \). Second, we calculate the desired summary statistic by summing over all branches. For example we have

\[
E[L_{s, t} \mid y] = \sum_{\text{branch } k \ a, b} \sum_{a, b} P(\gamma_k = a, \beta_k = b \mid y, t_k)
\]

\[
\cdot E[L_{ns, ts} \mid t_k, a, b],
\]

\[
E[N_{ts} \mid y] = \sum_{\text{branch } k \ a, b} \sum_{a, b} P(\gamma_k = a, \beta_k = b \mid y, t_k)
\]

\[
\cdot E[N_{ts} \mid t_k, a, b].
\]

The E-step thus consists of calculating conditional expectations of linear combinations of times such as \( E[L_{ns, ts} \mid t_k, a, b] \) and substitutions such as \( E[N_{ns} \mid t_k, a, b] \) where \( L_{ns, ts} \) and \( N_{ts} \) are given by Eq. (7.6). In our application \( n = 61 \) and the first type of statistics \( E[L_{ns, ts} \mid t, a, b] \) is (up to a factor \( p_{ab}(t) \)) on the form Eq. (7.4) with diagonal entries

\[
C_{ii} = \sum_j \pi_j k((i, j) \in L_{ns, ts})
\]

and all off diagonal entries equal to zero. The second type of statistics \( E[N_{ts} \mid t, a, b] \) is also on the form Eq. (7.4) with off-diagonal entries \( C_{ij} = q_{ij} k((i, j) \in L_{ts}) \) and zeros on the diagonal.

**Application 2:**

**Robust distance estimation**

The second application is a new approach for estimating labeled evolutionary distance, entitled robust counting and introduced in \([156]\). The purpose is to calculate a distance that is robust to model misspecification. The method is applied to labeled distances, for example, the synonymous distance between two coding DNA sequences. As it is believed that selection mainly acts at the protein level, synonymous substitutions are neutral and phylogenies built on these type of distances are more likely to reveal the true evolutionary history. The distance is calculated using the mean numbers of labeled substitutions conditioned on pairwise site patterns averaged over the empirical distribution of site patterns observed in the data. In the conventional method the average is done over the theoretical distribution of site patterns. The robustness is therefore achieved through the usage of more information from the data and less from the model.

Let \( Q \) be the rate matrix of the assumed model, \( P(t) = e^{Qt} \), the labeling be given through
a set of pairs $\mathcal{L}$ and the data be represented by a pairwise alignment $y = (y_1, y_2)$ of length $m$. As data only contains information about the product $Q_t$, where $t$ is the time distance between the sequences, we can set $t = 1$. Suppose we observe the complete data consisting of the types of substitutions that occurred in all sites and let $N_L = \sum_{i,j} N_{ij} \mathbb{I}(i, j) \in \mathcal{L}$ be the labeled number of substitutions. A natural labeled distance is given by $d_L = E[N_L]$. The labeled distance is estimated as the average across all sites of the expected number of labeled substitutions conditioned on the observed end points

$$d_L = \frac{1}{m} \sum_{s=1}^{m} E[N_L | X(0) = y_{1s}, X(1) = y_{2s}] = \frac{1}{m} \sum_{s=1}^{m} \left[ \left( \sum_{(i,j)} N_{ij} \mathbb{I}(i, j) \in \mathcal{L} \right) | 1, y_{1s}, y_{2s} \right].$$

Therefore this application requires evaluating a sum on the form Eq. (7.4) with off-diagonal entries $C_{ij} = q_{ij} \mathbb{I}((i, j) \in \mathcal{L})$ and zeros on the diagonal.

**Algorithms**

The calculation of $\Sigma(C; t)$ is based on the integrals $I_{cd}^{ab}(t)$. In this section we present three existing methods for obtaining the integrals and extend them to obtain $\Sigma(C; t)$.

**Eigenvalue decomposition (EVD)**

When the rate matrix $Q$ is diagonalizable, the computation of transition probabilities $p_{ab}(t)$ and integrals $I_{cd}^{ab}(t)$ can be done via the eigenvalue decomposition (EVD). EVD is a widely used method for calculating matrix exponentials. Let $Q = U \Lambda U^{-1}$ be the eigenvalue decomposition, with $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$. It follows that

$$P(t) = e^{Qt} = e^{(U \Lambda U^{-1})t} = U e^{\Lambda t} U^{-1}. \quad (7.7)$$

Because $\Lambda$ is diagonal, $e^{\Lambda t}$ is also diagonal with $(e^{\Lambda t})_{ii} = e^{\lambda_i t}$. Equation (7.3) becomes

$$I_{cd}^{ab}(t) = \sum_i U_{ai} (U^{-1})_{ic} \cdot \sum_j U_{dj} (U^{-1})_{jb} J_{ij}(t), \quad (7.8)$$

where

$$J_{ij}(t) = \begin{cases} 
    t e^{\lambda_i t} & \text{if } \lambda_i = \lambda_j, \\
    \frac{e^{\lambda_i t} - e^{\lambda_j t}}{\lambda_i - \lambda_j} & \text{if } \lambda_i \neq \lambda_j.
\end{cases} \quad (7.9)$$

Replacing $I_{cd}^{ab}(t)$ with Eq. (7.8) in Eq. (7.4), rearranging the sums and using

$$A_{ij} = \sum_d C_{cd} U_{dj}, \quad B_{ij} = J_{ij}(t) \sum_c (U^{-1})_{ic} A_{cj}, \quad D_{ib} = \sum_j B_{ij} (U^{-1})_{jb},$$

we find

$$\Sigma(C; a, b, t) = \sum_i U_{ai} D_{ib},$$

where $\circ$ represents the entry-wise product.

The eigenvalues and eigenvectors might be complex, but they come in complex conjugate pairs and the final result is always real; for more information we refer to the Supplementary Information in [105].

If the CTMC is reversible, the decomposition can be done on a symmetric matrix obtained from $Q$ (e.g. [82]), which is faster and tends to be more robust. Let $\pi$ be the stationary distribution. Due to reversibility, $\pi_a q_{ab} = \pi_b q_{ba}$, which can be written as $\pi Q = Q \pi$ where $\pi = \text{diag}(\pi)$. Let $S = \Pi^{1/2} Q \Pi^{-1/2}$. We have that

$$S^* = \Pi^{-1/2} Q^* \Pi^{1/2} = \Pi^{-1/2} (Q^* \Pi) \Pi^{-1/2} = \Pi^{-1/2} (\Pi Q) \Pi^{-1/2} = \Pi^{1/2} Q \Pi^{-1/2} = S,$$

where $S^*$ is the transpose of $S$. Then $S$ is symmetric. Let $\Lambda, V$ be its eigenvalues and eigenvectors, respectively. Then $V \Lambda V^{-1} = S = \Pi^{1/2} Q \Pi^{-1/2}$, which implies $Q = (\Pi^{-1/2} V) \Lambda (V^{-1} \Pi^{1/2})$ and it follows that $Q$ has the same eigenvalues as $S$ and $\Pi^{-1/2} V$ for eigenvectors.

The results can be summarized in the following algorithm.

**Algorithm 1: EVD**

**Input:** $Q, C, t$

**Output:** $\Sigma(C; t)$

Step 1: Determine eigenvalues $\lambda_i$ and the eigenvectors $U_i$ for $Q$, and compute $U^{-1}$.

Step 2: Determine $J(t)$ from Eq. (7.9).

Step 3: Determine $\Sigma(C; t)$ from Eq. (7.10).
7. Conditional expectations for CTMCs

Uniformization (UNI)

The uniformization method was first introduced in [97] for computing the matrix exponential \( P(t) = e^{Qt} \). In [85] it was shown how this method can be used for calculating summary statistics, even for statistics that cannot be written in integral form. Let \( \mu = \max_i (q_i) \) and \( R = \frac{1}{\mu} Q + I \), where \( I \) is the identity matrix. Then

\[
P(t) = e^{\mu(R-I)t} = \sum_{m=0}^{\infty} R^m \frac{(\mu t)^m}{m!} e^{-\mu t} \tag{7.11}
\]

Let \( \mu \) statistics that cannot be written in integral form.

We derive

\[
\Sigma(C; t) = \frac{1}{\mu} \sum_{m=0}^{\infty} \text{Pois}(m + 1; \mu t) \tag{7.13}
\]

\[
\sum_{l=0}^{m} R^l C R^{m-l}.
\]

The main challenge with this method is the infinite sum and we use Eq. (7.11) to determine a truncation point. In particular if we let \( \lambda = \mu t \) and truncate at \( s(\lambda) \) we can bound the error using the tail of the Poisson distribution

\[
|p_{ab}(t) - \sum_{m=0}^{\infty} (R^m)_{ab} \text{Pois}(m; \mu t)| \leq \sum_{m=s(\lambda)+1}^{\infty} \text{Pois}(m; \mu t).
\]

We have that, for large values of \( \lambda \), \( \text{Pois}(\lambda) \approx \mathcal{N}(\lambda, \lambda) \), where \( \mathcal{N}(\mu, \sigma^2) \) is the normal distribution with mean \( \mu \) and variance \( \sigma^2 \). Therefore, for large \( \lambda \), the error bound

\[
b = \sum_{m=s(\lambda)}^{\infty} \text{Pois}(m; \mu t)
\]

\[
\approx 1 - \Phi\left(\frac{s(\lambda) - \lambda}{\sqrt{\lambda}}\right),
\]

where \( \Phi(\cdot) \) is the cumulative distribution function for the standard normal distribution. Consequently, we can approximate the truncation point \( s(\lambda) \) with \( \sqrt{\lambda} \Phi^{-1}(1 - b) + \lambda \). If \( b = 10^{-8} \) we obtain \( \Phi^{-1}(1 - b) = 5.6 \).

Another way to determine \( s(\lambda) \) is to use \( R \) to evaluate \( \text{Pois}(m; \lambda) \) for values of \( m \) that gradually increase, until the tail is at most \( b = 10^{-8} \). Combining these two approaches, we performed a linear regression, approximating the tails from \( R \) to \( \lambda \). We obtained \( c_1 = 4.0731, c_2 = 5.6469, c_3 = 0.9963 \) but, in order to be conservative, we use \( s(\lambda) = \lfloor 4 + 6\sqrt{\lambda} + \lambda \rfloor \) where \( \lfloor x \rfloor \) is the smallest integer greater than or equal to \( x \). In Fig. 7.1 we compare the exact truncation value and the linear regression approximation. The linear regression provides an excellent fit to the tail of the distribution.

In summary, we have the following algorithm.

where Pois\( (m; \lambda) \) is the probability of \( m \) occurrences from a Poisson distribution with mean \( \lambda \).

Replacing Eq. (7.12) in Eq. (7.4), rearranging the sums and using that

\[
\sum_d C_{cd}(R^{m-l})_{db} = (CR^{m-l})_{cb} \tag{7.12}
\]

\[
\sum_e (R^l)_{ac} (CR^{m-l})_{cb} = (R^l CR^{m-l})_{ab},
\]

we derive

\[
\[\sum_{m=0}^{\infty} \text{Pois}(m + 1; \mu t)
\]

\[
\sum_{l=0}^{m} R^l C R^{m-l}.
\]
We are interested in the matrix exponential. This method for calculating Eq. (7.3) was developed in [204] and emphasized in [85]. Suppose we want to evaluate $I_{cd} = \sum_{m=0}^{\infty} R^m C R^{m-1}$ for $0 \leq m \leq s(\mu t)$, using that $A(m+1) = A(m)R + R^{m+1}C$.

**Algorithm 2: UNI**

**Input:** $Q, C, t$
**Output:** $\Sigma(C; t)$

Step 1: Determine $\mu, s(\mu t)$ and $R$.

Step 2: Calculate $R^m$ for $2 \leq m \leq s(\mu t)$.

Step 3: Calculate $A(m) = \sum_{l=0}^{m} R^l C R^{m-l}$.

Step 4: Determine $\Sigma(C; t)$ from Eq. (7.13).

**Exponentiation (EXPM)**

This method for calculating Eq. (7.3) was developed in [204] and emphasized in [85]. Suppose we want to evaluate $\int_{0}^{t} e^{Qu-Be^{Q(u-t)}du}$, where $Q$ and $B$ are $n \times n$ matrices. To calculate this integral, we use an auxiliary matrix $A = \begin{bmatrix} Q & B \\ 0 & Q \end{bmatrix}$ and the desired integral can be found in the upper right corner of the matrix exponential of $A$:

$$\int_{0}^{t} e^{Qu}Be^{Q(t-u)}du = (e^{At})_{1n, (n+1):2n}.$$  

We are interested in

$$I_{cd}^{ab}(t) = \int_{0}^{t} p_{ac}(u)p_{db}(t-u)du$$

where $I_{\{c,d\}}$ is a matrix with 1 in entry $(c, d)$ and zero otherwise. We can use this method to determine $I_{cd}^{ab}(t)$ by simply setting $B = I_{\{c,d\}}$.

**Algorithm 3: EXPM**

**Input:** $Q, C, t$
**Output:** $\Sigma(C; t)$

Step 1: Construct $A = \begin{bmatrix} Q & C \\ 0 & Q \end{bmatrix}$.

Step 2: Calculate $e^{At}$.

Step 3: $\Sigma(C; t)$ is the upper right corner of the matrix exponential.

**Testing**

We implemented the presented algorithms in R and tested them with respect to accuracy and speed.

**Accuracy**

The accuracy of the methods depends on the size of the rate matrix and the time $t$. To investigate how these factors influence the result, we used two different CTMCs that allow an analytical expression for Eq. (7.3). The first investigation is based on the Jukes-Cantor model where the rate matrix has uniform rates and variable size $n$:

$$q_{ij} = \begin{cases} -1 & \text{if } i = j \\ \frac{1}{n-1} & \text{if } i \neq j \end{cases}.$$
7. Conditional expectations for CTMCs

\[ Q \] has two unique eigenvalues: 0 with multiplicity 1 and \(-\frac{n}{n-1}\) with multiplicity \(n-1\). We obtain

\[
p_{ij}(t) = \begin{cases} 
\frac{1}{n} + \frac{n-1}{n} \exp\left(-\frac{nt}{n-1}\right) & \text{if } i = j \\
\frac{1}{n} - \frac{1}{n} \exp\left(-\frac{nt}{n-1}\right) & \text{if } i \neq j
\end{cases}
\]

\[ P_{cd}^{ab}(t) = \frac{1}{n^2} \]

\[
\begin{align*}
&\begin{cases} 
t + t \exp\left(-\frac{nt}{n-1}\right) & \text{if } a \neq c, \ d \neq b, \\
2(n-1) & \text{if } a = c, \ d = b,
\end{cases} \\
\begin{cases} 
t + (n-1)^2 t \exp\left(-\frac{nt}{n-1}\right) & \text{if } a = c, \ d = b, \\
2(n-1) & \text{if } a \neq c, \ d \neq b,
\end{cases} \\
\begin{cases} 
t - (n-1) t \exp\left(-\frac{nt}{n-1}\right) & \text{otherwise.}
\end{cases}
\end{align*}
\]

We compared the result from all three methods against the true value of Eq. (7.4) for size \(n\) ranging from 5 to 100, \(t = 0.1\) and random binary matrices \(C\). Entries in \(C\) are 1 with probability \(\frac{1}{2}\). For each fixed size, we generated 5 different matrices \(C\). The average normalized deviation is shown in Fig. 7.2.

The second CTMC is the HKY model with the stationary distribution \(\pi = (0.2, 0.2, 0.3, 0.3)\) and the transition/transversion rate ratio \(\kappa = 2.15\). The rate matrix

\[
Q = \begin{pmatrix}
\kappa & \pi_R & \pi_I \\
\pi_R & \pi_C & \pi_T \\
\pi_I & \pi_C & \pi_T
\end{pmatrix}
\]

has an analytic result for Eq. (7.3) which can be obtained through the eigenvalue decomposition. The eigenvalues and eigenvectors of \(Q\) are

\[
\lambda = (0, -1, -\pi_Y, -\pi_R, -\pi_R, -\pi_Y) ,
\]

\[
U = \begin{pmatrix}
1 - \pi_Y & 0 & -\pi_Y \\
1 - \pi_Y & 0 & 1 \\
1 & -\pi_Y & 0 \\
1 & 1 & 0
\end{pmatrix}
\]

\[
U^{-1} = \begin{pmatrix}
\pi_A & \pi_G & \pi_C & \pi_T \\
-\pi_A & -\pi_G & \pi_C & \pi_T \\
0 & 0 & \pi_C & \pi_T \\
-\pi_R & \pi_A & \pi_R & 0
\end{pmatrix}
\]

where \(\pi_R = \pi_A + \pi_G\) and \(\pi_Y = \pi_C + \pi_T\). From this, using the symbolic operations in Matlab [141], we obtained the final analytic expression for Eq. (7.3). Using this model we compared for all three methods the true value of Eq. (7.4) for various values of \(t\) and randomly generated binary matrices \(C\). For each \(t\) we generated 5 different matrices \(C\). The average normalized deviation is shown in Fig. 7.2.

In both cases, all methods showed good accuracy as the normalized deviation was no bigger than \(3 \times 10^{-9}\). We also note that EXPM tended to be the most precise while UNI provided the worst approximation.

To further investigate the accuracy, we performed calculations on randomly generated reversible rate matrices: we first obtained the stationary distribution from the Dirichlet distribution with parameter \(\alpha\) and finally calculated the remaining entries using the reversibility property. In all the runs the relative difference between EVD, UNI and EXPM was less than \(10^{-4}\). This indicated that all three methods have a similar performance in a wide range of applications.

**Speed**

**Partition of computation** Assume we need to evaluate \(\Sigma(C; t)\) for a fixed matrix \(C\) and multiple time points \(t \in \{t_1, \ldots, t_k\}\). In each iteration of the EM-algorithm in Application 1 we need this type of calculation while in order to calculate the labeled distance in Application 2 just one time point is required.

Using EVD (Algorithm 1) we do the eigenvalue decomposition (Step 1) once and then, for each time point \(t_i\), we apply Step 2 and Step 3. The eigenvalue decomposition, achieved through the R function `eigen`, has a running time of \(O(n^3)\). In Step 2 we determine \(J(t)\) and this takes \(O(n^2)\) time. Step 3 has a running time of \(O(n^3)\) due to the matrix multiplications.

If instead we apply UNI (Algorithm 2), we run Steps 1–3 for the largest time point \(\max(t_i)\) and then, for each time point \(t_i\), we apply Step 4. Steps 1–3 take \(O(s(\mu \max(t_i))n^3)\) time, and Step 4 takes \(O(s(\mu t_i)n^2)\) time for each \(i \in \{1, \ldots, k\}\). Therefore, even though the total time for both methods is \(O(n^3)\), the addition of one time point contributes with \(O(n^3)\) for EVD, but only \(O(s(\mu t)n^2)\) for UNI. Recall that the constant \(s(\mu t)\) is the truncation point for the infinite sum in the uniformization method.

In the case of EXPM (Algorithm 3) we need to calculate the matrix exponential at every sin-
7.2. Results

Figure 7.2: Accuracy results. Accuracy has been tested using JC and HKY models. For each run, the normalized deviation is calculated: 

\[
\frac{\hat{\Sigma}(C; a, b, t) - \Sigma(C; a, b, t)}{\Sigma(C; a, b, t)},
\]

where \(\Sigma\) is the correct value and \(\hat{\Sigma}\) is the calculated one. Each plotted point represents the average over \(a, b\) and 5 different randomly generated matrices \(C\) as described in the main text.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precomputation</th>
<th>Main Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Steps</td>
<td>Order</td>
</tr>
<tr>
<td>EVD</td>
<td>1</td>
<td>(O(n^3))</td>
</tr>
<tr>
<td>UNI</td>
<td>1–3</td>
<td>(O(s(\mu t)n^3))</td>
</tr>
<tr>
<td>EXPM</td>
<td>none</td>
<td>none</td>
</tr>
</tbody>
</table>

Table 7.1: Running time complexity.

The first experiment corresponded to running the EM algorithm on real data consisting of DNA sequences from the HIV pol gene described in [120]. HIV has been extensively studied with respect to selection pressure and drug resistance and in [120] the authors document convergent evolution in pol gene caused by drug resistance mutations. The observed data \(y\) was a multiple codon alignment of the sequences. For simplicity, we did not consider the columns with gaps or ambiguous nucleotides. To compare the performance of the methods as a function of the size of the data set, we applied the EM algorithm for 15 data sets containing from 2 up to 16 sequences each, extracted from the HIV pol gene data. For each set we assumed the sequences were related according to a
Figure 7.3: Experiments results. Running times for the eight experiments. Experiment 1: rate matrix estimation using EM. The plot shows the running time for calculating the statistics for each method, as a function of the number of sequences included in the data set. For experiments 2 and 3 we calculated the value of $\Sigma(C; t_k)$ for 10 time points $t_k$. Each plot starts with the running time of the precomputation and at position $k$ we plot the cumulative running time for precomputation and the evaluation of $\Sigma(C; t_i)$ for all $t_i \in \{t_1, \ldots, t_k\}$. The values of $t_k$ are provided in Table 7.2. Experiment 4: robust distance estimation. The plot shows the running time for computing the robust distance as a function of the evolutionary distance $t$. Experiments 5-8: similar as for experiments 2 and 3 but with a GTR model (experiments 5 and 6) and UNR model (experiments 7 and 8) instead of a GY model.

fixed tree; we have reconstructed the phylogenies in Mega [194] using the Jukes-Cantor model and Neighbor-Joining. We ran the EM algorithm until all three parameters converged. Experiments two and three used the previously estimated matrix $Q$ given by Eq. (7.5) with $\alpha = 10.5, \kappa = 4.27$ and $\omega = 0.6$. We let $C_{ij} = q_{ij}$ and $C_{ii} = 0$, corresponding to calculating the total number of expected substitutions $E[N \mid t, a, b]$, and computed the value of $\Sigma(C; t_k)$ for 10 equidistant sorted time points $t_k$ with $1 \leq k \leq 10$ (Table 7.2).

GTR In the fourth experiment we estimated the robust labeled distance of two sequences, using the same set-up as in [156]. We used the general time reversible (GTR) model with

$$Q = \begin{pmatrix}
q_{11} \pi_1 & q_{12} \pi_2 & q_{13} \pi_3 \\
q_{21} \pi_4 & q_{22} \pi_4 & q_{23} \pi_5 \\
q_{31} \pi_6 & q_{32} \pi_6 & q_{33} \pi_6 \\
q_{41} \pi_7 & q_{42} \pi_7 & q_{43} \pi_7 \\
q_{51} \pi_8 & q_{52} \pi_8 & q_{53} \pi_8 \\
q_{61} \pi_9 & q_{62} \pi_9 & q_{63} \pi_9 \\
q_{71} \pi_{10} & q_{72} \pi_{10} & q_{73} \pi_{10} \\
q_{81} \pi_{11} & q_{82} \pi_{11} & q_{83} \pi_{11} \\
q_{91} \pi_{12} & q_{92} \pi_{12} & q_{93} \pi_{12} \\
q_{101} \pi_{13} & q_{102} \pi_{13} & q_{103} \pi_{13}
\end{pmatrix},$$

where $r = (0.5, 0.3, 0.6, 0.2, 0.3, 0.2)$ and $\pi = (0.2, 0.2, 0.3, 0.3)$. For each considered evolution-
### 7.2. Results

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Experiment</th>
<th>$t_k$</th>
<th>$\mu t_k$</th>
<th>$s(\mu t_k)$</th>
<th>$t_k$</th>
<th>$\mu t_k$</th>
<th>$s(\mu t_k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GY</td>
<td>2</td>
<td>0.0017</td>
<td>0.0045</td>
<td>5</td>
<td>0.1</td>
<td>0.2668</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.0032</td>
<td>0.0085</td>
<td>5</td>
<td>0.2</td>
<td>0.5337</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.0046</td>
<td>0.0124</td>
<td>5</td>
<td>0.3</td>
<td>0.8005</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.0061</td>
<td>0.0163</td>
<td>5</td>
<td>0.4</td>
<td>1.0674</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.0076</td>
<td>0.0202</td>
<td>5</td>
<td>0.5</td>
<td>1.3342</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.0090</td>
<td>0.0241</td>
<td>5</td>
<td>0.6</td>
<td>1.6010</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.0105</td>
<td>0.0281</td>
<td>6</td>
<td>0.7</td>
<td>1.8679</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>0.0120</td>
<td>0.0320</td>
<td>6</td>
<td>0.8</td>
<td>2.1347</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.0135</td>
<td>0.0359</td>
<td>6</td>
<td>0.9</td>
<td>2.4015</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0150</td>
<td>0.0398</td>
<td>6</td>
<td>1.0</td>
<td>2.6684</td>
<td>17</td>
</tr>
<tr>
<td>GTR</td>
<td>5</td>
<td>0.1760</td>
<td>0.2668</td>
<td>8</td>
<td>0.1</td>
<td>0.1516</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.3520</td>
<td>0.5337</td>
<td>9</td>
<td>0.6</td>
<td>0.9098</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.5280</td>
<td>0.8005</td>
<td>11</td>
<td>1.1</td>
<td>1.6680</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.7039</td>
<td>1.0674</td>
<td>12</td>
<td>1.6</td>
<td>2.4262</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>0.8798</td>
<td>1.3342</td>
<td>13</td>
<td>2.1</td>
<td>3.1844</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>1.0558</td>
<td>1.6010</td>
<td>14</td>
<td>2.6</td>
<td>3.9426</td>
<td>20</td>
</tr>
<tr>
<td>UNR</td>
<td>7</td>
<td>1.2318</td>
<td>1.8679</td>
<td>15</td>
<td>3.1</td>
<td>4.7008</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1.4077</td>
<td>2.1347</td>
<td>15</td>
<td>3.6</td>
<td>5.4590</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>1.5837</td>
<td>2.4015</td>
<td>16</td>
<td>4.1</td>
<td>6.2172</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>1.7597</td>
<td>2.6684</td>
<td>17</td>
<td>4.6</td>
<td>6.9754</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0379</td>
<td>0.1516</td>
<td>7</td>
<td>0.1</td>
<td>0.4</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.2275</td>
<td>0.9098</td>
<td>11</td>
<td>0.6</td>
<td>2.4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.4170</td>
<td>1.6680</td>
<td>14</td>
<td>1.1</td>
<td>4.4</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.6066</td>
<td>2.4262</td>
<td>16</td>
<td>1.6</td>
<td>6.4</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.7961</td>
<td>3.1844</td>
<td>18</td>
<td>2.1</td>
<td>8.4</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.9857</td>
<td>3.9426</td>
<td>20</td>
<td>2.6</td>
<td>10.4</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>1.1752</td>
<td>4.7008</td>
<td>22</td>
<td>3.1</td>
<td>12.4</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1.3648</td>
<td>5.4590</td>
<td>24</td>
<td>3.6</td>
<td>14.4</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>1.5543</td>
<td>6.2172</td>
<td>26</td>
<td>4.1</td>
<td>16.4</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>1.7439</td>
<td>6.9754</td>
<td>27</td>
<td>4.6</td>
<td>18.4</td>
<td>49</td>
</tr>
</tbody>
</table>

Table 7.2: **Experimental design.** The table shows the time points $t_k$, $\mu t_k$ and the approximation of the Poisson tail $s(\mu t_k)$. For experiment 2, $t_k$ spanned the interval that contains the 10 longest branch lengths from the phylogeny of the 16 HIV pol sequences. In experiment 3 we started at 0.1 and ended at 1. We wished to design experiment 5 such that the corresponding $s(\mu t_k)$ was the same as $s(\mu t_k)$ from experiment 3. This allowed us to illustrate the relative performance of the methods when running on different sizes of the rate matrix. Experiment 6 was done on time points starting from 0.1 and ending at 4.6. As before, we wished to design experiment 7 such $s(\mu t_k)$ corresponded to experiment 6. Experiment 8 used the same $t_k$ as experiment 6.
ary distance $t$ between 0.1 and 1, we generated 50 pairwise sequence data sets of length 2000 which have evolved for time $t$ under the GTR model.

For labeling, we considered the jumps to and from nucleotide A, leading to $C_{ij} = q_{ij}$ if $i$ or $j$ represents nucleotide A. For each data set, we estimated the GTR parameters as described in [156] and calculated the robust distance. Experiments 5 and 6 used the same GTR matrix and $C_{ij} = q_{ij}$ if $i$ or $j$ represents nucleotide A and zero otherwise, and computed the value of $\Sigma(C; t_k)$ for 10 equidistant sorted time points $t_k$ with $1 \leq k \leq 10$ (Table 7.2).

**UNR** In the last two experiments we used the same set-up as in experiments 5 and 6 but with a different matrix and time points (Table 7.2). As the speed of EVD is influenced by the type of the model, we decided to employ a non-reversible matrix. We chose the unrestricted model and carefully set the rates such that the matrix has a complex decomposition

$$Q = \begin{pmatrix} -4 & 2 & 1 & 1 \\ 0 & -3 & 2 & 1 \\ 1 & 0 & -3 & 2 \\ 2 & 1 & 1 & -4 \end{pmatrix}.$$ 

Figure 7.3 shows the results. For experiments 1 and 4, the plots show the recorded running time under each set-up (different number of sequences or different evolutionary distance). For the remaining experiments each plot starts with the running time of the precomputation which, for UNI, is done on the largest time point $t_{10}$. Then, at position $k$, we plot the cumulative running time for precomputation and the evaluation of $\Sigma(C; t_i)$ for all $i \leq k$. Since EVD and EXPM have running times that are independent of $t_k$, the running times for these two algorithms are the same in experiments 2 and 3, 5 and 6, and 7 and 8. Even more, as EXPM is dependent only on the size of the matrix, the running times in experiments 5 - 8 are the same.

We observe that in all our experiments EXPM is the slowest method. Deciding if EVD or UNI is faster depends on the size and type of the matrix, the number of time points and the values of $s(\mu t)$. As the main computation for UNI has a running time of $O(n^3)$ as opposed to $O(n^5)$ for EVD (Table 7.1), this method should have an increased advantage when the rate matrix is bigger. This means that if many time points are considered, then UNI is generally the faster method. Importantly, we note that the EVD precomputation tends to be faster than the UNI precomputation. We remark that, in the first experiment, UNI proved to be the fastest method while, in the fourth experiment, UNI became slower with the increase of the evolutionary distance between the sequences and it was only faster than EVD for small distances ($< 0.2$). By setting $t_k$ in an appropriate manner (Table 7.2), we have the same running time for UNI and EXPM for experiment 7 compared to experiment 6. Due to the fact that in experiment 7 we used the UNR matrix, EVD is slower as opposed to experiment 6. In this case, the difference is observable but not very big, but as the size of the matrix increases, this discrepancy increases too. We also note that the difference between the reversible and non-reversible cases is enough to make UNI faster than EVD in the latter case.

### 7.3 Discussion

The EVD algorithm assumes that the rate matrix is diagonalizable. However, a direct calculation of Eq. (7.3) in the non-diagonalizable case is actually possible using the Jordan normal form for the rate matrix. Let $Q = JJP^{-1}$ where $J$ is the Jordan normal form of $Q$ and $P$ consists of the generalized eigenvectors (we recognize that we used $P$ and $J$ for other quantities earlier but for this discussion this should not cause any confusion and we prefer to use standard notation), i.e. $J$ has a block diagonal form $J = diag(J_1, \ldots, J_K)$ where $J_k = \lambda_k I + N$ is a matrix with $\lambda_k$ on the diagonal and 1 on the superdiagonal. In order to calculate Eq. (7.3), we use that

$$e^{Qt} = P\text{diag}(e^{tJ_1}, \ldots, e^{tJ_K}) P^{-1},$$

and noting that $N$ is a nilpotent matrix with degree $d_k$ (equal to the size of block $J_k$),

$$e^{tJ_k} = e^{t\lambda_k} e^{tN} = e^{t\lambda_k} \sum_{i=0}^{d_k-1} \frac{t^i}{i!} N^i.$$

It is evident that this procedure is feasible but also requires much bookkeeping.

In [205] an extension of uniformization, adaptive uniformization, is described for calculating transition probabilities in a CTMC. The basic idea is to perform a local uniformization instead of a global uniformization of the rate matrix and thereby have fewer jumps in the jump process. van Moorsel and Sanders [205] consider a model with
rate matrix
\[
Q = \begin{pmatrix}
-3\nu & 3\nu & 0 & 0 \\
\mu & -(\mu + 2\nu) & 2\nu & 0 \\
0 & \mu & -(\mu + \nu) & \nu \\
0 & 0 & 0 & 0 \\
\end{pmatrix},
\]
(state 4 is an absorbing state). If this process starts in state 1 then the first jump is to state 2 and the second is from state 2 to either state 1 or state 3. This feature can be taken into account by having a so-called adaptive uniformized (AU) jump process where the rate for the first jump is 3\nu, for the second is \mu + 2\nu and, assuming \mu + \nu > 3\nu, the rate for the third jump is \mu + \nu. From the third jump the rate in the AU jump process is \mu + 2\nu as in the standard uniformized jump process. The AU jump process has a closed-form expression for the jump probabilities (it is a pure birth process) but is of course more complicated than a Poisson jump process. The advantage is that the AU jump process exhibits fewer jumps. This procedure could very well be useful for codon models where the set of states that the process can be in after one or two jumps are limited because only one nucleotide change is allowed in each state change.

In an application concerned with modeling among-site rate variation, Mateiu and Rannala [140] apply the uniformization procedure to calculate the transition probabilities instead of the eigenvalue decomposition method. Mateiu and Rannala [140] show, in agreement with our results, that uniformization is a faster computational method than eigenvalue decomposition.

The presented methods are not the only ones for calculating the desired summary statistics. For example, in [49] it is suggested to determine the expected number of jumps from the direct calculation
\[
p_{ab}(t) = \int_0^t e^{Qs} q_{cd}\left(e^{Q(t-s)}q_{ac}\right) ds
\]
where the infinite sum is truncated at \( k = 10 \). The problem with this approach is that it is difficult to bound the error introduced by the truncation. In UNI a similar type of calculation applies but the truncation error can be controlled.

7.4 Conclusions
Recall that EVD assumes that the rate matrix is diagonalizable and this constraint means that EVD is less general than the other two algorithms. We have shown in the Discussion how a direct calculation of the integral (4) is actually still possible but requires much bookkeeping. On top of being less general, EVD is dependent on the type of the matrix: reversible or non-reversible. We have shown how this discrepancy can make EVD slower than UNI even when the state space has size of only 4.

We found that the presented methods have similar accuracy and EXPM is the most accurate one. With respect to running time, it is not straightforward which method is best. We found that both the eigenvalue decomposition (EVD) and uniformization (UNI) are faster than the matrix exponentiation method (EXPM). The main reason for EVD and UNI being faster is that they can be decomposed into a precomputation and a main computation. The precomputation only depends on the rate matrix for EVD while for UNI it also depends on the largest time point and the matrix \( C \). We also remark that EXPM involves the exponentiation of a matrix double in size. UNI is particularly fast when the product \( \mu t \) is small because in this case only a few terms in Eq. (7.13) are needed.

Authors’ contributions
PT extended the existing methods to linear combinations of statistics, implemented the algorithms and performed the testing. AH conceived the study and guided the development and evaluation of the methods. Both authors wrote the paper. All authors read and approved the final manuscript.

Acknowledgments
We are grateful to Thomas Mailund and Julien Y. Dutheil for very useful discussions on the presentation and implementation of the algorithms. We would also like to thank the anonymous reviewers for constructive comments and suggestions that helped us improve the paper.
Motif analysis has long been an important method to characterize biological functionality and the current growth of sequencing-based genomics experiments further extends its potential. These diverse experiments often generate sequence lists ranked by some functional property. There is therefore a growing need for motif analysis methods that can exploit this coupled data structure and be tailored for specific biological questions. Here, we present a motif analysis tool, Regmex (REGular expression Motif EXplorer), which offers several methods to identify overrepresented motifs in a ranked list of sequences. Regmex uses regular expressions to define motifs or families of motifs and embedded Markov models to calculate exact probabilities for motif observations in sequences. Motif enrichment is optionally evaluated using random walks, Brownian bridges, or modified rank based statistics. These features make Regmex well suited for a range of biological sequence analysis problems related to motif discovery. We demonstrate different usage scenarios including rank correlation of microRNA binding sites co-occurring with a U-rich motif. The method is available as an R package.
8. Motif discovery in ranked lists of sequences

8.1 Introduction

Motif discovery is a classical problem in sequence analysis and its scope broadens with modern sequencing technologies. A large number of tools are designed to find enriched motifs in sequences, with the majority aimed at finding motifs that are enriched in a foreground set of sequences relative to a background set. This is optimal for sequences where a categorical variable defines a foreground and a background. However, many experimental settings are associated with continuous variables where set-based methods are suboptimal. Instead of using a hard threshold to divide a continuous variable into foreground and background, it is more powerful to take the magnitude of the continuous variable directly into account.

More recently, motif enrichment methods have been developed that can exploit the ranking in a list of sequences, e.g. [56, 119, 170, 191, 203]. These methods seek to find the motifs that best correlate with the ranked sequence list. Most commonly, this is achieved by exhaustively searching through the space of all simple motifs of a given length k (k-mers). K-mers, ranked by their correlation measures, are then either output directly, clustered and used to define position weight matrices (PWMs) or used as seeds in a variety of downstream algorithms to refine the top correlating motifs.

A general challenge of motif analysis, and specifically of methods based on an exhaustive search, is the rapid increase in search space with motif size and complexity. This problem has been addressed in recent work by using suffix trees, allowing exhaustive searches of large spaces such as all variable gap motifs up to a given length [119]. However, functional motifs may display a much higher degree of complexity than current methods meet. Many snoRNAs, for example, are known to bind their targets in a composite motif consisting of two binding sites separated by a variable number of nucleotides. In addition, regulation of biological systems often rely on multiple factors acting in concert. For instance, endogenous RNAs can severely perturb regulatory networks of microRNAs [163]. It is thus valuable to be able to evaluate enrichments for subsets of binding sites in combination.

A central aspect in motif analysis of ranked sequences is the significance evaluation of the motif rank correlation. A number of approaches have been used, including linear regression models [29], Wilcoxon rank sum tests [188], a Kolmogorov-Smirnov based approach [98], a Brownian bridge based approach [170] and methods using variants of hyper geometric tests [56, 119, 203]. The various methods also have different standards for motif scoring in the sequences. Examples include simple presence/absence scores for each sequence [56, 98], dependence of sequence lengths and global base composition [203] and probabilistic scoring that models base composition of every sequence in the rank list [170].

Presence/absence scores in particular suffer a risk of bias because sequence length and composition is not included in the score model, which is a problem if e.g. sequence lengths are biased in the rank. Also, presence/absence score-based methods may be under-powered in situations where the number of motif occurrences in a single sequence matters.

Based on these issues, we see a need for a tool that allows hypotheses for flexible motifs to be evaluated, and calculates accurate sequence dependent p-values for motif observations. We present Regmex, a motif enrichment tool, with a number of new features aimed at accurate significance evaluation (see Fig. 8.1). First, we calculate sequence specific motif p-values that depend on both sequence lengths and base compositions using an embedded Markov model. Second, depending on the problem and hypothesis, motif

![Figure 8.1: Flow diagram of the procedures for calculating sequence specific p-values and rank correlation or clustering p-value in Regmex.](image-url)
8.2 Material and methods

The Regmex tool

In this study, we introduce Regmex, a motif analysis tool available as an R package. Regmex is designed with flexibility in mind to study rank correlation or clustering of motifs in a list of sequences. Briefly, it takes as input a list of sequences ranked by an experimental setting, and one or more motifs, each defined as a regular expression (RE) (see Fig. 8.1). The output, in its simplest form, contains the rank correlation or clustering \( RCPs \) for the input motifs. Alternatively, it is possible to get the underlying sequence specific \( p \)-values (SSPs) for motifs as well as count statistics etc.

To illustrate the power of REs in a biological sequence context, we consider the following examples:

1. A stem loop structure TTTCNNNGAAA found in the 3’UTR of many key inflammatory and immune genes [158]. Although this is a simple RE, it captures 64 11-mers in one expression, and Regmex reports the rank correlation \( p \)-value of the combined set.

2. A G-quadruplex structure, GGGLGGGLGGGLGGG, \( L = \text{[N|NN|NNN|NNNN]} \). This is found e.g. in telomeric regions [18].

3. Any size open reading frame, given by ATG(NNN)*((TGA)|(TAA)|(TAG)). This RE is an example of an enormous set, which would be difficult to obtain without a RE.

We note that an advantage of REs in relation to the motif enrichment problem is that a RE can be obtained for any set of simple motifs. Thus for example a set of experimentally verified binding sequences can be expressed as a RE, and matching will include exactly this set.

Sequence specific motif

\( p \)-value calculation

A central point in the way Regmex calculates a motif RCP is to calculate SSPs for observing the motif the observed number of times (\( n_{\text{obs}} \)) or more. Briefly, from a deterministic finite state automaton (DFA) associated with the RE motif, we identify a sequence specific transition probability matrix (TPM) which is used to build an embedded TPM (eTPM) specific for \( n_{\text{obs}} \) (see Fig. 8.2). The SSP is subsequently read from the eTPM raised to the power of the sequence length.

Deterministic finite state automaton

For any RE, the corresponding DFA can be built, which is the initial step in the SSP calculation (Fig. 8.2 B). The DFA starts in an initial state, accepts symbols (i.e. nucleotides) on the edges and moves through the states. The end state corresponds to having observed the RE. The DFA used here recognizes an extended regular expression, as described in [196]. The routine used to build the DFA for a given RE is implemented in Java, using [148], and supports standard regular expression operations (concatenation, union and Kleene star).

Markov embedding

The DFA graph structure can also be thought of as a Markov model, where instead of accepting symbols, it generates symbols on the edges with probabilities corresponding to the base frequencies in a given sequence. The Markov model can be represented by a transition probability matrix (TPM), which holds the probabilities of moving between states of the DFA given a randomly picked base from the sequence (Fig. 8.2 C). TPM\(^n\) will hold the probability of moving between states given \( n \) bases.

We are interested in the SSP and thus need to have a probability model that takes \( n_{\text{obs}} \) into account. Regmex does this by making a model expansion using the DFA as a template. We refer to this as an embedded DFA (eDFA) (Fig. 8.2...
Figure 8.2: (A) Motif in the form of a regular expression. Base coloring applies throughout the figure. (B) Deterministic finite state automaton corresponding to the regular expression in (A). Initial state is indicated in gray, end state is indicated by a double circle. (C) Transition state probability matrix (TPM) associated with the model in (B). (D) Embedded Markov Model (eDFA) for two observed occurrences of the motif. States are pre-indexed with number of prior motif observations. (E) Embedded transition state probability matrix (eTPM) associated with the eDFA. The yellow matrix is an exact copy of the yellow matrix from (C). The gray entries have zero probability. The end state transition probabilities of the DFA model (red/orange in (C)) are shifted forward and contain the initial state of the next motif occurrence, except for any end to end transition probability (occurs for REs ending with a *), which remains in the DFA template (red field). The final state of the eDFA (2,4 in D) is an absorbing state and all transition probability is in (2,4;2,4) indicated in black. (F) Heat diagrams of the \( n \)-step eTPM reflecting the probability of moving between states in the eDFA given a random sequence of length \( n \) with a specific base composition. The row corresponding to the initial state (0,1) holds the probability distribution of going from the start state to any state in the eDFA in \( n \) steps. The last entry of this row (red field) holds the probability of the observed number of motifs (\( n_{\text{obs}} \)) or more in the sequence (the SSP).

Specifically, the template DFA is copied \( n_{\text{obs}} \) times and outgoing edges of the end state(s) of the DFA template are moved to the corresponding states in the next template copy. This effectively allows the embedded model to count how many times the RE motif has been observed. The final state of the eDFA is absorbing, so no further motif observations are scored.

Again, the eDFA can be thought of both as an automaton accepting symbols or as a Markov model generating symbols on edges. As above, Regmex constructs a transition probability matrix (eTPM) based on the eDFA (Fig. 8.2 E). The eTPM\(^n\) holds probabilities of moving between states of the eDFA given a random sequence of length \( n \) with the observed base frequencies (Fig. 8.2 F). We can now extract the probability distribution of the RE motif in a given sequence by reading the row corresponding to the initial state (0,1) in the eTPM\(^n\). In particular the probability of observing the motif \( n_{\text{obs}} \) number of times or more (SSP) can be read in the final state column of the initial state row (red field in Fig. 8.2 F).

**Motif rank correlation \( p \)-value**

In the downstream analysis, Regmex uses the calculated SSPs when calculating the RCP. Because the characteristics of rank correlation may vary depending on the problem analyzed, the choice of method used to evaluate the correlation may differ in detection power. E.g. one test may have higher power for detecting long motifs occurring rarely in the sequence list and another may have superior
8.2. Material and methods

Figure 8.3: (A) Sequences enriched with a 7-mer motif (ACGTGAT) as indicated with red marks. Upper bars indicate sequence lengths, lower bars indicate SSPs for the motif. (B) Brownian bridge for the 7-mer motif in (A) (red) and for 500 random 7-mer motifs (gray). The RCP corresponding to the BB is indicated. (C) Random walk for the motif in (A) (red) and 500 random 7-mer motifs (black). The RCP corresponding to the RW is indicated. (D) Schematic of the MSR method. Lines represent sequences with lengths proportional to the probability of observing the motif one or more times. A motif occurrence is marked by an asterix. The RCP corresponding to the motif distribution is indicated.

sensitivity for frequent short motifs. In Regmex, we have implemented three methods for evaluating motif rank correlation or motif clustering, which have different strengths. These methods are based on Brownian bridge (BB), random walk (RW) and modified sum of rank (MSR) statistics. Figure 8.3 illustrates the concept underlying each of these statistics on a small list of 50 sequences with an enriched motif.

Brownian bridge method

This method is a re-implementation of the method developed by Jacobsen et al. [95] and recently implemented in cWords [170]. Our implementation differs in the calculation of the SSPs and in how we calculate the RCP (see supplemental methods for details). Briefly, the method calculates the max value $D$ of a running sum of mean adjusted log scores of the SSPs. The running sum starts and ends in zero and hence is a Brownian bridge under the null model (see Fig. 8.3 B). We identify the $p$-value from the analytical distribution of max values of a Brownian bridge.

Random walk statistic

The RW method is similar to the use of random walks in the BLAST algorithm [4]. This method is sensitive to clustering of motifs anywhere in the sequence list. The SSPs for a motif are transformed into steps in a walk (see Fig. 8.3 C). Under the null model the motif is not enriched and SSPs follow the uniform distribution. The SSPs are transformed into steps according to a scoring scheme where small $p$-values (SSPs) corresponds to a positive step and large $p$-values corresponds to a negative step. The exact scoring scheme is based on assumed motif densities in the foreground relative to the background, so that higher motif densities give rise to a higher walk in local regions of the sequence list. The RW starts over from zero every
8. Motif discovery in ranked lists of sequences

Figure 8.4: (A) Comparison of p-value output for the different rank correlation methods used in Regmex. One 7-mer motif (ACGTGAT) is inserted as indicated in the first half of 1000 sequences each with a length of 1000 bases. In replicates with no insertion the BB method was used, but the other methods gave similar results. Error bars indicate standard error of 100 replicates. (B) Up to four different 7-mer motifs inserted randomly in the first half of the sequences in (A). p-value output from Regmex using the BB method plotted against the number of inserted motifs. RE motifs define sets of one up to all four 7-mers as indicated, e.g. m1 | m2 = (ACGTGAT)|(GCATTGT). (C) p-value output from Regmex using the BB method plotted against the number of inserted motifs. Sets of four 7-mer motifs were inserted at fixed positions randomly among the first half of the sequences in (A), so that motifs occur together in the same sequences. RE motifs define sets of combinations of one up to all four 7-mers as indicated, e.g. m1 & m2 = (ACGTGAT^*GCATTGT)|(GCATTGT^*ACGTGAT), where N denotes any nucleotide.

time it reaches the lower bound of -1. This makes the RW method sensitive to local runs of enriched motifs in the sequence list. For significance evaluation, we find the probability of a walk with at least as high a max value under the null distribution. We do this using a recursion on an analytic expression for the max value distribution of random walks (see supplemental methods for details). Alternatively, we can use a geometric-like distribution (Gumbel distribution) as an approximation for the max value distribution [59].

Modified sum of ranks statistics

The MSR method is based on the idea of using a sum of rank test to determine a rank bias in motif containing sequences. Rather than summing ranks, MSR uses a sum of scores specific for the
8.3 Results

Combined motifs increase power

Because of different characteristics of the three methods for rank correlation evaluation, they perform differently in different scenarios. Figure 8.4 A illustrates their behavior when applied to a set of 1000 random sequences with a simple 7-mer motif inserted up to 100 times in the upper half of the sequence list. In this particular scenario, the RW approach has the highest sensitivity, followed by the BB method and the MSR method (Fig. 8.4 A). The RW method generally has a high sensitivity when the motif density is high, regardless of where in the sequence list it occurs. This is in contrast to both the MSR and BB methods, which are more sensitive to enrichment in the beginning or end of the sequence list. The rank sum derived nature of the MSR method yields a higher sensitivity for enrichment in the ends of longer sequence lists, while the BB method is highly superior in short sequence lists with moderate enrichment. (see Fig. 8.5).

When using SSPs rather than e.g. binary scores for motif observations in sequences, the benefit of differential scoring becomes clear. This means, e.g., that rank correlation of common and individually insignificant motifs can be better evaluated because their impact on the rank correlation is moderated by the significance of the observation. The same argument applies to rare, highly significant motifs. This, combined with RE motif definition, is useful in the case of evaluating rank correlation of combinations of motifs.

We used Regmex to evaluate rank correlation of combinations of inserted motifs in a set of ran-
8. **Motif discovery in ranked lists of sequences**

<table>
<thead>
<tr>
<th>microRNA</th>
<th>Seed Target(S)</th>
<th>(SN*U)</th>
<th>(UN*S)</th>
<th>UN*S</th>
<th>SN*U</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>miR-7</td>
<td>2.6e−03</td>
<td>1.5e−13</td>
<td>2.4e−11</td>
<td>9.6e−05</td>
<td>[72]</td>
<td></td>
</tr>
<tr>
<td>miR-9</td>
<td>6.6e−09</td>
<td>1.5e−17</td>
<td>2.6e−19</td>
<td>1.2e−05</td>
<td>[72]</td>
<td></td>
</tr>
<tr>
<td>miR-16</td>
<td>1.8e−178</td>
<td>7.3e−147</td>
<td>5.7e−65</td>
<td>9.2e−76</td>
<td>[124]</td>
<td></td>
</tr>
<tr>
<td>miR-106b</td>
<td>2.5e−99</td>
<td>9.7e−158</td>
<td>4.5e−145</td>
<td>1.8e−58</td>
<td>[124]</td>
<td></td>
</tr>
<tr>
<td>miR-122a</td>
<td>3.2e−02</td>
<td>4.1e−05</td>
<td>7.6e−04</td>
<td>2.7e−02</td>
<td>[72]</td>
<td></td>
</tr>
<tr>
<td>miR-128a</td>
<td>6.6e−19</td>
<td>2.2e−48</td>
<td>4.7e−33</td>
<td>8.2e−21</td>
<td>[72]</td>
<td></td>
</tr>
<tr>
<td>miR-132</td>
<td>2.4e−08</td>
<td>3.7e−27</td>
<td>3.9e−33</td>
<td>1.2e−07</td>
<td>[72]</td>
<td></td>
</tr>
<tr>
<td>miR-133a</td>
<td>5.1e−04</td>
<td>9.4e−09</td>
<td>5.7e−06</td>
<td>4.3e−03</td>
<td>[72]</td>
<td></td>
</tr>
<tr>
<td>miR-142</td>
<td>2.4e−05</td>
<td>1.4e−13</td>
<td>1.0e−11</td>
<td>6.0e−05</td>
<td>[72]</td>
<td></td>
</tr>
<tr>
<td>miR-148b</td>
<td>6.3e−09</td>
<td>1.9e−11</td>
<td>3.2e−12</td>
<td>3.4e−04</td>
<td>[72]</td>
<td></td>
</tr>
<tr>
<td>miR-181a</td>
<td>7.9e−17</td>
<td>3.9e−53</td>
<td>2.4e−46</td>
<td>5.1e−18</td>
<td>[72]</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.1: **Rank correlation p-values for URM (U) and seed target (S) motifs.** p-values for RE motifs involving URM1 (U) and microRNA seed site targets (S) in different combinations for microRNA over-expression data sets. All p-values were calculated with the BB method. N denotes any nucleotide.

dom sequences. First, we inserted four different simple 7-mers up to 100 times at random positions in the upper half of the ranked sequences. We looked at the behavior of Regmex when defining motifs as REs capturing different subsets of the 7-mers including from one up to all four (i.e. REs defined to capture presence of any member of the subset). We clearly see the effect of combining multiple simple motifs in a set (Fig. 8.4 B). When searching for motif 1 or 2 (RE = m1|m2), we see a marked increase in detection sensitivity starting at around 20 inserted motifs. As expected, this increases with number of inserted motifs. Rank correlation increases even more dramatically for the motif subsets of three or four 7-mers. We note that the SSPs become less significant when including more 7-mers in the motif, but because the number of inserted motif observations in the enriched end of the sequence list increases (up to 400 for four 7-mers vs. 100 for a single 7-mer), the RCP becomes more significant.

We next looked at the behavior of Regmex when calculating rank correlation of multiple motifs present in the same sequences. Such calculations may be relevant when two or more different factors acting on the same sequences could explain the sequence ranking. To this end, we inserted the four 7-mers together in the same sequences. This was done up to 100 times in the upper half of the sequence list. We used Regmex to calculate RCPs for subsets of combined motifs, i.e. RE motifs designed to capture the presence of one up to all four 7-mers in the same sequence. The SSPs, in contrast to before, now decrease with the number of 7-mers in the RE subset, whereas number of motif observations are identical for all four motifs (by construction). As expected, the detection power of the combined motifs is much higher than that of a single simple motif (Fig. 8.4 C).

These experiments show how more complex motifs, such as motif sets, can be captured by REs with great increase in power.

**U rich motifs and microRNA seed target sites as combined motifs**

As an example of a scenario where combinations of motifs are relevant, we looked for rank correlation of microRNA seed sites targets in combination with a U-rich motif (URM) in a number of microRNA over-expression data sets. URMs are known to bind HuD/ELAVL4 [20] and their presence in 3’UTRs has been shown to correlate with down regulation in several microRNA over-expression experiments [95]. Based on this finding, a model was proposed where URMs augment microRNA induced destabilization of target mRNAs [95]. We used Regmex to calculate RCPs for microRNA seed site targets and combinations of the target and URM1 as identified in [95] with sequence UUUUAAA. This was done using 11 different microRNA over-expression data sets [72, 124].

We first calculated RCPs for the microRNA seed site targets. For all data sets, we saw low RCPs for the relevant microRNA seed site target in 3’UTRs, demonstrating correlation between the motif and down-regulated genes (Table 8.1).
We next calculated RCPs for microRNA seed site targets and URMI in combination. To this end, we constructed REs of the form (UN*S)|(SN*U), where U denotes the URM, S denotes the microRNA seed site target and N denotes any nucleotide. This RE will capture all combinations of the URM and the seed site in either orientation. As expected, based on the previous findings \[95\], we consistently saw an even lower RCP for the RE motif capturing both the seed target and the URMI motif (Table 8.1). The experiment thus verifies earlier results showing URMI 3'UTR presence correlating with down-regulation.

We next asked whether RCPs are of similar magnitude when the URM is downstream or upstream of the seed target. Here we used Regmex with two REs, SN*U and UN*S for the downstream and upstream question respectively. We observed low RCPs for both the downstream and upstream case for all microRNAs, indicating that URMI correlates with down-regulation regardless of its relative position to the seed target (Table 8.1). Notably, we found that RCPs were lower for URMI upstream the seed target than for URMI downstream the seed target. This could indicate a preference in relative position between the URM site and the augmented microRNA seed site targets.

The example above illustrates how Regmex is useful in testing well defined hypotheses involving combinations of motifs defined as REs. We note that the outcome verifies earlier findings and further suggests a positional bias in the relative position between the URM site and the seed target.

The 8.4 Discussion

We have introduced Regmex, an R package for evaluating rank correlation or clustering of motifs in lists of sequences. Regmex differs from current methods in combining powerful RE motif definition with accurate sequence specific motif significance evaluation and a variety of correlation score statistics. This makes Regmex a flexible tool that expands on the type of motif correlation problems that current methods can handle.

Although Regmex handles e.g. traditional exhaustive k-mer screens as other methods \[170, 203\], it is designed for specific hypothesis testing that involves potentially complex RE motif correlation testing. In particular, Regmex can accurately evaluate rank correlation significance for arbitrary combined sets of simple motifs, such as sets of high scoring k-mers in conventional k-mer screens or combinations of microRNA seed sites. This is relevant for investigations of competitive endogenous RNAs, snoRNA target sites, etc.

The accuracy introduced by the embedded model comes at the cost of computational speed. In particular, motifs that occur frequently in the sequences and are also represented by DFAs with many states require large eTPMs, which may slow down the computation.

This can be countered by parallelization, which Regmex natively supports. Furthermore, data structures are simple enough to sub-divide and distribute calculations. Also, the embedded model may be reduced by introducing counting transitions, as suggested in \[173\].

Regmex offers three alternative ways of evaluating motif rank correlation, which differ in their null models. For the RW method, the null model is that motifs occur at random given the sequence compositions and lengths. The RW method is sensitive to stretches of low SSPs anywhere in the sequence list, and thus may find use in special cases where enrichment is expected off the ends. This could be the case if a sequence list represents consecutive functional sets of sequences, such as a gene ontologies or expression clusters.

Both the MSR and the BB methods are more sensitive to motifs occurring in the ends of the list, but have subtle differences in their null models. For the MSR method, the number of observed motifs in the sequence list is fixed, and only their distribution among the sequences vary under the null. For the BB method, the null is a uniform distribution of SSPs. Although this would suggest a bias for motifs occurring more frequently than expected, the transformation of SSPs into a Brownian bridge via a running sum normalizes for this effect. Thus both of these methods should be robust to motif occurrence bias. As noted (Fig. 8.3 and Fig. 8.5) they have different sensitivity in different scenarios. The MSR method tends to be more sensitive than the BB method for longer sequence lists and vice versa.

Regmex is implemented in R and offers a number of customization options including model options such as di-nucleotide dependence and a motif overlap option. Alternative outputs such as SSPs and \[n_{obs}\] combined with simple data formats makes Regmex well suited for a range of problems.
8.5 Supplementary information

Modified rank sum statistic
Let \( s_1, s_2, \ldots, s_N \) be a list of sequences ranked according to an experimental setting, and let \( n_i \) denote the number of observed motifs in \( s_i \). Under the null model, we assume \( n_i \sim \text{po}i(\lambda_i) \), with 
\[
\lambda_i = -\ln(1-p),
\]
where \( p \) is the probability of observing at least one motif in the sequence. This follows from the probability mass function of the Poisson distribution
\[
Pr(X = k) = \frac{\lambda^k}{k!} e^{-\lambda},
\]
since \( p = 1 - Pr(X = 0) = 1 - e^{-\lambda} \) we have 
\[
\lambda_i = -\ln(1-p). \]

If we think of motif occurrences as a Poisson process, where our "time axis" is composed of consecutive intervals of length \( \lambda_i \) ordered according to the experimental rank, motif occurrences are now, under the null hypothesis, uniformly distributed on the interval \([0, \lambda_i]\), where \( \lambda = \sum_{i=1}^{N} \lambda_i \).

We now calculate a score \( r_m \), corresponding to the mid point of the interval (sequence) in which a motif was observed,
\[
r_m = \frac{\sum_{i=1}^{m-1} \lambda_i + \sum_{i=1}^{m} \lambda_i}{2}.
\]

We associate the score with motif occurrences in the sequence list. Under the null hypothesis, the probability of observing a motif in a sequence is proportional to the interval length, and thus the expectation is that motif scores are uniformly distributed across the whole interval \([0, \lambda]\). Under the null model, the score for motif occurrences is thus normally distributed with mean \( \lambda^2 / 2 \) and variance \( \lambda^2 / 12 \).

We calculate the test statistic
\[
W = \sqrt{n} \left( \frac{\sum_{i=1}^{N} n_i r_i}{n} - \frac{\lambda}{2} \right) \sim \mathcal{N} \left( 0, \frac{1}{12} \right),
\]
where \( n = \sum_{i=1}^{N} n_i \). The motif correlation \( p \)-value is \( p = 2[1 - \Phi(|W|)] \).

Brownian bridge method
This method is a re-implementation of the method developed by Jacobsen et al. [95] and recently implemented in cWords [170]. Our implementation differs in the calculation of the sequence dependent motif \( p \)-values. The method calculates the max value \( D \) of a running sum of mean adjusted log scores of the ranked sequence dependent \( p \)-values \( p_i \) (SSPs) \( r_i = r_{i-1} + \ln(\bar{s}_i - \bar{s}) \), where \( \bar{s}_i = -\ln(p_i + \alpha) \) and \( \alpha \) is a score dampening factor of \( 10^{-5} \). \( \bar{s} \) is the mean of the log scores.

The running sum has the form of a bridge (starting and ending in 0), and the maximum value is compared to the theoretical distribution of the absolute maximum \( M \) of a Brownian bridge under the null model [17]
\[
Pr(M \geq m) = 1 - 2 \sum_{k=1}^{\infty} (-1)^k e^{-2k^2m^2/n},
\]
where \( n \) is the number of sequences in the sequence list.

Random walk statistics
The random walk (RW) method is inspired by the way RW theory is used in the BLAST algorithm to estimate significance of observed homologies between sequences [4]. In that case matches and mismatches become steps in a walk. Here, it is the list of sequence specific \( p \)-values (SSPs) for a motif that are transformed into steps in a walk. The maximum value of the walk is compared to the probability distribution of maxima under the null model.

Under the null model, sequence specific \( p \)-values (SSPs) are assumed to be uniformly distributed between 0 and 1. This situation is expected for (indefinitely) long random sequences. This condition, however, does not hold in general. For shorter sequences and longer motifs, the SSPs tend to discretize to small intervals specific for the number of observed motifs (see Fig. 8.6). In particular, the \( p \)-value for observing '0 or more' motif occurrences is always 1. However, for random sequences, the cumulative distribution of discrete SSPs approximates that of the uniform distribution in the SSP values [150] (see Fig. 8.6). Because the discrete SSPs depend highly on both the sequence and the motif, we normalize for the discrete effect. This is done by drawing \( p \)-values from the interval between the probability of observing \( n_{obs} \) or more and \( n_{obs} + 1 \) or more motifs. E.g. for a 6-mer not observed in a 1000 bases random sequence, we draw between \( p(n_{obs} \geq 0) = 1 \) and \( p(n_{obs} \geq 1) = 0.2 \). In this way the modified SSPs for random motifs in random sequences will follow the uniform distribution as required under the null model.

The modified \( p \)-values are now transformed into scores in a walk according to a scoring scheme

120
Figure 8.6: **Empirical distributions for sequence $p$-values.** (A) The plot shows the discrete behaviour of SSPs for a 3-mer motif (ATG) in 100,000 random sequences of length 100 nucleotides. All sequences were generated with equal nucleotide frequencies of 0.25. SSPs corresponding to different $n_{\text{obs}}$ are indicated. The distribution function for the SSPs define a step function that becomes nearly equal to the uniform distribution at the SSP values. (B) The plot shows the discrete behaviour of SSPs for random 2-7-mer motifs. The distributions were made for 1000 random sequences of 1000 bases. The sequences, unlike in (A), were generated by drawing with equal nucleotide probability, and thus do not have equal nucleotide frequencies. This is reflected in the smeared SSPs for a given motif. For longer motifs SSPs are more discrete, e.g. $p(n_{\text{obs}} \geq 0) = 1$ and $p(n_{\text{obs}} \geq 1) \sim 0.2$ for a 6-mer. Motifs used for SSP evaluation were drawn randomly from all motifs with corresponding length. A random uniform distribution $[0;1]$ is shown for comparison.

We are interested in the distribution of maximum values that a random walk visits before absorption, under the null model. The random walk is a particular case of a discrete-time Markov chain. We calculate the distribution of interest by building a Markov model with states $-1, 0, 1, \ldots, M$, where $M$ is the observed maximum height of the walk, and assign transition probabilities according to the scoring scheme. From the resulting probability distribution of maxima under the null model, we derive the $p$-value for the observed maximum height. Alternatively, we can calculate the $p$-value by using a geometric-like (Gumbel) distribution approximation for the random walk maxima as given in [59].
Algorithms for hidden Markov models restricted to occurrences of regular expressions

Paula Tataru¹ Andreas Sand¹,² Asger Hobolth¹
Thomas Mailund¹ Christian N. S. Pedersen¹,²

The paper presented in this chapter was published in 2013 in a special issue on Developments in Bioinformatics Algorithms of Biology.


Apart from minor typographical and formatting changes, the content of this chapter is identical to the journal paper.

Abstract

Hidden Markov Models (HMMs) are widely used probabilistic models, particularly for annotating sequential data with an underlying hidden structure. Patterns in the annotation are often more relevant to study than the hidden structure itself. A typical HMM analysis consists of annotating the observed data using a decoding algorithm and analyzing the annotation to study patterns of interest. For example, given an HMM modeling genes in DNA sequences, the focus is on occurrences of genes in the annotation. In this paper, we define a pattern through a regular expression and present a restriction of three classical algorithms to take the number of occurrences of the pattern in the hidden sequence into account. We present a new algorithm to compute the distribution of the number of pattern occurrences, and we extend the two most widely used existing decoding algorithms to employ information from this distribution. We show experimentally that the expectation of the distribution of the number of pattern occurrences gives a highly accurate estimate, while the typical procedure can be biased in the sense that the identified number of pattern occurrences does not correspond to the true number. We furthermore show that using this distribution in the decoding algorithms improves the predictive power of the model.

¹Bioinformatics Research Center, Aarhus University, Aarhus, Denmark
²Department of Computer Science, Aarhus University, Aarhus, Denmark
9. Restricted algorithms for HMMs

9.1 Introduction

A Hidden Markov Model (HMM) is a probabilistic model for sequential data with an underlying hidden structure. Because of their computational and analytical tractability, they are widely used especially in speech recognition [36, 64, 167], image processing [123] and in several applications in bioinformatics; e.g., modeling of proteins [100, 113, 115], sequence alignment [52, 53, 128], phylogenetic analysis [130, 183] and identification of coding regions in genomes [9, 127].

Patterns in the hidden structure are, however, often more relevant to study than the full hidden structure itself. When modeling proteins, one might be interested in neighboring secondary structures that differ, while for sequence alignments, the pattern could capture specific characteristics, such as long indels. In phylogenetic analysis, changes in the tree along the sequence are most relevant, while when investigating coding regions of DNA data, patterns corresponding to genes are the main focus.

Counting the number of occurrences of such patterns can be approached (as in the methods based on [114]) by making inferences from the prediction of a decoding algorithm; e.g., the Viterbi algorithm or the posterior-Viterbi algorithm. As we show in our experiments, this can give consistently biased estimates for the number of pattern occurrences. A more realistic method is presented in [10], where the distribution of the number of pattern occurrences is computed by means of Markov chain embedding. To our knowledge, this is the only study of patterns in the hidden sequence of HMMs. The problem of pattern finding in random sequences generated by simple models, such as Markov chains, has been intensively studied using the embedding technique [63, 155, 213].

We present a fundamentally different approach to compute the distribution of the number of pattern occurrences and show how it can be used to improve the prediction of the hidden structure. We use regular expressions as patterns and employ their deterministic finite automata to keep track of occurrences. The use of automata to describe occurrences of patterns in Markov sequences has been described previously in [126, 153, 155]. However, analyzing pattern occurrences in the hidden structure of HMMs by means of automata has not been done before. We introduce a new version of the forward algorithm, the restricted forward algorithm, which computes the likelihood of the data under the hidden sequence containing a specific number of pattern occurrences. This algorithm can be used to compute the occurrence number distribution. We furthermore introduce new versions of the two most widely used decoding algorithms, the Viterbi algorithm and the posterior-Viterbi algorithm, where the prediction is restricted to containing a certain number of occurrences, e.g., the expectation obtained from the distribution. We have implemented and tested the new algorithms and performed experiments that show that this approach improves both the estimate of the number of pattern occurrences and the prediction of the hidden structure.

The remainder of this paper is organized as follows: we start by introducing Hidden Markov Models and automata; we continue by presenting our restricted algorithms, which we then validate experimentally.

Hidden Markov models

A Hidden Markov Model (HMM) [167] is a joint probability distribution over an observed sequence \( y_{1:T} = y_1 y_2 \ldots y_T \in \mathcal{O}^* \) and a hidden sequence \( x_{1:T} = x_1 x_2 \ldots x_T \in \mathcal{H}^* \), where \( \mathcal{O} \) and \( \mathcal{H} \) are finite alphabets of observables and hidden states, respectively. The hidden sequence is a realization of a Markov process that explains the hidden properties of the observed data. We can formally define an HMM as consisting of a finite alphabet of hidden states \( \mathcal{H} = \{h_1, h_2, \ldots, h_N\} \); a finite alphabet of observables \( \mathcal{O} = \{o_1, o_2, \ldots, o_M\} \); a vector

\[
\Pi = (\pi_{h_i})_{1 \leq i \leq N} , \text{ where } \pi_{h_i} = \mathbb{P}(X_1 = h_i)
\]

is the probability of the hidden sequence starting in state \( h_i \); a matrix

\[
A = \{a_{h_i, h_j}\}_{1 \leq i, j \leq N} , \text{ where } a_{h_i, h_j} = \mathbb{P}(X_t = h_j \mid X_{t-1} = h_i)
\]

is the probability of a transition from state \( h_i \) to state \( h_j \); and a matrix

\[
B = \{b_{h_i, o_j}\}_{1 \leq j \leq M} , \text{ where } b_{h_i, o_j} = \mathbb{P}(Y_t = o_j \mid X_t = h_i)
\]

is the probability of state \( h_i \) emitting \( o_j \).

Figure 9.1 shows an HMM designed for gene finding. The observed sequence is a DNA sequence over the alphabet \( \mathcal{O} = \{A, C, G, T\} \), while the hidden states encode if a position is in a non-coding
9.1. Introduction

area \((N)\) or is part of a gene on the positive strand \((C)\) or on the negative strand \((R)\). The model incorporates the fact that nucleotides come in multiples of three within genes, where each nucleotide triplet codes for an amino acid. The set of hidden states is \(H = \{N, C_i, R_i | 1 \leq i \leq 3\}\). In practice, models used for gene finding are much more complex, but this model captures the essential aspects of a gene finder.

HMMs can be used to generate sequences of observables, but their main application is for analyzing an observed sequence, \(y_{1:T}\). The likelihood of a given observed sequence can be computed using the forward algorithm [167], while the Viterbi algorithm [167] and the posterior-Viterbi algorithm [60] are used for predicting a corresponding hidden sequence. All these algorithms run in \(O(TN^2)\) time, using \(O(TN)\) space.

The forward algorithm

The forward algorithm [167] finds the probability of observing \(y_{1:T}\) by summing the joint probability of the observed and hidden sequences for all possible sequences, \(x_{1:T}\). This is given by

\[
\mathbb{P}(y_{1:T}, x_{1:T}) = \pi_{x_1} b_{x_1, y_1} \prod_{t=2}^{T} a_{x_{t-1}, x_t} b_{x_t, y_t},
\]

\[
\mathbb{P}(y_{1:T}) = \sum_{x_{1:T}} \mathbb{P}(y_{1:T}, x_{1:T}),
\]

where Eq. (9.1) is the multiplication of the probabilities of transitions and emissions, which explain observing \(y_{1:T}\) with \(x_{1:T}\) as the hidden sequence: the HMM starts in state \(x_1\) and emits \(y_1\) from \(x_1\), and for all \(t = 2, \ldots, T\), it makes a transition from state \(x_{t-1}\) to \(x_t\) and emits \(y_t\) from \(x_t\).

The forward algorithm finds Eq. (9.2) by recursively filling up a table, \(\alpha\), with values

\[
\alpha_t(x_t) = \mathbb{P}(y_{1:t}, x_t) = \sum_{x_{t-1}} \mathbb{P}(y_{1:t} \mid x_{1:t-1}) \mathbb{P}(x_t | x_{t-1}),
\]

being the probability of observing \(y_{1:t}\) and being in state \(x_t\) at time \(t\). The recursion is

\[
\alpha_t(h_t) = \pi_t, b_{h_t, y_t},
\]

\[
\alpha_t(h_t) = b_{h_t, y_t} \sum_{j} \alpha_{t-1}(h_j) a_{h_j, h_t},
\]

(9.3)

and, finally, \(\mathbb{P}(y_{1:T}) = \sum_{x_{1:T}} \alpha_T(h_t)\).

The Viterbi algorithm

The Viterbi algorithm [167] finds the sequence of hidden states, \(x_{1:T}\), that maximizes the joint probability of the observed and hidden sequences Eq. (9.1). It uses the same type of approach as the forward algorithm: a new table, \(\omega\), is defined by

\[
\omega_t(x_t) = \max_{x_{t-1}} \{\mathbb{P}(y_{1:t}, x_{1:t})\},
\]

the probability of a most likely decoding ending in \(x_t\) at time \(t\), having observed \(y_{1:t}\). This can be obtained as follows

\[
\omega_t(h_t) = \pi_t, b_{h_t, y_t},
\]

\[
\omega_t(h_t) = b_{h_t, y_t} \max_j \{\omega_{t-1}(h_j) a_{h_j, h_t}\}.
\]

After computing \(\omega\), a most likely sequence of hidden states is retrieved by backtracking through the table, starting in entry \(\arg\max_h \{\omega_T(h_t)\}\).

The posterior decoding algorithm

The posterior decoding [167] of an observed sequence is an alternative to the prediction given by the Viterbi algorithm. While the Viterbi algorithm computes the decoding with the highest joint probability, the posterior decoding computes a sequence of hidden states, \(x_{1:T}\), such that
Restricted algorithms for HMMs

Figure 9.2: Two automata, $FA_H(r)$ and $EA_H(r)$, for the pattern $r = (NC_1) \mid (R_1N)$, $H = \{N\} \cup \{C_i, R_i \mid 1 \leq i \leq 3\}$, $Q = \{0, 1, 2, 3, 4\}$, $q_0 = 0$, and $A = \{2, 4\}$. States $1, 2$ and $3, 4$ are used for matching sequences ending with $NC_1$ and $R_1N$, respectively, as marked with gray boxes. The two automata differ only with respect to transitions from accepting states: the dotted transition belongs to $FA_H(r)$ and the dashed one to $EA_H(r)$. For clarity, the figure lacks transitions going from all states to state $0$ using $C_2, C_3, R_2$ and $R_3$.

$x_t = \arg\max_{h_t} \{\gamma_t(h_t)\}$ has the highest posterior probability

$$\gamma_t(h_t) = P(h_t \mid y_{1:T}).$$

If we let

$$\beta_t(h_t) = P(y_{t+1:T} \mid h_t)$$

we have

$$\gamma_t(h_t) = \frac{P(h_t, y_{1:t}) P(y_{t+1:T} \mid h_t)}{P(y_{1:T})} = \frac{\alpha_t(h_t) \beta_t(h_t)}{\sum_j \alpha_t(h_j)}.$$

The posterior decoding algorithm often computes decodings that are very accurate locally, but it may return syntactically incorrect decodings; i.e., decodings with transitions that have a probability of zero. The posterior-Viterbi algorithm [60] corrects for this by computing a syntactically correct decoding $x_{1:T}^* = \arg\max_{x_{1:T} \in A_p} \{\prod_{t=1}^T \gamma_t(x_t)\}$ with the highest posterior probability, where $A_p$ is the set of syntactically correct decodings. To compute this, a new table, $\tilde{\gamma}$, is defined by

$$\tilde{\gamma}_t(x_t) = \max_{x_{t+1} \in A_p} \{\gamma_t(x_t)\},$$

the maximum posterior probability of a decoding from $A_p$ ending in $x_t$ at time $t$. The table is filled using the recursion

$$\tilde{\gamma}_t(h_t) = \gamma_t(h_t),$$

$$\tilde{\gamma}_t(h_t) = \gamma_t(h_t) \cdot \max_{\{j : a_{h_t, h_j} > 0\}} \{\tilde{\gamma}_{t-1}(h_j)\}.$$

After computing $\tilde{\gamma}$, a decoding from $A_p$ with the highest posterior probability is retrieved by backtracking through $\tilde{\gamma}$ from entry $\arg\max_{h_t} \{\tilde{\gamma}_T(h_t)\}$. We note that, provided that the posterior decoding algorithm returns a decoding from $A_p$, the posterior-Viterbi algorithm will return the same decoding.

The posterior-Viterbi algorithm

The posterior decoding algorithm often computes decodings that are very accurate locally, but it may return syntactically incorrect decodings; i.e., decodings with transitions that have a probability of zero. The posterior-Viterbi algorithm [60] corrects for this by computing a syntactically correct decoding $x_{1:T}^* = \arg\max_{x_{1:T} \in A_p} \{\prod_{t=1}^T \gamma_t(x_t)\}$ with the highest posterior probability, where $A_p$ is the set of syntactically correct decodings. To compute this, a new table, $\tilde{\gamma}$, is defined by

$$\tilde{\gamma}_t(x_t) = \max_{x_{t+1} \in A_p} \{\gamma_t(x_t)\},$$

the maximum posterior probability of a decoding from $A_p$ ending in $x_t$ at time $t$. The table is filled using the recursion

$$\tilde{\gamma}_t(h_t) = \gamma_t(h_t),$$

$$\tilde{\gamma}_t(h_t) = \gamma_t(h_t) \cdot \max_{\{j : a_{h_t, h_j} > 0\}} \{\tilde{\gamma}_{t-1}(h_j)\}.$$

After computing $\tilde{\gamma}$, a decoding from $A_p$ with the highest posterior probability is retrieved by backtracking through $\tilde{\gamma}$ from entry $\arg\max_{h_t} \{\tilde{\gamma}_T(h_t)\}$. We note that, provided that the posterior decoding algorithm returns a decoding from $A_p$, the posterior-Viterbi algorithm will return the same decoding.

Automata

In this paper, we are interested in patterns over the hidden alphabet $H = \{h_1, h_2, \ldots, h_N\}$ of an HMM. Let $r$ be a regular expression over $H$, and let $FA_H(r) = (Q, H, q_0, \delta)$ [201] be the deterministic finite automaton (DFA) that recognizes the language described by $(h_1 \mid h_2 \mid \ldots \mid h_N)^*(r)$, where $Q$ is the finite set of states, $q_0 \in Q$ is the initial state, $A \subseteq Q$ is the set of accepting states and $\delta : Q \times H \rightarrow Q$ is the transition function. $FA_H(r)$ accepts any string that has $r$ as a suffix.
We construct the DFA $EA_H(r) = (Q, \mathcal{H}, q_0, A, \delta_E)$ as an extension of $FA_H(r)$, where $\delta_E$ is defined by

\[
\forall q \in Q \setminus A, \quad \forall h \in \mathcal{H}, \quad \delta_E(q, h) = \delta(q, h),
\]

\[
\forall q \in A, \quad \forall h \in \mathcal{H}, \quad \delta_E(q, h) = \delta(q_0, h).
\]

Essentially, $EA_H(r)$ restarts every time it reaches an accepting state. Figure 9.2 shows $FA_H(r)$ and $EA_H(r)$ for $r = (NC_1) \mid (R_1N)$ with the hidden alphabet $\mathcal{H} = \{N\} \cup \{C_i, R_i \mid 1 \leq i \leq 3\}$ of the HMM from Fig. 9.2. Both automata have $Q = \{0, 1, 2, 3, 4\}$, $q_0 = 0$, and $A = \{2, 4\}$. State 1 marks the beginning of $NC_1$, while state 3 corresponds to the beginning of $R_1N$. State 2 accepts $NC_1$, and state 4 accepts $R_1N$. As $C_2, C_3, R_2$ and $R_3$ are not part of $r$, using them, the automaton restarts by transitioning to state 0 from all states. We left these transitions out of the figure for clarity. The main difference between $FA_H(r)$ and $EA_H(r)$ is that they correspond to overlapping and non-overlapping occurrences, respectively. For example, for the input string, $R_1NC_1$, $FA_H(r)$ first finds $R_1N$ using state 4, from which it transitions to state 2 and matches $NC_1$. However, after $EA_H(r)$ recognizes $R_1N$, it transitions back to state 0, not matching $NC_1$. The algorithms we provide are independent of which of the two automata is used, and therefore, all of the automaton is to switch between them when needed. In our implementation, we used an automata library for Java [148] to obtain $FA_H(r)$, which we then converted to $EA_H(r)$.

### 9.2 Results and discussion

Consider an HMM as defined previously, and let $r$ be a regular expression over the alphabet of hidden states, $\mathcal{H}$. We present a modified version of the forward algorithm to compute the distribution of the number of occurrences of $r$, which we then use in an adaptation of the Viterbi algorithm and the posterior-Viterbi algorithm to obtain an improved prediction.

#### The restricted forward algorithm

Let $O_r(x_{1:T})$ be the number of matches of $r$ in $x_{1:T}$. We wish to estimate $O_r$ by using its probability distribution. We do this by running the HMM and $FA_H(r)$ in parallel. Let $FA_H(r)_t$ be the state in which the automaton is after $t$ transitions, and define

\[
\hat{\alpha}_t(x_t, k, q) = \sum_{\{x_{1:t-1} : O_r(x_{1:t}) = k \}} \mathbb{P}(y_{1:t}, x_{1:t}, FA_H(r)_t = q),
\]

(9.4)

\[
\hat{\alpha}_t(x_t, k, q) = \sum_{\{x_{1:t-1} : O_r(x_{1:t}) = k \}} \mathbb{P}(y_{1:t}, x_{1:t}, FA_H(r)_t = q) = \sum_{q' \in \delta^{-1}(q, x_t)} \mathbb{P}(y_{1:t-1}, x_{1:t-1}, FA_H(r)_{t-1} = q') \cdot \sum_{q' \in \delta^{-1}(q, x_t)} \mathbb{P}(y_{1:t-1}, x_{1:t-1}, FA_H(r)_{t-1} = q').
\]

(9.5)

Using Eqs. (9.4) and (9.5), the recursion for $\hat{\alpha}$ becomes

\[
\hat{\alpha}_1(h_i, k, q) = \pi_{h_i} b_{h_i, y_t} \mathbb{I}(q_0 \in \delta^{-1}(q, h_i)) \cdot \begin{cases} 1 & \text{if } k = 0, \\ \mathbb{I}(q \in A) & \text{if } k = 1, \\ 0 & \text{otherwise} \end{cases}
\]

\[
\hat{\alpha}_t(h_i, k, q) = b_{h_i, y_t} \sum_j a_{h_i, j} \hat{\alpha}_{t-1}(h_j, k - \mathbb{I}(q \in A), q') \cdot \sum_{q' \in \delta^{-1}(q, h_i)} \hat{\alpha}_{t-1}(h_j, k - \mathbb{I}(q \in A), q').
\]

These probabilities now allow for the evaluation of the distribution of the number of occur-
stances, conditioned on the observed data
\[
P(k \text{ occurrences of } r \mid y_{1:T}) = \frac{\mathbb{P}(k \text{ occurrences of } r, y_{1:T})}{\mathbb{P}(y_{1:T})} = \frac{1}{\mathbb{P}(y_{1:T})} \sum_{i,q} \alpha_T(h_i, k, q),
\]
from which the expectation can be computed. The likelihood of the data can be obtained from either the forward or restricted forward algorithm,
\[
\mathbb{P}(y_{1:T}) = \sum_i \alpha(h_i) = \sum_{i,k,q} \alpha_T(h_i, k, q).
\]
The \( \hat{\alpha} \) table has \( T \cdot N \cdot m \cdot |Q| \) entries, and the computation of each requires \( \mathcal{O}(N|Q|) \), leading to a \( \mathcal{O}(TN^2m|Q|^2) \) running time and a space consumption of \( \mathcal{O}(TNm|Q|) \). In practice, both time and space consumption can be reduced. The restricted forward algorithm can be run for values of \( k \) that are gradually increasing up to \( k_{\text{max}} \) for which \( \mathbb{P}(\text{at most } k_{\text{max}} \text{ occurrences of } r \mid y_{1:T}) \) is greater than, e.g., 99.99%. This \( k_{\text{max}} \) is generally significantly less than \( m \), while the expectation of the number of matches of \( r \) can be reliably calculated from this truncated distribution. The space consumption can be reduced to \( \mathcal{O}(N|Q|) \), because the calculation at time \( t \) for a specific value, \( k \), depends only on the results at time \( t-1 \) for \( k \) and \( k-1 \).

**Restricted decoding algorithms**

The aim of the restricted decoding algorithms is to obtain a sequence of hidden states, \( x_{1:T} \), for which \( O_{\cdot r}(x_{1:T}) \in [l, u] \), where \( l \) and \( u \) are set to, for example, the expected number of occurrences, which can be calculated from the distribution. The restricted decoding algorithms are built in the same way as the restricted forward was obtained: a new table is defined, which is filled using a simple recursion. The evaluation of the table is followed by backtracking to obtain a sequence of hidden states, which contains between \( l \) and \( u \) occurrences of the pattern. The two restricted decoding algorithms use \( \mathcal{O}(TNu|Q|) \) space and \( \mathcal{O}(TN^2u|Q|^2) \) time.

The entries in the table for the restricted Viterbi algorithm contain the probability of a most likely decoding containing \( k \) pattern occurrences, ending in state \( x_t \) and automaton state \( q \) at time \( t \) and having observed \( y_{1:t} \).

\[
\hat{\omega}_t(x_t, k, q) = \max \left\{ \mathbb{P}(y_{1:t}, x_{1:t}, FA_r(r)_t = q) \middle| x_{1:t-1}: O_{\cdot r}(x_{1:t})=k \right\},
\]
with \( k = 0, \ldots, u \). The corresponding recursion is
\[
\hat{\omega}_t(h_i, k, q) = \pi_{h_i}b_{h_i,y_i}\mathbb{1}(q_0 \in \delta^{-1}(q, h_i))
\cdot \begin{cases} 1(q \not\in A) & \text{if } k = 0, \\ 1(q \in A) & \text{if } k = 1, \\ 0 & \text{otherwise} \end{cases},
\]
\[
\hat{\omega}_t(h_i, k, q) = b_{h_i,y_i} \cdot \max_j \left\{ \alpha_{h_i, h_j} \cdot \max_{q' \in \delta^{-1}(q, h_i)} \{\hat{\omega}_{t-1}(h_j, k - 1(q \in A), q')\} \right\}.
\]
The entry \( \arg \max_{i,k \in [u, q]} \{\hat{\omega}_T(h_i, k, q)\} \) is the starting point for the backtracking.

For the restricted posterior-Viterbi algorithm, we compute the highest posterior probability of a decoding from \( A_p \) containing \( k \) pattern occurrences, ending in state \( x_t \) and automaton state \( q \) at time \( t \) and having observed \( y_{1:t} \).

\[
\hat{\gamma}_t(x_t, k, q) = \max \left\{ \mathbb{P}(x_{1:t}, FA_r(r)_t = q \mid y_{1:T}) \middle| x_{1:t-1}: O_{\cdot r}(x_{1:t})=k \right\}.
\]
We have
\[
\hat{\gamma}_1(h_i, k, q) = \gamma_1(h_i) \cdot \mathbb{1}(q_0 \in \delta^{-1}(q, h_i))
\cdot \begin{cases} 1(q \not\in A) & \text{if } k = 0, \\ 1(q \in A) & \text{if } k = 1, \\ 0 & \text{otherwise} \end{cases},
\]
\[
\hat{\gamma}_t(h_i, k, q) = \gamma_t(h_i) \cdot \max_{j, a_{i,h_i}} \left\{ \max_{q' \in \delta^{-1}(q, h_i)} \{\hat{\gamma}_{t-1}(h_j, k - 1(q \in A), q')\} \right\}.
\]
The entry \( \arg \max_{i,k \in [u, q]} \{\hat{\gamma}_T(h_i, k, q)\} \) is the starting point for the backtracking.

**Experimental results on simulated data**

We implemented the algorithms in Java, validated and evaluated their performance experimentally as follows: We first generated a test set consisting of 500 pairs of observed and hidden sequences for each length \( L = 500, 525, \ldots, 1,500 \) from the gene finder in Fig. 9.2. As the HMM is used for finding genes, an obvious choice of pattern is \( r = (NC_1) \mid (R_1N) \), corresponding to the start
9.2. Results and discussion

Figure 9.3: Normalized difference, $\frac{\text{estimate} - \text{true value}}{\text{true value}}$, between the true number of pattern occurrences, the number given by the two unrestricted decoding algorithms and the expected number of pattern occurrences computed using the restricted forward algorithm. For each sequence length, we show the median of the normalized differences in the 500 experiments, together with the 0.025 and 0.975 quantiles, given as error bars.

of a gene. For each of the sequences, we estimated the number of overlapping pattern occurrences with the expected number of pattern occurrences, computed using the restricted forward algorithm, which we then used to run the restricted decoding algorithms. We also computed the prediction given by the two unrestricted decoding algorithms for comparison.

Counting pattern occurrences

Figure 9.3 shows the power of the restricted forward algorithm and the two unrestricted decoding algorithms to recover the true number of pattern occurrences. For each given length, we computed the normalized difference, $\frac{\text{estimate} - \text{true value}}{\text{true value}}$, and plotted the median over the 500 sequences together with the 0.025 and 0.975 quantiles, given as error bars. As Fig. 9.3 shows, the expectation gives a very good estimate. The decoding algorithms’ performances are significantly lower, and they always give underestimates. This may be partly due to the model structure, where transitions to and from coding regions have low probability, leading to few, but long, genes.

Quality of predictions

We compared the predictive power of the two decoding algorithms in the original and restricted versions, using the expectation for the number of pattern occurrences. For each length in the test set, we measured the quality of each method, both at the nucleotide and gene level, following the analysis in [28]. Because the measures we use require binary data, we first converted the true hidden sequence and decoding to binary data containing non-coding areas and genes, by first considering the genes on the reverse strand as non-coding and calculating the measures for the genes on the direct strand, and vice versa. The final plotted measures are the averages obtained from the two conversions.

Nucleotide level. To investigate the quality at the nucleotide level, we compared the decoding and the true hidden state position by position. Each position can be classified as a true positive (predicted as part of a gene when it was part of a gene), true negative (predicted as non-coding when it was non-coding), false positive (predicted as part of a gene when it was non-coding) and false negative (predicted as non-coding when it was part of a gene). Using the total number of true positives ($tp$), true negatives ($tn$), false positives ($fp$) and false negatives ($fn$), we calculated the sensitivity, specificity and Matthew’s correlation coefficient (MCC)

$$sens = \frac{tp}{tp + fn}, \quad spec = \frac{tn}{tn + fp},$$

$$mcc = \frac{tp \cdot tn - fp \cdot fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}}.$$  

Sensitivity and specificity are always between zero and one and relate to how well the algorithms are able to find genes (true positives) and non-coding regions (true negatives), respectively. MCC reflects the overall correctness and lies between $-1$ and $1$, where $1$ represents perfect prediction.

Gene level. When looking at the decoding position by position, genes that are predicted cor-
9. Restricted algorithms for HMMs

Figure 9.4: Error types at the gene level. A predicted gene is considered one true positive if it overlaps with at least 50% of a true gene and one false positive if there is no true gene with which it overlaps by at least 50%. Each true gene for which there is no predicted gene that overlaps by at least 50% counts as one false negative. True genes that are covered more than 50% by predicted genes, but for which there is no single predicted gene that covers a minimum of 50% are disregarded.

Figure 9.5: Prediction quality at the nucleotide level. The plot shows the average sensitivity, specificity and Matthew’s correlation coefficient (MCC) for the decoding algorithms. We ran the restricted decoding algorithms using the expectation calculated from the distribution returned by the restricted forward algorithm. The plots show a zoom-in of the three measures. As both sensitivity and specificity are between zero and one, the Y-axes in these two plots have the same scale.

Figure 9.6: Prediction quality at the gene level. The plot shows the average recall, precision and F-score for the decoding algorithms. We ran the restricted decoding algorithms using the expectation calculated from the distribution returned by the restricted forward algorithm. The plots show a zoom-in of the three measures with the same scale on the Y axes.
rectly do not contribute to the measures in an equal manner, but rather, the longer the gene, the more contribution it brings. However, it is interesting how well the genes are recovered, independent of how long they are. To measure this, we consider a predicted gene as one true positive if it overlaps by at least 50% of a true gene and as one false positive if there is no true gene with which it overlaps by at least 50%. Each true gene for which there is no predicted gene that overlaps by at least 50% counts as one false negative; see Fig. 9.4. In this context, true negatives, i.e., the areas where there was no gene and no gene was predicted, are not considered, as they are not informative. The final measures are the recall, precision and F-score

\[
\text{recall} = \frac{tp}{tp + fn}, \quad \text{precision} = \frac{tp}{tp + fp}, \quad \text{F-score} = 2 \cdot \frac{\text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}} = \frac{2 \cdot tp}{2 \cdot tp + fn + fp}.
\]

They are all between zero and one and reflect how well the true genes have been recovered. The recall gives the fraction of true genes that have been found, while the precision gives the fraction of the predicted genes that are true genes. The F-score is the harmonic mean of the two.

Figures 9.5 and 9.6 show the quality of predictions at the nucleotide and gene level, respectively. When comparing the Viterbi algorithm with the restricted Viterbi algorithm, it is clear that the restriction brings a great improvement to the prediction, as the restricted version has an increased power in all measures considered, with the exception of precision, where the Viterbi algorithm shows a tendency of increased precision with sequence length. However, when comparing the posterior-Viterbi algorithm with its restricted version, it is apparent that the restricted version does as good at the nucleotide level, but it performs worse at the gene level. By inspecting the predictions of the two methods, it was clear that the restricted posterior-Viterbi obtained an increased number of genes dictated by the expectation by just fractioning the genes predicted by the posterior-Viterbi. We believe this happens because the posterior-Viterbi algorithm finds the best local decoding, and therefore, adding global information, such as the total number of genes in the prediction, does not aid in the decoding. On the other hand, as the Viterbi algorithm finds the best global decoding, using the extra information results in a significant improvement of the prediction. Overall, at the nucleotide level, the posterior-Viterbi shows the best performance, while at the gene level, the restricted Viterbi has the highest quality. One might expect this, given the nature of the algorithms.

Apart from these experiments, we also ran the algorithms on the annotated E. coli genome (GenBank accession number U00096). In this set of experiments, we split the genome into sequences of a length of approximately 10,000, for which we ran the algorithms as previously described. We found the same trends as in the performance for simulated data (results not shown). When using the E. coli data, we also recorded the running time of the algorithms, and we found that the restricted algorithms are about 45 times slower than the standard algorithms, which is faster than the worst case scenario, which would lead to a \(k \cdot |Q|^2 = 7 \cdot 5^2 = 175\) slowdown, as the average expectation of the number of patterns per sequence was \(k = 7\).

9.3 Conclusions

We have introduced three novel algorithms that efficiently combine the theory of Hidden Markov Models with automata and pattern matching to recover pattern occurrences in the hidden sequence. First, we computed the distribution of the number of pattern occurrences by using an algorithm similar to the forward algorithm. This problem has been treated in [10] by means of Markov chain embedding, using simple finite sets of strings as patterns. Our method is, however, more general, as it allows the use of regular expressions.

From the occurrence number distribution, we calculated the expected number of pattern matches, which estimated the true number of occurrences with high precision. We then used the distribution to alter the prediction given by the two most widely used decoding algorithms: the Viterbi algorithm and the posterior-Viterbi algorithm. We have shown that in the case of the Viterbi algorithm, which finds the best global prediction, using the expected number of pattern occurrences greatly improves the prediction, both at the nucleotide and gene level. However, in the case of the posterior-Viterbi algorithm, which finds the best local prediction, such an addition only fragments the predicted genes, leading to a poorer prediction. Overall, deciding which algorithm is best depends on the final measure used, but as our focus was on finding genes, we conclude that the restricted Viterbi algorithm showed the best result.
9. **Resticted algorithms for HMMs**

As the distribution obtained from the restricted forward algorithm facilitates the calculation of the distribution of the waiting time until the occurrence of the $k$th pattern match, the restricted Viterbi algorithm could potentially be further extended to incorporate this distribution while calculating the joint probability of observed and hidden sequences.

Weighted transducers [147] are sequence modeling tools similar to HMMs, and analyzing patterns in the hidden sequence can potentially also be done by composition of the transducers, which describe the HMM and the automaton.

Our method can presumably be used with already existing HMMs to improve their prediction, by using patterns that reflect the problems studied using the HMMs. For example, in [9], an HMM is used for finding frame-shifts in coding regions. In this situation, the pattern would capture the sequence of hidden states that corresponds to a frame-shift. In HMMs used for phylogenetic analysis [130, 183], the hidden states represent trees, and an event of interest is a shift in the tree. A pattern capturing a change in the hidden state could thus aid in improving the prediction. In HMMs built for probabilistic alignments [128], the pattern could capture the start of an indel, and our method could potentially aid in finding a more accurate indel rate estimate. There is therefore a substantial potential in the presented method to successfully improve the power of HMMs.

**Acknowledgments**

We are grateful to Jens Ledet Jensen for useful discussions in the initial phase of this study.
Evolving stochastic context-free grammars for RNA secondary structure prediction

James WJ Anderson¹ Paula Tataru² Joe Staines³ Jotun Hein¹ Rune Lyngsø¹

The paper presented in this chapter was published in 2012 in BMC Bioinformatics.


Apart from minor typographical and formatting changes, the content of this chapter is identical to the journal paper.

Abstract

Background: Stochastic Context-Free Grammars (SCFGs) were applied successfully to RNA secondary structure prediction in the early 90s, and used in combination with comparative methods in the late 90s. The set of SCFGs potentially useful for RNA secondary structure prediction is very large, but a few intuitively designed grammars have remained dominant. In this paper we investigate two automatic search techniques for effective grammars – exhaustive search for very compact grammars and an evolutionary algorithm to find larger grammars. We also examine whether grammar ambiguity is as problematic to structure prediction as has been previously suggested.

Results: These search techniques were applied to predict RNA secondary structure on a maximal data set and revealed new and interesting grammars, though none are dramatically better than classic grammars. In general, results showed that many grammars with quite different structure could have very similar predictive ability. Many ambiguous grammars were found which were at least as effective as the best current unambiguous grammars.

Conclusions: Overall the method of evolving SCFGs for RNA secondary structure prediction proved effective in finding many grammars that had strong predictive accuracy, as good or slightly better than those designed manually. Furthermore, several of the best grammars found were ambiguous, demonstrating that such grammars should not be disregarded.

Availability: The data and source code used in this article are available under GPL 3.0 at http://www.stats.ox.ac.uk/~anderson/evolgram.html

¹Department of Statistics, University of Oxford, Oxford, UK
²Bioinformatics Research Centre, Aarhus University, Aarhus, Denmark
³Department of Computer Science, University College London, London, UK
10. Evolving SCFGs

10.1 Background

RNA secondary structure prediction is the process of predicting the position of hydrogen bonds in an RNA molecule based only on its nucleotide sequence. These predictions can be used to better understand the functioning of cells, characteristics of gene expression and the mechanisms involved in protein production [142]. Early attempts at systematic prediction include [161]; who simply evaluated all possible structures with respect to free energy functions. Later, thermodynamic principles were used to advance free energy methods in algorithms such as UNAfold [138] and RNAfold [86]. (For a good overview of RNA secondary structure prediction, see [65].) Whilst energy minimization models have proved popular, SCFG based methods also have their merits.

Stochastic context-free grammars

A context-free grammar $G$ (henceforth abbreviated to “grammar”) is a 4-tuple $(N, V, P, S)$ consisting of a finite set $N$ of non-terminal variables; a finite set $V$ of terminal variables that is disjoint from $N$; a finite set $P$ of production rules; and a distinguished symbol $S \in N$ that is the start symbol. Each production rule replaces one non-terminal variable with a string of non-terminals and terminals.

A grammar generating strings which may be interpreted as addition/multiplication expressions using only the number 1, may be represented thus

$S \rightarrow F + S | F$

$F \rightarrow 1 | (S) | F \times F.$

Note that each instance of $S$ (standing for sum) generates a sum of $n \geq 1$ terms ($F + F + \ldots F$), and each $F$ (standing for factor) generates a 1, a product of terms ($F \times F$), or a whole expression within parentheses. It should be clear that many different expressions might be generated: $1+1+1, (1+1^2)1+1(1)$, and so forth. Formally, the grammar has non-terminal variables $S, F$, terminal variables $(), +, *, 1$, production rules $S \rightarrow F+S, S \rightarrow F, F \rightarrow 1, F \rightarrow (S)$ and start symbol $S$.

The order in which the production rules are used forms the derivation of a string. One valid derivation would be $S \Rightarrow F \Rightarrow (S) \Rightarrow (F) \Rightarrow (1)$, generating the string ‘(1)’ and using the sequence of production rules $S \rightarrow F, F \rightarrow (S), S \rightarrow F, F \rightarrow 1$.

A SCFG is a grammar with an associated probability distribution over the production rules which start from each $T \in N$. Beginning with the start symbol and following production rules sampled from the relevant distribution, a string of terminal variables can be produced (if the grammar terminates). Choosing nucleotide symbols or the three characters used in dot-parenthesis notation as terminal variables, SCFGs can be constructed which produce strings corresponding to nucleotide sequences or secondary structures.

Application of SCFGs to RNA secondary structure prediction

The use of SCFGs in RNA secondary structure prediction was based on the success of Hidden Markov Models (HMMs) in protein and gene modeling [113]. Any attempt to apply HMMs to RNA secondary structure was prevented by the long-range interactions in RNA [167]. SCFGs, being generalizations of HMMs, offer a solution. This was first exploited by [179] and then developed by others (e.g. [118]).

The Pfold algorithm [107, 108] is one of the most successful approaches using SCFGs. It is designed to produce an evolutionary tree and secondary structure from an aligned set of RNA sequences. Pfold uses a SCFG designed specifically for RNA secondary structure prediction (denoted KH99 in this paper). Therefore, when only considering a single sequence, Pfold is simply a SCFG prediction method. There are other approaches which predict secondary structures from aligned RNA sequences, such as RNAalifold [15] and Turbofold [80]. However, we are concerned with the single-sequence prediction problem, so these are not used here.

While KH99 was effective, it seems to have been chosen relatively arbitrarily, in that there is little discussion about the motivation behind the choice of production rules. This problem was addressed by [44], in which nine different lightweight SCFGs were evaluated on a benchmark set of RNA secondary structures. The set of grammars they tested, however, was by no means exhaustive. All of these grammars were constructed by hand and there was little motivation for their production rules (except for the extension to stacking grammars: production rules such as $p^{\hat{b}} \rightarrow a^p a \hat{a}$ for nucleotides $a, \hat{a}, b, \hat{b}$). This suggested that a computational search of a large space of grammars might find stronger grammars, which is what we have attempted in this paper.

Evolutionary approaches have already been
implemented for HMMs. Indeed, [211] used one to find the best HMM for protein secondary structure prediction. A mild improvement was found compared to HMMs which had been constructed by hand. It is hard to tell quite how conclusive the results were since the limited size of the data set forced training and testing to be done on the same data. Given the size of the data set, over-training may have caused unreasonably high quality predictions. Clearly though, the method is potentially very powerful.

10.2 Methods

In this paper, only structure generating grammars are considered (i.e. those which have terminal variables in \{\text{,}, \text{.}\}). Strings generated by these grammars uniquely define secondary structures, with a dot ‘.’ representing an unpaired nucleotide and an opening parenthesis indicating pairing with its corresponding closing parenthesis. Once a structure is generated, a generative model for sequences is to stochastically allocate nucleotides to each site according to the frequency of occurrence (paired and unpaired) in some set of trusted sequences and structures.

Normal forms

To develop algorithms for analyzing sequences under grammatical models, it is convenient to restrict the grammar to a normal form, with only a few possible types of productions. The normal form most commonly used is Chomsky Normal Form (CNF), as every context-free grammar is equivalent to one in CNF. However, a grammar in CNF cannot introduce the corresponding parentheses of paired nucleotides in a single production, and therefore does not capture structure in a straightforward manner. Thus it was necessary to create a new double emission normal form (so called because paired bases are emitted simultaneously) which was able to capture the fundamental features of RNA secondary structure: branching, unpaired bases, and paired bases. For any combination of non-terminals \(T, U, V\) we allow only rules of the following form:

\[
\begin{align*}
T & \rightarrow UV \\
T & \rightarrow . \\
T & \rightarrow (U)
\end{align*}
\]

This normal form allows the development of the structural motifs commonly found in RNA. For example (where \(V_i\) correspond to non-terminals) base-pair stacking can be generated by rules of the type \(V_1 \rightarrow (V_1), \) hairpins by \(V_1 \rightarrow (V_2), V_2 \rightarrow V_2 V_3\) \(V_2 V_3\) and \(V_3 \rightarrow . .\), and bulges by \(V_1 \rightarrow (V_2), V_2 \rightarrow V_2 V_1\) and \(V_3 \rightarrow .(V_2).\)

Furthermore, with the exception of the ability to generate empty strings, this normal form allows the expression of dependencies of any context-free grammar producing valid structures. It was also designed to avoid cyclical productions; that is, combinations of production rules which result in the same string that they started from. These are undesirable as they permit a countably infinite number of derivations for some strings. For this reason, rules of the form \(T \rightarrow \epsilon\) and \(T \rightarrow U\) were not considered. There is, however, nothing intrinsically wrong with these rules; it is quite possible to create strong grammars with these rules present (many of those used in [44] have rules of this type).

As a result of eliminating these rules, many grammars already established in RNA secondary structure prediction cannot be exactly replicated, since they are not initially in the above normal form. For example, the KH99 contains the rule \(S \rightarrow L.\) As this normal form is an extension of CNF, any context-free grammar can be converted to this normal form, maintaining paired terminal symbol emissions. For example if \(S \rightarrow L\) and \(L \rightarrow .)(F)\), the forbidden rule \(S \rightarrow L\) would be replaced by \(S \rightarrow .)(F).\) This produces the same strings, and a given probability distribution for stochastic grammars can even be conserved. However, the transformation will often change the set of parameters in the model, which may result in different predictions when the production rule probabilities are inferred.

Secondary structure prediction

Secondary structure can be predicted by two methods, both of which are used here. Firstly, one can find the maximum likelihood derivation of a sequence, during which a structure is generated. The Cocke-Younger-Kasami (CYK) algorithm [218] determines, by dynamic programming, the probability of the most likely derivation, and so backtracking can be used to find the most likely structure. It is designed for grammars in CNF, though there are established methods of dealing with grammars in a different normal form [32].

Secondly, one can employ a posterior decoding method using base-pairing probability matrices. The base-pair probability matrix for a SCFG
are obtained via the inside and outside algorithms [117]. The secondary structure with the maximum expected number of correct positions can then be calculated via dynamic programming. Our decoding algorithm follows [37], including a $\gamma$ parameter specifying the trade-off between correct base pairs relative to correct unpaired positions. For assessing the fitness of a grammar in the genetic algorithm, a value of $\gamma = 2$ was used, so as to maximize the expected number of positions correctly predicted.

Both methods were used in the search, as this additionally gave a chance to compare the two prediction methods.

Ambiguity and completeness

A grammar is said to be ambiguous if it produces more than one derivation for a given structure [171]. If structure $A$ has one derivation with probability 0.3, and structure $B$ two derivations, each with probability 0.25, the CYK algorithm will choose structure $A$, while structure $B$ is more probable. This may reduce the quality of predictions using the CYK algorithm. The posterior decoding sums over all derivations in prediction, so might be affected less by grammar ambiguity.

Nine grammars are tested in [44], of which two are ambiguous. They find that the CYK algorithm does choose suboptimal structures, and that the ambiguous grammars perform poorly relative to the unambiguous grammars. Consequently, efforts have been made to avoid ambiguity [171]. The conclusion that ambiguous grammars are undesirable is not necessarily justified. The ambiguous grammars in [44] are small, with at most two non-terminal variables, and one might expect them to be ineffective regardless. Their poor predictive quality may be due to deficiency in design rather than ambiguity.

We define a grammar to be complete if it has a derivation for all possible structures which have no hairpins shorter than length two. Clearly the ability to generate all structures is desirable for a grammar. However, one should be careful not to overemphasize this desirability, even for a complete grammar, once parameters have been inferred, converting it to a SCFG, it is unlikely that all structures have a probability significantly different from zero for any sequence. Similarly, the posterior decoding is not prone to grammar incompleteness in the same way that CYK is since, in theory, after obtaining the probabilities of unpaired and paired positions, structures which cannot be derived with the grammar can still be predicted. One can perform a heuristic test for completeness by testing on a sample of strings. Ambiguity can also be checked for heuristically [171], but determining grammar ambiguity and completeness is undecidable [90].

Practically, it is very difficult to ensure both unambiguity and completeness. A grammar that is complete and unambiguous cannot be simply modified without compromising one of the properties. Adding any production rules (if they are ever used) will create ambiguity by providing additional derivations. Equally, removing production rules will create incompleteness (unless the rule is never used in a derivation), as the original grammar is assumed unambiguous. Because of this, an automated grammar design based on simple-step modification is practically impossible without creating ambiguous and incomplete grammars. Moreover, grammars that are unambiguous and complete are vastly outnumbered by grammars that are not. Therefore, grammars not possessing these desirable qualities must be considered and as a result our grammar search serves as a test of the capabilities of ambiguous or incomplete grammars.

Parameter inference

Training data, consisting of strings of nucleotides and trusted secondary structures, is used to obtain the probabilities associated with each production rule, as well as paired and unpaired nucleotide probabilities. If derivations are known for the training sequences, then there are simple multinomial maximum likelihood estimators for the probabilities. Usually, though, the derivation is unknown. Again, one can estimate probabilities by finding derivations for the training set using CYK, or by the inside and outside algorithms.

For the CYK algorithm, in the case of ambiguous grammars, one cannot know which derivation produced the known structure, so probabilities cannot be obtained. Consequently, we train these grammars using the same approach as [44] to ensure comparability. That is, we randomly select one derivation. For unambiguous grammars, such as KH99, this has no effect on the training. As with the prediction, inside-outside training works for unambiguous and ambiguous grammars alike.

Again, both CYK and inside-outside were used for parameter inference in the search and evaluation.
Evolutionary algorithm

With the double emission normal form, for $m$ non-terminal variables there are $2^{m^3 + m^2 + m}$ grammars ($m^3$ production rules of type $T \rightarrow UV$, $m^2$ of type $T \rightarrow (U)$ and $m$ of type $T \rightarrow .$). An evolutionary algorithm would allow for efficient exploration of the space of grammars in the above normal form. The way that the algorithm searches the space is determined by the design of the initial population, mutation, breeding and selection procedure. To find effective grammars, these must be well designed.

Initial population

When forming the initial population, the size of the space of grammars quickly becomes problematic. The space is clearly large, even for small $m$, so the population size cannot approach that usually afforded in evolutionary algorithms [70]. We start with an initial population of small grammars, and use mutation and breeding rules which grow the number of non-terminal variables and production rules. Our initial population comprised sixteen grammars, of the form

$$S \rightarrow \{ SS \mid SB \mid BS \mid BB \mid (S) \} .$$

$$B \rightarrow .$$

where between zero and four of the $S \rightarrow UV$ rules were excluded. We also tried initial populations containing the SCFGs from [44] to consider examining SCFGs close to these.

Mutation

Mutations constitute the majority of movement through the search space, so are particularly important. They give the grammar new characteristics, allow it more structural freedom, and add production rules which may be used immediately or may lie dormant. For non-terminals $V_i \in N$, and corresponding production rules $P_{V_i}$, the allowed stochastic mutations were

- The start variable (and corresponding production rules) change,
- A production rule is added or deleted,
- A new non-terminal variable $V'$ is added along with two new rules that ensure that $V'$ is reachable and that $P_{V'}$ is not empty,
- A non-terminal variable is created with identical rules to a pre-existing one,
- A production rule of the form $V_i \rightarrow V_j V_k$ is changed to $V_i \rightarrow V_j V_l$, $V_i \rightarrow V_j V_k$ or $V_i \rightarrow V_j V_p$, or production rule of the form $V_i \rightarrow (V_j)$ is changed to $V_i \rightarrow (V_k)$.

This form of mutation is very basic, but allows many structural features to develop over generations. The rate of mutation determines movement speed through the search space and development of these structural features. Adding rules too slowly prevents grammars from developing structure, while too many results in a lot of ambiguity and thus creates ineffective grammars. Deleting rules almost always results in a worse grammar. To aid the grammar design, especially in consideration of facets of the normal form, the rule $B \rightarrow .$ was kept constant in the evolutionary process.

More complex mutation is clearly possible. The derivation could be used to find the rules used more often and make mutations of those rules more or less likely. A model for simultaneous mutations could be developed, which might be able to make use of expert understanding of RNA structure, in combination with an evolutionary search. We have found the above model to give sufficient mobility in the search space, and therefore did not investigate other extensions.

Breeding

The breeding model forms a grammar which can produce all derivations of its parent grammars. The grammar $G$ formed from breeding $G_1$ and $G_2$ has start symbol $S$, non-terminals $V_1$, $V_2$, ..., $V_n$ and $W_1$, $W_2$, ..., $W_m$, $B$, terminals $.,()$ and production rules

- $P_S = P_{S_1} \cup P_{S_2}$,
- For $V_i$: $P_{V_i}$, where all occurrences of $S_1$ are replaced with $S$,
- For $W_i$: $P_{W_i}$, where all occurrences of $S_2$ are replaced with $S$.

This breeding model was chosen to keep the size of the grammar relatively small, whilst allowing expression of both bred grammars to be present in derivations.

Selection

We grow the population in each generation by introducing a number of newly mutated or bred grammars, then we pare it back to a fixed population size by stochastic elimination. We determine the probability of elimination of a grammar by the inverse of some fitness measure. Fitness functions
we use include mountain metric distances \[149\] between the predicted structure and the trusted structure, sensitivity (True Positives/(True Positives + False Negatives)) and positive predictive value (PPV) (True Positives/(True Positives + False Positives)). We follow the definition of accuracy in \[44\], where a base pair constitutes a true positive if it is present in both the predicted and true structures, false positive if present in only the predicted structure, and false negative if present in only the trusted structure. We also tried these accuracy measures in combination with other factors in the fitness function, including largest correctly predicted length, inability to predict structures, and cost for grammar complexity.

10.3 Results and discussion

Figure 10.1 shows a typical realization of the search process. The average fitness (here, smaller fitness is desirable) of the population is shown, as well as the fitness of the best recorded (champion) grammar. The average fitness of the population falls consistently as stronger grammars are found for approximately 100 generations, and then only minor improvements to the champion grammar were found. However, the population fitness continues to fluctuate as areas around the local optimum are searched.

Across all our experiments, over 300,000 grammars were searched. A number of strong grammars were found using both CYK and IO training and testing, denoted GG1–GG6 (Table 10.1). KH99' is KH99 in the double emission normal
Figure 10.1: Fitness evolution. The change over generations in average fitness of population, and the fitness of the best SCFG. Here, a lower fitness is more desirable, the SCFG predicting better secondary structure. Many improvements to both the whole population and best SCFG are made in the first 100 or so generations. After this, the best SCFG does not become much better, but the average population fitness continues to fluctuate. Clearly the algorithm continues to explore alternative SCFGs and tries to escape the local optimum.

Table 10.1: Grammars found using CYK and IO training and testing.

<table>
<thead>
<tr>
<th>KH99'</th>
<th>A → BA</th>
<th>.</th>
<th>(C)</th>
<th>A → AA</th>
<th>BA</th>
<th>.</th>
<th>(A)</th>
<th>(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>.</td>
<td>(C)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>BA</td>
<td>(C)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GG1</th>
<th>B</th>
<th>.</th>
<th>(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>BA</td>
<td>(C)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GG2</th>
<th>A → AB</th>
<th>BA</th>
<th>BB</th>
<th>CB</th>
<th>BC</th>
<th>.</th>
<th>(B)</th>
<th>(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>AA</td>
<td>AB</td>
<td>BA</td>
<td>BB</td>
<td>BC</td>
<td>CA</td>
<td>CB</td>
<td>.</td>
</tr>
</tbody>
</table>

| GG3     | A → AB | BA | BB | AA | DD | (A) | (B) | (C) | (D) |
|---------|-------|---|-----|----|-----|----|------|------|
| B       | .     |   |     |    |     |    |      |      |
| C       | AA    | . | (D) |

<table>
<thead>
<tr>
<th>GG4</th>
<th>A → CC</th>
<th>CB</th>
<th>BC</th>
<th>EC</th>
<th>(A)</th>
<th>(E)</th>
<th>B</th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>AA</td>
<td>HF</td>
<td>(G)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GG5</th>
<th>D → .</th>
<th>(E)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>GG6</th>
<th>A → DE</th>
<th>AB</th>
<th>BA</th>
<th>AH</th>
<th>.</th>
<th>(F)</th>
<th>(H)</th>
<th>E → .</th>
<th>(H)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>(H)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>BB</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

form. Results on the sensitivity, PPV, and F-score of each grammar can be found in Table 10.2, in addition to the benchmark with the data from [44], and results on different training and testing methods can be found in Table 10.3. Table 10.3 also gives the scores of the combined best prediction, calculated by selecting, for each structure, the prediction with the highest F-score, and then recording the sensitivity, PPV, and F-score for that prediction. This shows grammars with very different structures perform well on the same (full evaluation) data set. KH99’ is still a strong performer, but we have shown that there exist many others which perform similarly (these GG1–GG6 form just a subset of the good grammars found in the search).
10. Evolving SCFGs

<table>
<thead>
<tr>
<th>Grammar</th>
<th>KH99'</th>
<th>GG1</th>
<th>GG2</th>
<th>GG3</th>
<th>GG4</th>
<th>GG5</th>
<th>GG6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensitivity</td>
<td>0.496</td>
<td>0.505</td>
<td>0.408</td>
<td>0.413</td>
<td>0.474</td>
<td>0.469</td>
<td><strong>0.526</strong></td>
</tr>
<tr>
<td>PPV</td>
<td>0.479</td>
<td>0.481</td>
<td><strong>0.551</strong></td>
<td>0.550</td>
<td>0.545</td>
<td>0.467</td>
<td>0.479</td>
</tr>
<tr>
<td>F-score</td>
<td>0.478</td>
<td>0.441</td>
<td>0.473</td>
<td>0.470</td>
<td>0.461</td>
<td>0.339</td>
<td>0.488</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Grammar</th>
<th>KH99'</th>
<th>GG1</th>
<th>GG2</th>
<th>GG3</th>
<th>GG4</th>
<th>GG5</th>
<th>GG6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensitivity</td>
<td>0.465</td>
<td>0.466</td>
<td>0.372</td>
<td>0.379</td>
<td>0.408</td>
<td><strong>0.487</strong></td>
<td>0.465</td>
</tr>
<tr>
<td>PPV</td>
<td>0.406</td>
<td>0.405</td>
<td>0.643</td>
<td><strong>0.646</strong></td>
<td>0.344</td>
<td>0.432</td>
<td>0.376</td>
</tr>
<tr>
<td>F-score</td>
<td><strong>0.480</strong></td>
<td>0.468</td>
<td>0.466</td>
<td>0.472</td>
<td>0.430</td>
<td>0.479</td>
<td>0.451</td>
</tr>
</tbody>
</table>

Table 10.2: Sensitivity and specificity of evolved SCFGs and other prediction methods. The sensitivities, PPVs, and F-scores of grammars GG1–GG6 and KH99' on the evaluation set and on the [44] data, as well as the benchmarked sensitivities and PPVs of UNAfoldv3.8 and RNAfoldv1.85 on the [44] data. Results for KH99 and Pfold are taken from [44].

<table>
<thead>
<tr>
<th>Method</th>
<th>KH99</th>
<th>UNAfold</th>
<th>RNAfold</th>
<th>Pfold</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE data</td>
<td>Sensitivity</td>
<td>0.47</td>
<td>0.558</td>
<td>0.558</td>
</tr>
<tr>
<td>PPV</td>
<td>0.45</td>
<td>0.501</td>
<td>0.495</td>
<td>0.69</td>
</tr>
</tbody>
</table>

Table 10.3: Sensitivity and specificity of evolved SCFGs using different training and testing methods. The sensitivities, PPVs, and F-scores of grammars GG1–GG6 and KH99' on the evaluation set, using different methods of training and testing. ‘CYK’ indicates that the CYK algorithm was used, and ‘IO’ that the inside and outside algorithms were used. The column ‘Best’ was calculated by selecting, for each structure, the prediction with the highest F-score, and then recording the sensitivity, PPV, and F-score for that prediction. It is perhaps not surprising that the ‘best’ predictions for CYK are better than the ‘best’ predictions for IO, as IO is in some sense averaging over all predictions. One might expect the predictions to be more similar than those from CYK, as seen by comparing IO values for GG6 and ‘best’, giving less increase when considering those with best F-score.

GG1 is KH99' with two rules added, \(A \rightarrow AA\) and \(A \rightarrow (A)\). These rules were used infrequently (probabilities 0.007 and 0.047 respectively). The mild improvement in functionality allows for an additional fraction of base pairs to be correctly predicted.

GG2 and GG3 were found using the posterior decoding version of the evolutionary algorithm. They have a high density of rules, that is many rules for each non-terminal variable. Particularly, GG2 has almost all of the rules it is possible for it to have, given \(B \rightarrow \) . is kept constant through the evolutionary algorithm. Given this, it is not surprising that they performed poorly using the CYK training and testing. However, with posterior decoding, the base-pair probabilities are still effective for good predictions.

GG4 has only two variables (\(A\) and \(C\)) used almost exclusively in producing base pairs. It then uses various exit sequences to generate different sets of unpaired nucleotides and returns to producing base pairs. Finally, GG5 and GG6 are typical of larger grammars we have found with complex structure. It is not obvious to us how their structure relates to their success in secondary structure prediction. GG4, GG5, and GG6 were all found using the CYK version of the evolutionary algorithm, and perhaps their complex structure can be accredited to this. GG6 is a strong performer throughout, particularly when considering...
10.3. Results and discussion

Figure 10.2: Sensitivity/PPV curve. A graph showing how sensitivity and PPV change for grammars when the posterior decoding parameter $\gamma$ is changed to alter the frequency of base-pair prediction. The parameter $\gamma$ ranged from 0.05 to 5 in increments of 0.05 for each grammar, and the sensitivity and PPV were measured on the full evaluation set. Grammar GG6 shows further strong performance here, having the largest area underneath the graph.

Most grammars achieved lower predictive power on the Dowell and Eddy dataset. The difference in performance between KH99 and KH99′ is small and confirms that the representation of KH99 as KH99′ is a good one. Particularly noteworthy is the performance of GG3 and GG5. GG3 has had a considerable increase in PPV, likely due to the posterior decoding prediction method. Given many of the structures in the Dowell and Eddy dataset contain pseudoknots, other grammars score poorly trying to predict pairs where there are not, in contrast to GG3. By predicting fewer base pairs, GG3 gains higher PPV as more of them are correct, but lower sensitivity. GG5 is a grammar which was unremarkable in its results on the original data set, however, it has outperformed the rest of the grammars on the benchmark set and is the only grammar with improved sensitivity when compared to the RNASTRAND dataset.

Figure 10.2 shows the sensitivity-PPV curve for grammars KH99′ and GG1–GG6. This was produced using the posterior decoding method by varying the parameter $\gamma$. GG6 again proves to do slightly better than the other grammars, having the largest area underneath the curve. KH99′ does not distinguish itself much from the other grammars, being in the middle in terms of area underneath the curve.

Overall, the grammars found in the evolutionary search still perform well because they are not over-adapted to deal with the original data. Determining which is best depends on the measure of strength of prediction, whether the size of the grammar is a concern, ability to approximate structures with pseudoknots effectively, and so on. However, it is clear that a selection of effective grammars has been found. Results shown by UNAfold and RNAfold continue to be superior to those produced by SCFG methods.

We also checked that the grammars obtained from the evolutionary algorithm do not merely produce similar structures to KH99′ by using different derivations. To do this we define the relative sensitivity of method A with respect to method B as the sensitivity of method A as a predictor of the structures produced by method B. The relative PPV is defined in a similar manner. We then compared the predictions of the grammars by building a heat map of the relative sensitivities and PPVs (Fig. 10.3), using the evaluation set. As expected, KH99′ and GG1 predict almost identical structures, as they are highly similar. Similarly, it is perhaps not surprising that GG2 and GG3 have very similar predictions given they produce structure through posterior decoding. The rest of the methods have sensitivity and PPV relative to other prediction methods of approximately 0.4–0.6. As they are designed to predict RNA secondary structure using the same training set, one would expect some similarity in the predictions, although not as much as with KH99′ and GG1. This is confirmed by our results, suggesting that the new grammars produce different kinds of structures which are good representations of RNA secondary structure.

Further analysis of KH99

To test the local features of the space, we evaluated variations of KH99′ against the full data set. Where a single rule was deleted, only one grammar had prediction accuracy of the order of KH99′.
10. Evolving SCFGs

Figure 10.3: Relative sensitivities/PPVs. A heat map showing the relative sensitivities and PPVs of the different prediction methods, or between prediction method and known structure. KH99′ and GG1, produce very similar structures which is not surprising, given they were found by changing one and two rules of KH99′ respectively. Otherwise, the methods have relative sensitivities/PPVs of approximately 0.5–0.6, which is as expected, given they are all designed to predict RNA secondary structure. However, it is clear that they are markedly distinct from KH99′ in their structure predictions.

Figure 10.4: Local search results. Summary of the effects of adding one (giving 32 grammars) or two (giving 961 grammars) production rules to KH99′. The plot shows the cumulative proportion of grammars with given sensitivity. The grammars’ sensitivity is mostly still equal to the sensitivity of KH99′, with only a few outliers. GG1 was the top outlier for two production rule added. In this sense the space is reasonably flat.

Figure 10.5: Brute force search. The distribution of sensitivity and corresponding PPV of grammars with at most 2 nonterminal variables. Approximately one quarter of grammars have sensitivity 0, as many cannot produce long strings. It is only the larger grammars that start to predict long strings which might correspond to structure. However, the prediction quality is still poor by both measures.
Table 10.4: Ambiguity and completeness. Ambiguity and completeness of KH99′ and GG1–GG6 grammars. All grammars found in the search were ambiguous. Some of the grammars found (GG4 and GG5), are incomplete but heuristically it seems that the structures that cannot be generated have little biological relevance.

This is the grammar without the rule $A \rightarrow (C)$ (which would have been used only infrequently in KH99′, with probability 0.014). It is clear that deleting rules has a strong negative effect on the predictive power of KH99′, given that no others have sensitivity greater than 0.25. Of course, this might be expected given that this SCFG has been constructed manually, and it is therefore unlikely to have unnecessary production rules.

With addition of rules, the number of grammars to check quickly becomes large. With one production rule added, 32 grammars must be evaluated, with two added this increases to 496. A similar local search for larger grammars would be impractical, since there are many more grammars with one or two altered production rules (for GG6, there are 584 grammars with only one new production rule, and 170,236 with two). Ambiguity of tested grammars had little or no effect. Results of this local search can be seen in Fig. 10.4.

**Brute force search**

The brute force search illustrated how, with this normal form, larger grammars are needed to provide effective prediction. Most small grammars will only be able to produce one type of string. Also, it suggested that the existing grammars are close to locally optimal and that the space around them is quite flat, demonstrating the need for intelligent searching methods. Figure 10.5 illustrates the distribution of sensitivity across the space of grammars with at most 2 non-terminal variables. No grammar has sensitivity higher than 0.25 and approximately one quarter of grammars have sensitivity 0 (those which cannot produce long strings).

**Ambiguity and completeness**

One of the results of the search which we find most interesting is the ambiguity and completeness of GG1–GG6, shown in Table 10.4. All grammars found in the search were ambiguous, and still predicted structure effectively.

In particular, ambiguous grammar GG1 performed better than KH99′, being a slight modification of it. Particularly, it is clear that GG2 and GG3 have many different derivations for each structure, and their strong performance relies on this ambiguity, as they perform poorly when tested with CYK. GG5 demonstrates further that ambiguous grammars can even be effective at approximating structures with pseudoknots. The effectiveness of some ambiguous grammars is likely due to the prediction algorithm picking structures that, whilst perhaps suboptimal, are close to what the best prediction would be. Clearly there is room for a further investigation into quite why some grammars cope better with ambiguity than others.

Similarly, it might be surprising that some of the grammars found (GG4 and GG5), are incomplete. However, heuristically it seems that the structures that cannot be generated have little biological relevance (e.g. GG4 cannot generate “(...)(...)(...)”). In some sense therefore, the incompleteness is permissible, as the grammar is still able to generate any relevant structure.

**10.4 Conclusions**

Our brute force search and search around KH99 demonstrate that intelligent searching methods are necessary, and overall, the method of evolving SCFGs for RNA secondary structure prediction proved effective. We found many grammars with strong predictive accuracy, as good or better than those designed manually. Furthermore, several of the best grammars found were both ambiguous and incomplete, demonstrating that in grammar design such grammars should not be disregarded.

One of the strengths of the method is the ease of application and effectiveness for RNA structure problems. In particular, grammatical models are used in phylogenetic models of RNA evolution [21]
which make use of manually constructed grammars, and so the accuracy might be improved with automated grammar design.

Overall though, whilst many grammars have been found with good predictive power, the space of grammars grows rapidly with the number of non-terminals, so we cannot conclude that no better grammars exist. The effectiveness of the search heuristic is supported by the fact that we consistently find grammars on par with the best manually created grammars. The equally consistent inability to achieve any significant improvement on this level of performance, and the relative limited prediction correlation between the many good grammars found points to the inherent challenge of grammar design, or indeed to the limitations of SCFG based methods as a whole. It appears that the number of grammars able to achieve this level of performance is large, and may depend little on the overall grammar structure, and at the same time it appears that a performance improvement may be difficult or impossible to achieve with a grammatical approach.

**Author’s contributions**

JWJA conceived the idea in discussion with RL and JH. JWJA then developed the methodology with PT and JS, with help from RL. PT and JS then designed and wrote the code, and results were analyzed and written up by JWJA, with help from PT and JS. All authors were involved in critical redrafting of the manuscript.

**Acknowledgments and funding**

JWJA would like to thank the EPSRC for funding. JH would like to acknowledge the Miller Institute for funding and hospitality. All authors would like to thank the Department of Plant Science, University of Oxford for their support and use of facilities.
Oxfold: kinetic folding of RNA using stochastic context-free grammars and evolutionary information

James WJ Anderson¹ Pierre A Haas² Leigh-Anne Mathieson³
Vladimir Volynkin⁴ Rune Lyngsø¹ Paula Tataru⁵ Jotun Hein¹

The paper presented in this chapter was published in 2013 in Bioinformatics.


Apart from minor typographical and formatting changes, the content of this chapter is identical to the journal paper.

Abstract

Motivation: Many computational methods for RNA secondary structure prediction, and, in particular, for the prediction of a consensus structure of an alignment of RNA sequences, have been developed. Most methods however ignore biophysical factors such as the kinetics of RNA folding; no current implementation considers both evolutionary information and folding kinetics, thus losing information which, when considered, might lead to better predictions.

Results: We present an iterative algorithm, Oxfold, in the framework of stochastic context-free grammars, that emulates the kinetics of RNA folding in a simplified way, in combination with a molecular evolution model. This method improves considerably upon existing grammatical models that do not consider folding kinetics. Additionally, the model compares favorably to non-kinetic thermodynamic models.

Availability: http://www.stats.ox.ac.uk/~anderson

¹Department of Statistics, University of Oxford, Oxford, UK
²Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, UK
³Department of Computer Science, University of British Columbia, Canada
⁴European Bioinformatics Institute, Hinxton, UK
⁵Bioinformatics Research Centre, Aarhus University, Aarhus, Denmark
11. Introduction

The function of ribonucleic acid molecules (RNA) is known to depend on their three-dimensional structure, which, in turn, depends on their secondary structure, a scaffold of basepairs formed by hydrogen bonds between nucleotides, for thermodynamic stability and molecular function. Accurate prediction of RNA secondary structures, however, falls short of being adequately solved. By contrast, the spreading of next-generation sequencing technologies and new methods in transcriptomics have increased the importance of RNA secondary structure prediction. This is exemplified by the growing amount of biological RNA data available in databases such as Rfam [66] and RNA STRAND [8].

Computational RNA secondary structure prediction methods have been used for a number of years: some of the first attempts (e.g. [161]) simply evaluate the free energy of all possible secondary structures, postulating that the minimum free energy structure is the functional one. These thermodynamic models were later refined to take into account biological and thermodynamic principles, and are used to great effect in algorithms such as RNAFold [86] and UNAFold [138] which rely on a large number of experimentally determined parameters.

Alternative approaches use the framework of Stochastic Context-Free Grammars (SCFGs) to find the most likely structure given their training data, postulating that this is the functional structure. Among the first to describe such models were [55]. Many different grammatical models for RNA secondary structure prediction have been implemented [6, 44, 107, 108].

If one seeks to build better grammatical models for RNA secondary structure prediction, one can take essentially two different routes:

- build more complex grammars that express higher-order dependencies such as basepair stacking and, for instance, thereby emulate the nearest-neighbor model underlying thermodynamic approaches;
- include additional biological and physical information about the sequences, for example at the level of the prior pairing and unpairing probabilities of the grammar.

The first approach has been taken by [151] and [178]. The latter developed a language to translate a wide variety of probabilistic and thermodynamic models for RNA secondary structure prediction into the language of SCFGs, yielding highly complex grammars with a large number of parameters.

We however follow the second approach, which was pioneered by [107, 108], who coupled a simple grammar to an evolutionary model to obtain better estimates of the prior base-pairing probabilities when folding an alignment of RNA sequences. Most current approaches to RNA secondary structure prediction are static, insofar as they assess structures based on their constituent elements like base pairs and loops but with no contribution from the path followed to form these elements. The importance of folding mechanisms was noted by [202], and [75] who studied the folding of intermediate stems. We note that “the differences between real structures and the minimum energy states are believed to be determined mainly by defects in the energy rules used or by the existence of specific folding pathways capturing molecules in local minima” [75].

Just as comparative structure prediction is based on the observation that structure is important for function and hence conserved, since folding kinetics are important for either guiding or determining structure formation, we would expect evolution to exert selection on the kinetics, too. Previously, evolutionary models [107, 108] and kinetic models (e.g. [41, 214]) have been implemented, but have not been combined. It is therefore important to implement folding kinetics in an evolutionary framework.

In this paper, we work in the framework of the fundamental problem of predicting a consensus structure for a given, fixed alignment of RNA sequences. We incorporate folding kinetics, in a simplified way, into an evolutionary grammatical model in an iterative framework. Further, we introduce a distance function to incorporate information about the relationships between different pairs of columns, thus adopting the second of the above approaches. The resulting model is benchmarked against PPfold [193], a parallelized implementation of the Pfold algorithm of [108]. Additionally, we compare it to a thermodynamic model, RNAalifold [15, 87].

11.2 Methods

Background: grammatical models

A context-free grammar [35] is a four-tuple \((\mathcal{N}, \mathcal{V}, \mathcal{P}, S)\) consisting of a finite set \(\mathcal{N}\) of non-terminals, a finite set \(\mathcal{V}\), disjoint form \(\mathcal{N}\), of ter-
minals, a finite set \( \mathcal{P} \) of production rules, and a distinguished starting symbol \( S \in \mathcal{N} \). Each production rule replaces a non-terminal with a string of non-terminals and terminals.

For example, the context-free grammar underlying the Pf algorithm of [107, 108] is represented as

\[
S \rightarrow LS \mid L \\
L \rightarrow . \mid (F) \\
F \rightarrow LS \mid (F)
\]

It has non-terminals \( S, L, F \) and terminals \( ., (, ) \), representing unpaired and paired nucleotides in the dot-parenthesis representation of RNA secondary structures. For example, the string \((()())\) is produced by the derivation

\[
S \rightarrow L \quad \text{using rule } S \rightarrow L \\
\rightarrow (F) \quad \text{using rule } L \rightarrow (F) \\
\rightarrow (()) \quad \text{using rule } F \rightarrow (F) \\
\rightarrow (()()) \quad \text{using rule } F \rightarrow LS \\
\rightarrow (()) \quad \text{using rule } LS \rightarrow L \\
\rightarrow (()) \quad \text{using rule } L \rightarrow .
\]

In the grammar of [107], the starting symbol \( S \) produces loops, while \( F \) produces stems and \( L \) determines whether a loop position should be a single base, or the start of a new stem.

A stochastic context-free grammar (SCFG) is a context-free grammar with an associated probability distribution over the production rules. Thus each string produced by the grammar (by beginning with the starting symbol and following production rules) is given a certain probability, which gives a probability distribution over RNA secondary structures. The rule probabilities are determined by inside-outside training, an expectation maximization technique [117].

To complete the model, we require the prior probabilities of a dot representing any of the four nucleotides A, C, G, U and of a pair of parentheses representing any of the sixteen corresponding basepairs. For example, the probability of producing an A from the non-terminal L is the product of the probability of the rule \( L \rightarrow . \) and of the prior nucleotide probability of a dot representing an A. For single sequences, these are simply the frequencies observed on training data.

For alignments of multiple sequences, rather than using the simple heuristic of just multiplying the maximum likelihood estimates of the pairing probabilities of the bases in a given pair of columns in each sequence to estimate the prior base-pairing probabilities of that pair of columns, the Pf algorithm uses an evolutionary model: each pair of columns is assumed to evolve independently according to a continuous Markov process with rates given by the branch lengths of an evolutionary tree estimated from the alignment. The base-pairing probabilities are then determined by post-order traversal [61] on the evolutionary tree. Gaps in the alignment are treated as unknown nucleotides.

One possible candidate for the consensus structure is the structure with the highest probability under grammar and evolutionary model, obtained using the CYK algorithm, a dynamic programming algorithm [46]. However, this ignores contributions from other possible structures. Most current implementations therefore predict the structure with the highest expected number of correctly predicted basepairs (maximum expected accuracy, MEA, estimation). The latter is determined, using a dynamic programming algorithm, from the posterior pairing and unpairing probabilities, i.e. the pairing and unpairing probabilities given the sequence data and the model [108] (supplementary material), which, in turn, are determined from the matrices of inside and outside probabilities associated with the SCFG [117].

For RNA secondary structure prediction, it is most convenient to write the production rules in the double-emission form of [6], which only allows rules of the forms

\[
U \rightarrow . \\
U \rightarrow VW \\
U \rightarrow (V)
\]

where \( U, V, W \) denote generic non-terminals. Throughout this paper, we use the grammar of [107], rewritten in double-emission form, viz

\[
S \rightarrow LS \mid . \mid (F) \\
L \rightarrow . \mid (F) \\
F \rightarrow LS \mid (F)
\]

This slightly reformulated version of the grammar produces the same probability distribution over strings, and so, predictions will be the same. The generalized expressions for the inside-outside and posterior probabilities used in this paper are given in this double-emission form.

**Folding kinetics:**

**Iterative helix formation**

The kinetics of RNA folding have been studied by [39], who determined the speed at which helices
form. They showed that helices form quickly from a local basepair, in the sense that, once the first basepair of a helix has formed, nearby bases are more likely to pair.

This motivates emulating the kinetics of RNA folding in a simplified way by forming helices iteratively. Iterative helix formation that has also been used by [80]. Once a suitable candidate basepair has been identified, a helix containing that basepair is formed.

Local helix formation:
iterative MEA estimation

We postulate that the first helix to form is the helix (without bulges) containing the basepair

\[
(i_{\text{max}}, j_{\text{max}}) = \arg \max_{(i,j)} \left\{ \hat{P}_{\text{paired}}(i,j) \right\}, \tag{11.1}
\]

where we use hats to denote the posterior pairing and unpairing probabilities obtained from the grammar. From a technical point of view, taking maximal probabilities in this way can be considered as a greedy approximation of the CYK algorithm [46].

In the framework of iterative helix formation, the statistic corresponding to MEA estimation is the expected difference in the number of correctly predicted basepairs after pairing bases \(i\) and \(j\),

\[
\Delta(i, j) = \hat{P}_{\text{paired}}(i, j) - \frac{1}{2} \left( \hat{P}_{\text{unpaired}}(i) + \hat{P}_{\text{unpaired}}(j) \right), \tag{11.2}
\]

Just as the difference in Eq. (11.2) is naturally interpreted, by analogy with thermodynamic models, as a measure of the energy and therefore, of the stability of a basepair, the corresponding basepairing probabilities can be considered as a measure of the time it takes for that basepair to form. If we approximate basepair formation as a continuous Markov process with rate equal to the posterior pairing probability, the time until helix formation has the exponential distribution with mean equal to the inverse of the posterior pairing probability. With this interpretation in mind, Eq. (11.1) expresses the pairing of bases in the physical order.

At each iteration a new helix containing \((i_{\text{max}}, j_{\text{max}})\) and such that, for each basepair \((i, j)\) in the helix, \(\Delta(i, j) > 0\), is determined conditional on previously formed helices. By folding helices in one go, the fact that helices form quickly is taken into account. More helices are formed until \(\Delta(i_{\text{max}}, j_{\text{max}}) < \delta\), for some threshold \(\delta > 0\).

MEA estimation hinges on the assumption that the posterior base-pairing probabilities given by the grammatical model are equal to the probability that a given basepair is correct. In fact, small positive values of the difference in Eq. (11.2) are not reliable, so requiring \(\delta > 0\) might be expected to increase the positive predictive value of the algorithm. From a more physical point of view, to additionally require \(\Delta(i, j) > \delta > 0\) for the basepair \((i_{\text{max}}, j_{\text{max}})\) is to incorporate the physics that once the first basepair is formed, nearby bases are more likely to pair. This local basepair needs to be “strong” enough for its dissociation time to be long enough for other basepairs to form. This also addresses the issue of the geometric and therefore unphysical distribution of helix lengths in the grammar [107].

A remark on the evolutionary model

For alignments of sequences or subsequences with high primary sequence conservation, the evolutionary might miss “obvious” helices, since it introduces extra uncertainty. This is especially relevant in iterative helix formation, because we only ever try to pair bases that have high posterior pairing probabilities. For this reason, rather than using the evolutionary model as in [108], we use a mixed model: at the very start, we form basepairs with very high posterior pairing probabilities using the simpler heuristic of just multiplying the base-pairing probabilities for each sequence to obtain the prior base-pairing probabilities, and then switch to the full evolutionary model. The ability to mix different methods is a strength of the iterative approach.

Bayesian weighting:
the distance function

In the model we have built up to this point, distinct columns and pairs of columns are, insofar as prior unpairing and pairing probabilities are concerned, assumed to be independent. This does not reflect biological fact, as interlacing structures prevent each other from forming and a column cannot pair with two different columns at the same time.

We note that this kind of crossing interaction may lead to pseudoknot formation. Standard SCFG approaches are unable to predict pseudoknots [23]: in standard MEA estimation, one hopes that the model predicts the more stable of
the two interlacing structures. In the iterative framework, the extra information is used to address, more generally, the most obvious drawback of the iterative approach: once a helix that is incompatible with the correct structure has been formed, the final prediction is likely to be poor. From a kinetic perspective, base pairs frequently blocked by other, transient, base pairs, would take longer to form, as the underlying continuous Markov process is only enabled during intervals where the base pair is not in conflict with other base pairs.

To include these physical dependencies between columns in the model, we look to penalize the pairing of two columns if there exist columns between that are likely to form a basepair incompatible with these two columns. Since standard SCFG approaches cannot model pseudoknots, here, “incompatible” does not only refer to basepairs that share a position with these two columns, but also to basepairs that would form a pseudoknot with these two columns. Thus we discount the prior base-pairing probabilities by an exponential factor based on a distance function, so that they take the form

\[ P_{\text{paired}}(s_i, s_j) = \exp\left(- \frac{d(i, j)}{K |j - i|} \varpi(s_i, s_j)\right), \]

where \( s_i \) denotes the \( i \)-th column of the alignment, and where \( d(i, j) \) is a distance function to be specified, \( K \) is a weighting parameter, and \( \varpi(s_i, s_j) \) are the usual base-pairing probabilities, derived from an evolutionary model or the simple heuristic mentioned previously.

We choose a distance function such that two columns are “far away” from each other if there are columns between them that are likely to form a basepair incompatible with these two columns. Each intermediate column \( k \), with \( i < k < j \), is given a weight equal to the probability of that column forming a basepair incompatible with \( (k-1, j) \). Figure 11.1 shows an example of the distance function on a partially folded structure. Formally, we define, for \( i \leq j \),

\[
d(i, j) = \begin{cases} 
0 & \text{if } i = j, \\
d(\tau, j) & \text{if } i, \tau \text{ pair and } i \leq \tau \leq j, \\
\beta(i, j) + d(i + 1, j) & \text{otherwise.}
\end{cases}
\]

(11.3)

For example, in Fig. 11.1, we calculate the distance between \( i \) and \( j \), following Eq. (11.3). We have

\[
d(i, j) = \beta(i, j) + d(i + 1, j),
\]

\[
d(i + 1, j) = \beta(i + 1, j) + d(i + 2, j),
\]

and so on. Further, \( d(k, j) = d(k, j) \), as \( k \) and \( k \) are paired. Continuing,

\[
d(k + 1, j) = \beta(k, j) + d(k + 1, j),
\]

and so on; finally, \( d(j, j) = 0 \).

Here, \( \beta(i, j) \) is the probability that column \( i + 1 \) forms a basepair that is incompatible with \( (i, j) \), so that, the events in question being disjoint,

\[
\beta(i, j) = 1 - \hat{P}_{\text{unpaired}}(i + 1) - \sum_{k=i+2}^{j-1} \hat{P}_{\text{paired}}(i + 1, k).
\]

Thus the posterior pairing probabilities are used to guide the folding of the next iteration (and
11. Oxfold

\[
\begin{align*}
\text{calculate } & \Rightarrow (s_i, s_j) \text{ (prior pairing probabilities)}, \\
& \Rightarrow (s_i) \text{ (prior unpairing probabilities)} \\
\text{loop} & \\
\text{find possible basepairs (conditional on existing structure)} \\
\text{calculate distances } & \mathcal{d}(i, j) \\
\text{calculate inside-outside probabilities } & I(U, i, j), O(U, i, j) \\
\text{calculate posterior probabilities } & \mathcal{P}_{\text{paired}}(i, j), \mathcal{P}_{\text{unpaired}}(i) \\
(l_{\max}, j_{\max}) & \leftarrow \arg \max_{l, j} \{\mathcal{P}_{\text{paired}}(i, j)\} \\
\text{if } \Delta(l_{\max}, j_{\max}) > \delta \text{ then} \\
\text{find maximal helix containing } (l_{\max}, j_{\max}) \\
\text{update existing structure} \\
\text{else} & \\
\text{break} \\
\text{end if} & \\
\end{align*}
\]

Inside-Outside Probabilities:

\[
I(U, i, j) = \begin{cases} 
\sum_{l \neq j} \sum_{v \in V} [P(U \rightarrow V)I(V, i, k)] & \text{if } i = j; \\
+ \sum_{l \neq j} [P(U \rightarrow V)(i, j + 1)] & \text{if } i < j; \\
0 & \text{otherwise.}
\end{cases}
\]

\[
O(U, i, j) = \begin{cases} 
\sum_{v \in V} \sum_{k \neq 0} [P(U \rightarrow W)O(V, i, k)] & \text{if } U = S, i = 1, j = \ell; \\
+ \sum_{v \in V} [P(U \rightarrow V)(i, j + 1)] & \text{if } i < j; \\
1 & \text{otherwise.}
\end{cases}
\]

Posterior pairing and unpairing probabilities:

\[
\begin{align*}
\mathcal{P}_{\text{paired}}(i, j) & = \frac{1}{I(U, i, j)} \exp \left( -\frac{d(i, j)}{K[j - i]} \right) \mathcal{P}(s_i, s_j) \\
& \times \sum_{l = 1} \left[ O(U, i, j)I(V, i + 1, j - 1)P(U \rightarrow V) \right] \\
\mathcal{P}_{\text{unpaired}}(i) & = \frac{1}{I(U, i, j)} \sum_{l = 1} \left[ O(U, i, j)P(U \rightarrow V) \right]
\end{align*}
\]

Figure 11.2: Simplified pseudocode summarizing the full kinetic folding algorithm. The inside-outside and posterior probabilities are written in the double emission form of [6], and include the distance function and structural constraints. The modifications needed to take account of structural constraints and to introduce the distance function are shown in gray. Capital letters denote generic non-terminals, while lower-case letters denote column indices and \( s_i \) denotes the \( i \)-th column of the alignment; \( \ell \) denotes the total sequence length. See text for detailed explanation.

11.3 Discussion

The full kinetic model for RNA secondary prediction, Oxfold, was benchmarked against PPfold [193], a parallelized implementation of the Pfold model of [107, 108]. We also evaluate the performance of RNAalifold [15, 87], a thermodynamic model without evolutionary information.

Benchmarking data and parameters

For benchmarking purposes, we have created a curated RNA dataset based on the Rfam database.
11.3. Discussion

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>F-Score</th>
<th>Sensitivity</th>
<th>PPV</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNAalifold</td>
<td>0.704</td>
<td>0.748</td>
<td>0.689</td>
</tr>
<tr>
<td>PPfold</td>
<td>0.673</td>
<td>0.650</td>
<td>0.728</td>
</tr>
<tr>
<td>Iterative without evolutionary model, without distance function</td>
<td>0.684</td>
<td>0.698</td>
<td>0.694</td>
</tr>
<tr>
<td>Iterative without evolutionary model, with distance function</td>
<td>0.688</td>
<td>0.696</td>
<td>0.703</td>
</tr>
<tr>
<td>Iterative with evolutionary model, without distance function</td>
<td>0.698</td>
<td>0.666</td>
<td>0.780</td>
</tr>
<tr>
<td>Oxfold (full kinetic model)</td>
<td>0.723</td>
<td>0.688</td>
<td>0.800</td>
</tr>
</tbody>
</table>

Table 11.1: Comparison of the performance of different algorithms on the test data set of 41 alignments. The algorithms presented in this paper are compared to PPfold [193] and RNAalifold [15, 87]. See Methods section for details. The values shown are the averages of the F-score, sensitivity and PPV of the alignments in the test data set. The values shown in bold type are the maximum values in the respective column.

Alignments of homologous RNA sequences with their consensus secondary structure were extracted from among those Rfam seed alignments that bear the “published” tag. From these, 50 alignments with at least 50 sequences were randomly selected. We note that random data selection ensures reliability of results. In a pre-filtering step we discarded outlier sequences with many/long insertions and deletions from each family, using a similar approach to that of PPfold [193], which does not consider columns with more than 75% of gaps for pairing. Indels were first determined relative to a family consensus sequence, then a total mismatch score was calculated based on indel lengths, and sequences that had significantly larger mismatch score than the family mean were deleted. Further random selection was performed to reduce these to alignments of five sequences each; the results of [107, 108], suggest that this suffices to take into account the evolutionary information. Because of the computational complexity of the model (which we discuss further below), we restricted to 41 alignments of length up to 214, with an average length of 105.

The consensus secondary structures given in Rfam may be slightly different from the secondary structures that the individual sequences fold into. In this sense, we cannot say that our secondary structures are experimentally verified, but the approach of comparing predictions to these secondary structures is commonly used in analysis of comparative prediction methods. As with all benchmarks of this nature [15, 107], this should be taken into account.

It is known that grammar performance depends on data sets [178]. Consequently, it is important to monitor data set dependence, in particular to avoid over-fitting. For these reasons, the grammar parameters and evolutionary trees used for benchmarking purposes were those of PPfold. In particular, the grammar underlying our present approach is essentially the simple grammar for which [178] did not find evidence of over-fitting. The other parameters were chosen heuristically: the parameter $\delta$ was set to 0.5 to make the first, local basepair stable enough for its dissociation time to be long enough for other basepairs to form. This is a trade-off between losing sensitivity at high values of $\delta$ and losing PPV at low values of $\delta$. Similarly, $K$ determines the amount of penalization by the distance function; setting $K = 0.5$ leads to a maximum penalization of about one order of magnitude.

**Benchmarking statistics**

We assess the performance of these models on a single alignment by calculating the sensitivity, positive predictive value (PPV), defined by

$$
sensitivity = \frac{TP}{TP + FN},
$$

$$
PPV = \frac{TP}{TP + FP},
$$

where TP, FP, and FN denote the number of true positives (number of correctly predicted basepairs), false positives (wrong basepairs predicted) and false negatives (true basepairs not predicted), respectively. We also determine the F-score, which is the harmonic mean of sensitivity and PPV:

$$
F-score = \frac{2TP}{2TP + FN + FP}.
$$

The averaged values of the F-scores, sensitivities and PPVs of the structures predicted by the kinetic models are compared, in Table 11.1, to those of PPfold and RNAalifold. The predictions of
Figure 11.3: Comparison of the F-score, sensitivity and PPV of the consensus structures predicted by Oxfold (the full kinetic model presented in this paper) and those of PPfold [193] and RNAalifold [15, 87] respectively, for the sequences in the test data set of 41 alignments. The outliers marked A and B are discussed in the text.

Figure 11.4: Reference structure and predictions by PPfold and Oxfold, for the outlier sequence marked A. The shaded basepairs are paired by PPfold. After predicting the first two helices, Oxfold chooses that same helix as its next candidate helix, but then exits because the δ cutoff is not met.

Figure 11.5: Reference structure and predictions by PPfold, the iterative method with the evolutionary model, but without the distance function, and Oxfold, for the outlier sequence marked B. The value of the Δ difference defined in Eq. (11.1) is about 0.822 for the shaded incompatible helix predicted without the distance function; the value for the correct helix predicted by the full Oxfold algorithm is 0.997.
the full kinetic model presented in this paper are compared to those of PPfold and RNAalifold in Fig. 11.3.

Discussion of results

We note that Oxfold performs better, on average, than PPfold in terms of averaged sensitivity, PPV and F-score. Additionally, it has a higher F-score and PPV than RNAalifold, though the thermodynamic model has a higher sensitivity. In particular, we observe that Oxfold has a noticeably higher PPV than PPfold and RNAalifold.

We also note that including an evolutionary model decreases the sensitivity of the algorithm, but greatly increases the PPV. Moreover, the distance function does not seem to be of much use without an evolutionary model. Both of these observations are compatible with the conclusion that the posterior probabilities without the evolutionary model are less reliable (in the sense that they somehow correlate with correctness) than those obtained with an evolutionary model. The fact that the iterative model without distance function, both with the mixed and the unmixed evolutionary model, have essentially the same PPV, gives further weight to this conclusion.

Also, the model with evolutionary information, but without distance function, performs mildly better than PPfold. This is notable, because this method is the iterative method applied to the standard PPfold model. By conditioning on known structure, the grammar model no longer has probability mass contributed from incompatible structures, which one might hope would lead to a better structure prediction.

Since PPfold and Oxfold share the same grammatical framework, outliers in Fig. 11.3 can be attributed to the iterative method developed in this paper. We discuss two such outliers, marked A and B in Fig. 11.3. Figures 11.4 and 11.5 show the corresponding predictions. In outlier A, the prediction cutoff δ is not met by a basepair predicted by PPfold, upon which Oxfold terminates prediction (consequently, the PPVs are very similar, but the sensitivity of PPfold is higher). By contrast, outlier B is an example of an alignment where Oxfold performs noticeably better than PPfold. Without the distance function, Oxfold has zero F-score on this alignment; with this distance function, the F-score rises to 0.5, illustrating how the distance function facilitates the prediction of correct basepairs.

With the interpretation of the difference in Eq. (11.2) as a measure of the stability of a basepair, and of the posterior pairing probabilities as a measure of the inverse time it takes to form a basepair (discussed in the Methods section), the fact that this model works indicates that RNA folds a stable scaffold before less stable substructures with short dissociation times start to appear and disappear (rather than folding its functional secondary structure while such substructures appear and disappear).

11.4 Conclusions

In this paper, we have incorporated kinetic effects into a grammatical model for RNA secondary structure prediction by iterative formation of helices and by taking into account some relationships between columns of an alignment by means of a distance function. Conceptually, introducing a distance function is the analogue, at the level of the emission probabilities of the grammar, of including (albeit possibly different) information about the relationships between columns in the alignment by making the production rules of the grammar more complex. The performance of the kinetic model suggests that the dynamical aspects of RNA folding should not be disregarded in SCFG approaches to RNA secondary structure prediction.

Incorporating co-transcriptional effects into the model might therefore be a possible next step: Kramer and Mills [112], have shown that RNA folds as it is being transcribed, usually in the 5' to 3' direction. Thus the 5' end of the RNA molecule is allowed to fold before the sequence has been entirely transcribed, resulting in intermediate structures that do not necessarily exist in the final, functional structure, since the speed of stem formation greatly exceeds the speed of transcription [75]. Moreover, [144], have demonstrated “with statistical significance that co-transcriptional folding strongly influences RNA sequences in two ways: (1) alternative helices that would compete with the formation of the functional structure during co-transcriptional folding are suppressed and (2) the formation of transient structures which may serve as guidelines for the co-transcriptional folding pathway is encouraged”. [75], for instance, have incorporated co-transcriptional effects into a thermodynamic model by using a genetic algorithm.

Nevertheless, the fundamental problems af-
fecting SCFG algorithms listed by [107], still remain very much topical. Here, we discuss three issues of particular relevance to our algorithms:

**Pseudoknots**

As mentioned previously, standard SCFG approaches cannot predict pseudoknots [23]. Leaving the framework of stochastic context-free grammars, but still using a formal grammar, it is possible to predict pseudoknotted structures [176]. The iterative method presented in this paper could be adapted to predict pseudoknots either by adapting the definition of permissible basepairs or by using the methods described by [176]. Similarly, we would expect biophysical folding mechanisms to be conserved in pseudoknotted structures as in non-pseudoknotted structures.

**Computational complexity**

Standard SCFG algorithms for RNA secondary structure prediction have computational complexity $O(\ell^3)$, where $\ell$ is the sequence length [107]. Hence the kinetic model presented in this paper has complexity $O(\ell^4)$, and a co-transcriptional algorithm along the lines of the algorithm of [75] (i.e. folding longer and longer subsequences, starting at the 5’ end) would have a complexity $O(\ell^5)$, which makes the algorithms even more expensive than standard prediction approaches. Whereas it is straightforward, even intrinsic, to reuse computations for shorter subsequences in current methods for thermodynamic models, this may seem much more complex in a kinetic model, as the optimum pathway may completely change upon the elongation of a subsequence. However, when the aim is to include co-transcriptional effects, it is not unreasonable to assume that relevant models can be formulated allowing algorithms with complexity lower than $O(\ell^5)$. This matters especially for co-transcriptional folding, since one expects transcriptional effects to be stronger for longer sequences.

**Non-canonical basepairs**

The existence of non-canonical basepairs is a possible complication in RNA secondary prediction, for long, correct helices with non-canonical basepairs may appear less attractive than short, yet spurious helices. This observation ties the non-canonical basepairs issue somewhat to the geometric, and therefore unphysical, distribution of the helix lengths in the grammar [107]. Here, we have addressed that issue by making the pairing of the first basepair of a helix more expensive than that of later pairs (as discussed in the Methods section). The effect of more complex distributions of helix lengths has previously been studied by [44] and [178] who considered more complex grammars, allowing for stacking non-terminals in the grammar.

Nonetheless, RNAalifold [15, 87] does not allow non-canonical basepairs in its default settings, whereas PPfold [193] associates very low pairing probabilities with non-canonical basepairs. Some gain in sensitivity might therefore be possible by allowing some non-canonical basepairs in pairs of alignment columns at low probability cost, but more insight into the role of non-canonical basepairs (and a corresponding model) may well be required.

**Acknowledgments and funding**

This work was carried out as part of the Oxford Summer School in Computational Biology, 2012, in conjunction with the Department of Plant Sciences and the Department of Zoology. We would like to thank the Oxford Supercomputing Centre for computational resources and support, and the EU grant COGANGS.

This work was supported by the Engineering and Physical Sciences Research Council (JWJA), the Biotechnology and Biological Sciences Research Council (PAH).
Bibliography


Applications and Many Core Architecture, pages 23–35, 2008. 124


158


[96] P. A. Jenkins, P. Fearnhead, and Y. S. Song. Tractable stochastic models of evolution for loosely


[136] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947. 46
[141] MATLAB. *version 7.10.0 (R2010a)*. Natick, Massachusetts, 2010. 104


163
Bibliography


[185] J. Sirén. Statistical models for inferring the structure and history of populations from genetic data. PhD thesis, University of Helsinki, Faculty of Science, Department of Mathematics and Statistics, 2012. 21, 72, 78


164


