Improvement of the jpHMM approach to recombination detection in viral genomes and its application to HIV and HBV

Dissertation
zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultäten
der Georg-August-Universität zu Göttingen

vorgelegt von

Anne-Kathrin Schultz

aus Göttingen

Göttingen 2011

D 7

Referent:	Prof. Dr. Burkhard Morgenstern
Korreferent:	Prof. Dr. Stephan Waack
Tag der mündlichen Prüfung:	27. April 2011

Acknowledgements

At this point, I would like to thank all the people who have supported and accompanied me throughout the last years. I am very grateful to Professor Burkhard Morgenstern for giving me the opportunity to carry out this project in his department. Besides many fruitful discussions, he supported the progress of this thesis by encouraging numerous collaborations throughout the world. This, and the opportunity to participate in many international conferences let me see many great places like Taiwan, Lisbon, Oxford, the USA and Canada.

Many thanks go to the second referee of this thesis, Professor Stephan Waack, for many helpful discussions and "committee meetings".

My special thanks belong to Professor Mario Stanke for supervising this thesis during his post-doc time in Göttingen. Even after having moved to Greifswald he has always been open to suggestion. With his expertise as well as many stimulating discussions and ideas he substantially contributed to the success of this project.

A big thank you goes to all present and former members of the biological and the medical bioinformatics departments. In particular, I would like to thank Ingo Bulla for supporting this project with many helpful suggestions and his C++ expertise. Special thanks go to Isabelle Heinemeyer for numerous cheerful discussions while sharing the office with me. Furthermore, I thank Rasmus Steinkamp, Maike Tech and Torsten Schöps for their helpfulness in cases of technical difficulties, and Katharina Hoff and Peter Meinicke for sharing their biological as well as methodological knowledge. Many thanks go to Thomas Lingner for his encouragement while writing this thesis.

To Ming Zhang, Thomas Leitner and Bette Korber from Los Alamos National Laboratory, USA, Professor Dan Brown from the University of Waterloo, Canada, and Professor Paul Dény from INSERM in Lyon, France, I am very grateful for long-standing, close collaborations which resulted in a lot of interesting and successful projects.

Many thanks to all my friends and my family for being part of my life during the last years. Special thanks go to Annette Illgen and Tilman Sauer, to Karl Pietscher and to Yannick, Julia, Susanne and Ludwig Schultz, for their incomparable and inexhaustible support and patience. I am also grateful to Tilman and Julia for proofreading this thesis.

Abstract

Accurate virus genotyping and the detection of recombinant strains are of crucial importance for understanding viral evolution as well as the design of potential vaccines and treatment strategies. A very accurate tool for detecting recombinations in genomic HIV-1 sequences is jpHMM (jumping profile Hidden Markov Model). For a given sequence, it predicts recombination breakpoints and assigns a parental subtype to each segment in between two breakpoints. In this thesis, modifications and extensions of jpHMM are carried out to improve the reliability of the recombination prediction, to reduce the runtime of the program and to allow the analysis of recombinations in circular genomes.

As incorrect subtype assignments or recombination predictions may lead to wrong conclusions in epidemiological or vaccine research, it is important to assess the reliability of the predicted recombination in a particular sequence. For this reason, the output of jpHMM is extended to include a tagging of regions where the model is uncertain about the predicted subtype and an interval estimate for each predicted breakpoint. It is shown that this extension strongly improves the reliability of the recombination prediction.

To allow an efficient application of jpHMM to large data sets or species with a large number of subtypes, the complex architecture of the model is substantially modified. Evaluation on HIV-1 as well as hepatitis B virus (HBV) data shows that these modifications lead to a considerable reduction of the runtime of the program.

Furthermore, an extension of jpHMM to detect recombinations in viruses with circular genomes such as HBV is introduced. Recombination analysis in circular genomes is usually done on artificially linearized sequences using linear models. Since these models are normally unable to model dependencies between nucleotides at the 5’ and 3’ end of a sequence, this can result in inaccurate predictions of breakpoints and thus in incorrect classifications of circular genomes. In contrast, the circular jpHMM takes into account the circularity of the genome. Its accuracy is evaluated on a large set of recombinant HBV sequences. Additionally, about 3000 full-length HBV sequences are studied to detect so-called *circulating recombinant forms* (CRF). For this, certain criteria for classifying recombinant HBV sequences are proposed. Based on these criteria, 17 CRFs can be identified.

Contents

Abbreviations **V**

List of Figures **VIII**

List of Tables **IX**

1 Introduction **1**

2 Background **9**

- 2.1 Viruses . . . 9
  - 2.1.1 HIV . . . 9
  - 2.1.2 HBV . . . 12
- 2.2 Methods for recombination detection . . . 17
  - 2.2.1 Overview . . . 17
  - 2.2.2 jpHMM . . . 18

3 Improvements, extensions and modifications of jpHMM **27**

- 3.1 Uncertainty regions and breakpoint intervals . . . 27
  - 3.1.1 Forward and backward variables . . . 28
  - 3.1.2 Posterior probabilities . . . 30
  - 3.1.3 Uncertainty regions and breakpoint intervals . . . 31
- 3.2 Restriction of the search space of the Viterbi algorithm . . . 35
  - 3.2.1 Location of the input sequence relative to the reference genome . . . 35
  - 3.2.2 Definition of active states in the Viterbi algorithm . . . 36
- 3.3 A modified jpHMM architecture with mute jump states . . . 47
  - 3.3.1 Number of jumps in a (simple) jpHMM . . . 47
  - 3.3.2 A modified (simple) jpHMM with a reduced number of jumps . . . 48
  - 3.3.3 A modified jpHMM including all state types . . . 49
  - 3.3.4 Memory required for storing the jumps in a jpHMM . . . 51

3.3.5 Complexity of the Viterbi algorithm . . . 52
3.3.6 Jumps in case of non-consensus columns . . . 53
3.4 A jpHMM for recombination prediction in circular genomes . . . 57
3.4.1 Extension of query sequence and multiple sequence alignment . . . 59
3.4.2 Pre-alignment of extended query to extended alignment with BLAT 61
3.4.3 Recombination prediction with jpHMM based on pre-alignment . . 61
3.4.4 Uncertainty regions and breakpoint intervals in circular genomes . . 62
3.4.5 Workflow for the recombination prediction in circular genomes . . 63

4 Implementation 65
4.1 Revision of the source code . . . 65
4.2 Program workflow . . . 66
4.3 Input . . . 67
4.4 Output . . . 68
4.5 Runtime . . . 71
4.6 Availability . . . 72
4.6.1 Webserver . . . 72

5 Results and discussion 75
5.1 Data . . . 75
5.1.1 HIV . . . 75
5.1.2 HBV . . . 77
5.2 jpHMM parameter estimation . . . 81
5.2.1 HIV-1 . . . 81
5.2.2 HBV . . . 81
5.3 Uncertainty regions and breakpoint intervals . . . 87
5.3.1 Accuracy of predicted recombination patterns . . . 87
5.3.2 Sensitivity of predicted breakpoint intervals . . . 88
5.3.3 Accuracy of predicted subtypes . . . 90
5.3.4 Comparison to a naïve approach . . . 90
5.3.5 Default posterior probability threshold for breakpoint intervals . . . 92
5.3.6 Discussion . . . 93
5.4 Restriction of the search space of the Viterbi algorithm . . . 95
5.4.1 Parameters . . . 95
5.4.2 Accuracy of pre-defined active alignment column intervals . . . 96
5.4.3 Accuracy of jpHMM on the basis of pre-defined active alignment column intervals . . . 98

5.4.4 Discussion . . . 98
5.5 A modified jpHMM architecture with mute jump states . . . 101
5.5.1 Accuracy of the modified jpHMM with mute jump states . . . 101
5.5.2 Accuracy of the modified jpHMM with mute jump states in combination with a pre-alignment with BLAT . . . 103
5.5.3 Discussion . . . 103
5.6 Runtime of different jpHMM versions . . . 105
5.6.1 Comparison of the runtime of different jpHMM versions . . . 105
5.6.2 Runtime for datasets with a large number of subtypes . . . 106
5.6.3 Discussion . . . 108
5.7 A jpHMM for circular genomes and its application to HBV . . . 109
5.7.1 Settings . . . 110
5.7.2 Accuracy . . . 111
5.7.3 Application to real-world HBV genomic sequences . . . 112
5.7.4 Proposal for a classification system for recombinant forms of HBV 115
5.7.5 Comparison to other studies . . . 123
5.7.6 Discussion . . . 124

6 Conclusions 127

List of Publications 131

References 133

Abbreviations

AIDS	Aquired Immuno Deficiency Syndrome
BLAST	Basic Local Alignment Search Tool
BLAT	BLAST-like alignment tool
BPI	breakpoint interval
cccDNA	covalently closed circular DNA
CRF	circulating recombinant form
DNA	deoxyribonucleic acid
HBV	hepatitis B virus
HCV	hepatitis C virus
HDV	hepatitis delta virus
HIV	human immunodeficiency virus
HIV-1	HIV type 1
HIV-2	HIV type 2
HMM	hidden Markov model
HPV	human papillomavirus
HXB2	HIV reference genome, GenBank accession number K03455
JALI	jumping alignment algorithm
jpHMM	jumping profile HMM
LANL	Los Alamos National Laboratory
mRNA	messenger RNA
nt	nucleotide
PCR	polymerase chain reaction
pgRNA	pregenomic RNA
qp	query sequence position
rRNA	ribosomal RNA
RNA	ribonucleic acid
SGE	Sun Grid Engine
SIV	simian immunodeficiency virus

UR	uncertainty region
URF	unique recombinant form

List of Figures

2.1 Geographic distribution and prevalence of HIV subtypes and CRFs 12
2.2 Geographic distribution of HBV genotypes 15
2.3 Simplified topology of a jpHMM 19
2.4 Toy example of a jpHMM 20

3.1 Examples for uncertainty regions 31
3.2 Examples for breakpoint intervals 32
3.3 Definition of active columns in the pre-alignment with BLAT 41
3.4 Example for non-overlapping BLAT and beam-search intervals 44
3.5 Jumps between match states in the original jpHMM 48
3.6 Jumps in a jpHMM with a modified architecture I 49
3.7 Jumps in a jpHMM with a modified architecture II 51
3.8 Jumps in the original jpHMM in case of gaps 54
3.9 Jumps in a jpHMM with a modified architecture in case of gaps I 55
3.10 Jumps in a jpHMM with a modified architecture in case of gaps II 56
3.11 Location of breakpoints in linearized sequences of circular genomes 58
3.12 Extension of full-length query sequences for circular genomes 59
3.13 Alignment of (extended) query sequences to an extended multiple alignment for circular genomes 60

4.1 Posterior probabilities plotted for an example HIV-1 sequence 69
4.2 Output of the jpHMM recombination prediction in circular form 71
4.3 Webserver output for HIV including UR and BPI 73

5.1 Example recombinants with overlapping breakpoint intervals I 116
5.2 Example recombinants with overlapping breakpoint intervals II 118
5.3 Comparison of segments in between two breakpoint intervals 119
5.4 Recombination prediction for a circulating recombinant form of HBV . . . 122
5.5 Recombination prediction for a unique recombinant form of HBV 123

List of Tables

3.1 Comparison of the number of jumps per column in different jpHMM versions 49
3.2 Comparison of the required memory of different jpHMM versions 52
3.3 Comparison of the number of iterations the Viterbi algorithm 53

5.1 Comparison of the accuracy of predicted BPI and BPI of fixed length 89
5.2 Accuracy of the predicted subtypes at each query sequence position 91
5.3 Comparison of the accuracy of subtypes predicted with a jpHMM incl. UR and BPI and the original jpHMM 92
5.4 Comparison of the accuracy of predicted BPI and BPI of fixed length for the default posterior probability threshold 92
5.5 Accuracy of active alignment column intervals 97
5.6 Comparison of the runtime of different jpHMM versions 106
5.7 Comparison of the jpHMM runtime for datasets with a large number of subtypes 107
5.8 Accuracy of recombination prediction for artificial HBV recombinants with breakpoints at random positions 112
5.9 Accuracy of recombination prediction for artificial HBV recombinants with fixed breakpoint positions 113
5.10 Frequency of pure genotypes of HBV predicted with jpHMM 114
5.11 Recombination patterns for CRFs of HBV predicted with jpHMM 121

Chapter 1

Introduction

Many of the severest and most infectious diseases are caused by viruses. A well-known example is the Aquired Immuno Deficiency Syndrome (AIDS) that is caused by the human immunodeficiency virus (HIV). It is estimated that the global AIDS pandemic has claimed 25 million deaths world-wide [100] since AIDS has been reported for the first time in 1981 [13]. The estimated number of people living with HIV in 2008 was 33.4 million, among whom about 2.7 million have become infected in 2008. Up to now, no effective vaccine against HIV has been found. The development of antiretroviral drugs to slow down the course of the disease has lead to a decline of the morbidity and mortality of HIV infections [70] but the number of people dying because of AIDS-related illnesses is still very high (about two million in 2008 [100]).

One of the major obstacles for the design of antiviral drugs and vaccines is the high genetic variability of viruses [38]. Besides mutation, viral evolution is driven by recombination of viruses from different clades or subtypes. Recombination can occur as a consequence of a superinfection with at least two different subtypes and lead to so-called *inter-subtype* recombinants [24].

Recombination in HIV and HBV

Two types of HIV, named HIV-1 and HIV-2, are known to exist. HIV-1 is classified into several groups that arose due to independent cross-species transmissions from chimpanzees and gorillas (section 2.1). One of these groups, called group M, is mainly responsible for the global HIV pandemic. It is divided into nine genetically distinct subtypes, A-D, F-H, J and K, whereas some of them have been further subdivided into sub-subtypes [80]. Recombination among these subtypes is very common [32]. A recombinant form that has been identified in at least three epidemiological unlinked individuals is called a *circulating* *recombinant form* (CRF). Up to now, 48 CRFs have been identified [49] and the number is increasing rapidly. It is estimated that about 18 % of all infections worldwide are associated with CRFs [30]. In addition to CRFs, a large number of recombinant forms identified in only one individual exists. These recombinant forms are called *unique recombinant forms* (URF).

Besides HIV, we study recombinations in genomic sequences of the hepatitis B virus (HBV). Chronic hepatitis B infection can lead to serious illness, such as liver cirrhosis and hepatocellular carcinoma, as well as death. It is estimated that more than two billion people worldwide have been infected with HBV [102], among whom about 360 million are chronically infected. For HBV, eight different genotypes, A-H, have been classified (section 2.1). For most of them, several subgenotypes have been defined. Recently, two further putative genotypes, tentatively named I and J, have been identified, which are still subject of debate. Also in HBV, recombination among genotypes is very common. In contrast to HIV-1, recombinants are incorporated in the classification system of genotypes and subgenotypes on the basis of sequence similarity [17, 96, 1]. Thus, recombinant forms are not distinguished from pure (sub)genotypes which confounds the exact definition of pure genotypes.

Importance of accurate genotyping and recombination detection

HIV subtypes and HBV genotypes (Figs. 2.1 and 2.2, section 2.1) are unevenly distributed throughout the world [58, 48]. For example, HIV-1 subtype C accounts for about 50% of HIV infections worldwide [30]. In some geographic regions even recombinant forms, such as CRF02_AG, a recombinant of subtypes A and G, have become the most prevalent strains. Possible reasons for this uneven distribution may lie in the "fitness" of viral strains, such as, for example, replicative fitness, transmissibility, or fitness in terms of disease progression and sensitivity to antiretroviral therapy [99, 65, 76]. How subtypes or recombination exactly influence such biological features has, however, not yet been understood.

Also in HBV, the understanding of the role of genotypes and recombination in the outcome of a HBV infection is limited. Some genotypes, for example, appear to be associated with a higher risk of developing hepatocelluar carcinoma or a higher severity of chronic HBV than others [52, 59]. In Germany, HBV genotyping is recommended before starting a certain antiviral therapy [16].

Thus, accurate genotyping and the detection of recombinant strains are of crucial importance for getting insights into molecular epidemiology and viral evolution as well as for understanding the influence of genetic diversity on clinical outcomes.

Recombination detection in viral sequences with jpHMM

A large variety of programs for recombination analysis in viruses has been developed during the last years [78] (section 2.2). One of the most accurate methods in detecting recombination breakpoints in HIV-1 sequences is jpHMM [87, 109]. jpHMM (*jumping profile Hidden Markov Model*) is a probabilistic model that we developed during my diploma thesis [84] to compare single nucleotide sequences to a given multiple alignment of a sequence family. It was applied to detect recombinations in genomic sequences from HIV as well as hepatitis C virus (HCV).

A detailed description of the jpHMM approach can be found in section 2.2. According to this, given a partition of the multiple sequence alignment into subclasses, called *subtypes*, each subtype is modeled as a profile Hidden Markov Model (profile HMM) [43, 21]. In addition to the usual transitions *within* a profile HMM, transitions, called *jumps*, *between* the profile HMMs for different subtypes are allowed (for more details see section 2.2 and Figs. 2.3 and 2.4). To these jumps, a *jump probability* is assigned. As for other HMMs, the alignment of a query sequence to the given sequence family is then defined by the most probable path through the model generating the sequence, allowing jumps between the different subtypes. This alignment is called the *jumping alignment* of the query sequence to the given alignment. The most probable path is called the *Viterbi path* [101] and is determined with the well-known *Viterbi algorithm*. Positions of jumps between different subtypes define recombination breakpoints.

jpHMM was evaluated on real virus genome sequences as well as simulated recombined genome sequences [87]. Comparing single representative HIV-1 sequences, the recombination breakpoints identified by jpHMM were found to be significantly more accurate than breakpoints defined by methods that are traditionally used.

Aims of this thesis

The aim of this thesis is to improve the jpHMM recombination prediction in terms of the reliability of the predicted subtypes and breakpoints. Furthermore, the program should be modified and extended to reduce the runtime and to make it applicable to other kinds of species. In the following, a broad outline of the concepts is given.

Reliability of the recombination prediction

As incorrect subtype assignments or recombination predictions may lead to wrong conclusions in epidemiological or vaccine research, it is always important to know how reliable the predicted parental subtypes and breakpoint positions in a particular sequence or a particular region of a sequence are. For example, a breakpoint that is located in a conserved genomic region cannot be determined precisely by jpHMM. The position of the predicted breakpoint within this conserved region is chosen somehow arbitrarily by the model. Additionally, it is possible that, in a certain region of the sequence, two different subtypes are equally likely to be the parental subtype. But, since only one of the subtypes can be predicted with the model, one of the potential parental subtypes remains undetected. For this reason, the output of jpHMM is extended to include a tagging of regions where the model is uncertain about the predicted parental subtype and an interval estimate for each predicted breakpoint in addition to predicting its precise position (section 3.1).

Reduction of the runtime of jpHMM

Currently, more than $2,300$ full-length and more than $330,000$ fragmental HIV-1 sequences are available in the Los Alamos HIV sequence database [49] and the number is increasing rapidly. These sequences are subject of systematic resubtyping to enable a better understanding of the dynamics of the global HIV-1 epidemic and the role of recombination. Leading HIV experts from the Los Alamos National Laboratory (LANL) already analyzed, in close collaboration with us, more than $9,400$ HIV-1 sequences from three epidemically important regions using jpHMM among other tools [108]. For 4.9% of these sequences, subtype assignments were different from the ones published in the original literature, demonstrating that a careful reclassification is necessary. Also for other viruses or species, the amount of data will increase rapidly due to advanced sequencing techniques.

The current average jpHMM runtime on a desktop PC for HIV-1 sequences is ≥ 10 minutes per sequence [87] being a consequence of the very large number of states in the underlying HMM. This is too time-consuming for such large-scale studies. Therefore, a method is developed that restricts the search space of the Viterbi algorithm and thus reduces the runtime and memory of jpHMM (section 3.2).

Application of jpHMM to other species

During the development of jpHMM, researchers from other fields approached us with the request to adapt jpHMM to other species. Two major concerns were the identification of chimeras in 16S rRNA and the recombination detection in circular genomes such as HBV.

Sequence families with many subtypes - modification of the model architecture

Comparative analysis of 16S rRNA is the ’gold-standard’ for the identification and phylogenetic classification of bacteria ([25], reviewed in [105]). But in case of studying mixed bacterial populations, the analysis of 16S rRNA can be hampered by chimeric artifacts of different species that can be generated during PCR amplification [51, 39]. To avoid false identification of chimeric sequences as novel taxa [28] or other distortions in bacteria population studies, the detection of these chimeric sequences is very important. Also, it is important to identify the "parental" species a chimeric sequence is composed of. But, depending on the examined taxonomic rank, e.g. phyla or families, the number of parental species taken into consideration can be very large, even comprising several hundreds. The current version of jpHMM is not applicable to such big sequence families due to the complexity of the model which is determined by the number of jumps between subtypes in the model. In the current implementation, the number of jumps per column is quadratic in terms of the number of subtypes. In section 3.3, a modification of the jpHMM architecture that reduces the number of jumps in the model to be linear in terms of the number of subtypes instead of being quadratic is presented. This modification can also be useful in other applications such as the detection of recombinations between sub-subtypes, of which the number can be very high as one can see for HBV where 35 subgenotypes have been identified up to now.

Recombination detection in circular genomes

Sequence analysis of *circular* genomes differs from the analysis of linear genomes. In linearized sequences of circular genomes, dependencies between the 5’ and the 3’ end of the sequence exist. Such dependencies cannot be modeled with a *linear* model like jpHMM as linear models artificially introduce a breakpoint at a certain position in the genome that does not exist in reality. This problem also affects recombination detection in circular genomes: If a breakpoint is located very close to the 5’ or the 3’ end of the sequence the recombination segment at the end of the sequence may be too short to provide enough information for a clear distinction of the subtypes. In this case, the breakpoint would probably be missed since the information about the nucleotide composition at the other end of the sequence is not taken into account. Additionally, such a breakpoint is predicted implicitly at the chosen origin for the sequence coordinates if the two subtypes predicted at both ends of the sequence are different. In section 3.4, an extension of jpHMM for its application to viruses with circular genomes is presented. It predicts recombinations without assuming a particular origin for the sequence coordinates and thus is not biased against recombination breakpoints near the chosen origin. To our knowledge, this is the first approach for recombination detection in circular genomes explicitly taking into account the circularity of the genome.

Proposal for a classification system for recombinant forms of HBV

For HIV-1 and HCV, a well-defined nomenclature exists, clearly distinguishing between pure subtypes and recombinant forms [80, 46]. For HBV, the current classification system is only based on sequence similarity resulting in the classification of recombinant forms as genotypes or subgenotypes. But, for recombination detection tools, a clear definition of pure genotypes is necessary to detect further recombinant forms of known genotypes. Assigning a recombinant form of two genotypes as a subgenotype of one of these genotypes makes it impossible for these tools to distinguish between these two genotypes in further analyses.

In a large-scale study, about 3000 complete HBV genome sequences are evaluated with the circular version of jpHMM to *define circulating recombinant forms of HBV* (section 5.7.4). The aim is to identify recombinant forms that stem from *independent recombination events* and have been established in the population. For this purpose, certain criteria for classifying recombinant sequences are proposed.

Revision of the jpHMM source code

In addition to all modifications of jpHMM, main parts of the original jpHMM source code have been rewritten to provide an object-oriented C++ program that is easy to use and to be modified or extended by other users (section 4.1). Furthermore, webserver enquiries are now managed using the resource management system *Sun Grid Engine* (SGE) [26].

Structure of this thesis

In chapter 2, first, an overview of the biological background on HIV and HBV, including the genetic variability and recombination mechanisms, is given (section 2.1). Then, existing recombination detection tools are presented and the jpHMM approach is described (section 2.2).

In chapter 3, the methods that have been conceived during this thesis are described, starting with the extension of the jpHMM output to include a tagging of regions with uncertain parental subtype predictions and interval estimates for breakpoints (section 3.1). This is followed by a description of the restriction of the search space of the Viterbi algorithm in jpHMM (section 3.2). The modification of the jpHMM architecture that reduces the number of jumps in the model to be linear instead of quadratic in terms of the number of subtypes is described in section 3.3. In section 3.4, a jpHMM for recombination prediction in circular genomes is presented.

The new implementation of jpHMM and its extensions are described in chapter 4, including the input and output format as well as the webserver.

In chapter 5, the results of the methods developed in this thesis are presented and discussed. First, the selection of a suitable background alignment (section 5.1) and the estimation of the jpHMM parameters for the application of jpHMM to HBV genomes (section 5.2) are described. The accuracy of the jpHMM recombination prediction including uncertainty regions and breakpoint intervals is evaluated in section 5.3. In sections 5.4 and 5.5, the quality of different modifications of jpHMM is assessed. The runtime of all jpHMM versions is compared in section 5.6. In section 5.7, first, the accuracy of the circular version of jpHMM for HBV is evaluated. Then, this method is applied to classify all available complete genome sequences of HBV, and CRFs of HBV are defined.

The criteria that we propose to classify recombinant forms of HBV and conceivable approaches to answer the question when the recombination takes place during the replication cycle of HBV are discussed in chapter 6.

Chapter 2

Background

2.1 Viruses

This section will give an overview of the biological background of two viruses that have been examined in this thesis, HIV (subsection 2.1.1) and HBV (subsection 2.1.2). Besides a short introduction into the structure of the genomes and the life cycles, the genetic diversity of the viruses is presented and current discussions about the classification system and nomenclature for these viruses are portrayed.

2.1.1 HIV

HIV belongs to the genus *Lentivirus* in the family of *Retroviridae*. The estimated number of people living with HIV in 2008 was 33.4 million [100], among whom about 2.7 million have become infected in 2008. HIV causes the *Aquired Immuno Deficiency Syndrome* (AIDS) and it is estimated that in 2008 two million people worldwide died due to AIDS-related illnesses. The most heavily affected region by HIV is Sub-Saharan Africa where about $71\,\%$ of all new HIV infections in 2008 occurred, and about 22.4 million people are estimated to be infected with HIV. In some African countries like Botswana, Lesotho and Swaziland, the estimated HIV prevalence is higher than $20\,\%$.

2.1.1.1 Genome

The HIV genome is a single-stranded RNA genome that has a length of approximately $9,500$ nucleotides (nt). It encodes nine proteins: two structural proteins called gag and env, an enzymatic protein called pol, two regulatory proteins called tat and rev, and four accessory proteins called vif, vpr, vpu and nef. The structure of the HIV genome can be seen in Figure 4.3 (p. 73).

2.1.1.2 Replication

Retroviruses replicate via reverse transcription of their RNA genome. After entering the cell, the single-stranded RNA genome is transcribed into a double-stranded DNA using the reverse transcriptase (reviewed in [64]). This double-stranded DNA is then integrated into the host genome and replicated as a part of it. New genomic RNAs are synthesized from this integrated DNA.

2.1.1.3 Types, groups, subtypes and sub-subtypes

Two types of HIV, named HIV-1 and HIV-2, are known. HIV-2 is restricted primarily to West Africa [79]. HIV-1 is phylogenetically divided into three groups, called M (main), O (outlier) and N (non-M/non-O), that arose due to independent transmissions of simian immunodeficiency virus (SIV) from chimpanzees and gorillas into humans [89]. Recently, a new HI virus has been identified. It is closely related to gorilla SIV and it is proposed to designate a new HIV-1 group P [73].

Viruses of group M dominate the global HIV epidemic and are divided into 9 genetically distinct subtypes, A-D, F-H, J and K, whereas some of them have been further subdivided into sub-subtypes (A1, A2, F1, F2) [80]. This classification system is based on the phylogenetic distance of the viruses and has grown historically. For example, subtypes B and D have been defined as two different subtypes, but their phylogenetic distance corresponds rather to that of two sub-subtypes [80]. CRF01_AE (see section 2.1.1.4 "Genetic recombination" for the description of CRFs) was initially classified as subtype E, but later its recombinant structure was detected. Up to now, no complete genome sequence of subtype E has been detected, so CRF01_AE contains the only information of subtype E (`http://www.hiv.lanl.gov/content/sequence/HIV/CRFs/CRFs.html#CRF01`).

2.1.1.4 Genetic recombination

Among these subtypes, recombination is very common [32]. Recombinant forms with an identical mosaic structure that have been identified in at least three epidemiological unlinked individuals, define a *circulating recombinant form* (CRF). Up to now, 48 CRFs and a large number of *unique recombinant forms* (URF), i.e. recombinant forms that have only been identified in one individual, have been identified. A list of known CRFs can be found at the LANL HIV sequence database [49] at
`http://www.hiv.lanl.gov/content/sequence/HIV/CRFs/CRFs.html`.

2.1.1.5 Mechanism of recombination

Each retroviral particle contains two copies of the genomic RNA. If a cell is infected with two different HIV strains, two genetically different RNA copies can be co-packaged in one virion. If this heterozygous virion subsequently infects a new cell, strand-switching of the reverse transcriptase between these two RNA templates during the DNA synthesis can lead to a recombination of two subtypes (reviewed in [64]).

2.1.1.6 Geographic distribution of subtypes and CRFs

The HIV-1 subtypes and CRFs are unevenly distributed throughout the world (Figure 2.1). For example, the most prevalent subtype in Europe and North America is subtype B, while in South-East Asia subtypes B and C and B/C recombinants are predominant [58]. The highest genetic diversity can be found in Central West Africa, where all subtypes and many CRFs exist. In this geographic region, CRF02_AG is the most prevalent strain.

Subtypes A, B, C, and D, and the two CRFs CRF01_AE and CRF02_AG dominate the global epidemic. Hemelaar *et al.* [30] estimated that in 2004, subtype C accounted for about 50 % of all HIV-1 infections worldwide, and that about 18 % of all infections worldwide are associated with CRFs. Subtypes F and G have a low global prevalence but they are involved in circulating recombinant forms such as CRF12_BF and CRF13_BG or CRF02_AG [58].

2.1.1.7 Recombinants versus subtypes

The HIV classification system of subtypes and CRFs has grown historically. Recently, Abecasis *et al.* [2] has hypothesized that subtype G is a circulating recombinant form including subtypes A and J and a putative subtype G parent instead of being a pure subtype. They also found no evidence for recombination within CRF02_AG which was originally classified as a CRF. This hypothesis was refuted in [108] on the basis of large-scale subtyping analysis. This discussion shows that the classification of HIV sequences into subtypes and CRFs has not been completed yet and that accurate HIV classification methods are needed to understand the evolutionary history of HIV.

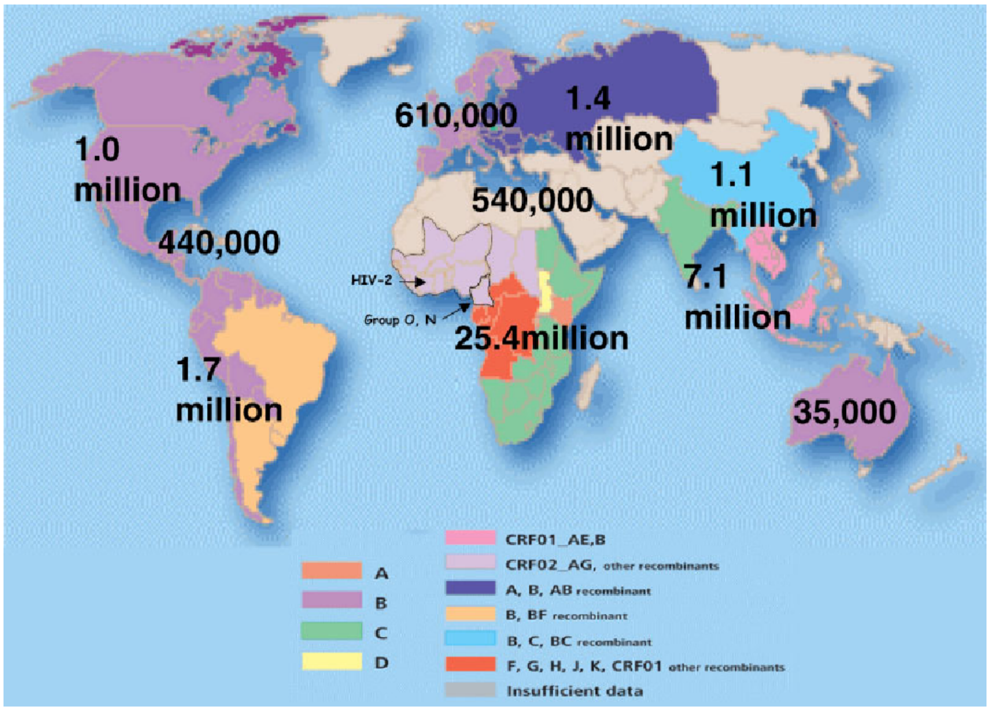


Figure 2.1: Geographic distribution and prevalence of HIV subtypes and CRFs. From [58].

2.1.2 HBV

HBV belongs to the genus *Orthohepadnavirus* in the family *Hepadnaviridiae*. It is estimated that more than two billion people worldwide have been infected with HBV [102], among whom about 360 millions are chronically infected with HBV. Chronic hepatitis B infection can lead to serious illness, such as liver cirrhosis and hepatocellular carcinoma, as well as death. A mathematical model developed by Goldstein *et al.* [27] estimated 620.000 deaths by HBV-related diseases for the year 2000.

2.1.2.1 Genome

The hepatitis B virus has a partially double-stranded, circular DNA genome with a length of approximately $3,200$ bp. The exact length of the genome depends on the genotype (section 2.1.2.3, "HBV genotypes") and ranges from $3,182$ bp (genotype D) [68] to $3,248$ bp (genotype G) [95]. The genome encodes four partially overlapping open reading frames: the surface (S), the core (C), the polymerase (P) and the X (X) gene.

2.1.2.2 Replication

Similar to many retroviruses like HIV, HBV replicates through reverse transcription of a RNA intermediate which is called *pregenomic RNA* (pgRNA) [97]. After the infection of a hepatocyte, the partially double-stranded genome is transported to the nucleus where it is repaired to form a *covalently closed circular DNA* (cccDNA) by completing the partial plus strand. The cccDNA is then transcribed by the RNA polymerase and the pgRNA is produced. In the cytoplasm, the pgRNA is encapsidated along with the viral polymerase. Within this nucleocapsid, reverse transcription of the pgRNA takes place producing the minus strand of the DNA. The partially double-stranded genome is then generated by the synthesis of the plus strand from the minus strand. From these nucleocapsids, either new virions can be formed, that can infect new cells, or the genome is delivered to amplify the pool of cccDNA in the nucleus [88, 7, 36].

2.1.2.3 Genotypes and subgenotypes

Eight *genotypes* of HBV, named alphabetically A-H, have been identified. The classification of HBV viruses into genotypes is based on the divergence of complete nucleotide genomes. In Okamoto *et al.* [68], firstly the four genotypes A, B, C and D were defined on the basis of a sequence divergence of at least $8.0\,\%$. The threshold of $8.0\,\%$ sequence divergence became standard for genotype classification, and four further genotypes, E and F [67, 66, 63], G [95] and H [6] have been identified. In 2008, Kramvis *et al.* demonstrated, on the basis of phylogenetic analysis and pairwise comparisons of 670 complete HBV genomes, that a nucleotide divergence $\geq 7.5\,\%$ can be used as a criterion for the classification of HBV genotypes [41].

Recently, two further putative genotypes, tentatively named I and J, have been identified. Sequences of genotype I have been identified as recombinants of genotypes A and C [29, 107] or A, C and G [34, 69], and are defined as a new genotype on the basis of the mentioned criterion of a sequence divergence $\geq 7.5\,\%$. Until now genotype J has only been isolated in one patient from Japan [98].

For genotypes A - D and F, several *subgenotypes* (A1-A5, B1-B7, C1-C5, D1-D5 and F1-F4) have been identified (Reviewed in [48]) on the basis of a nucleotide divergence higher than $4\,\%$ and lower than $7.5\,\%$. This number is still increasing, e.g. recently, subgenotypes B8, C6 and C7, D6, D7 and D8 [62, 12, 55, 60, 1] have been proposed.

2.1.2.4 Genetic recombination

Recombination among HBV genotypes or subgenotypes is very common. It probably occurs during coinfection with different HBV genotypes, but the exact mechanism of recombination is not yet known.

Up to now, recombinants have been observed among all genotypes except genotype H. A study of Simmonds and Midgley [90] on all published complete genome sequences of HBV, available in October 2004, revealed 24 phylogenetically independent recombinant forms. All detected recombinant forms are recombinations of two genotypes. Most of them have two recombination breakpoints, but also four and six breakpoints have been observed. Additionally, recombinations with unknown genotypes as well as recombinations between human and primate sequences have been found.

2.1.2.5 Mechanism of recombination

The mechanism of intergenotype recombination has not been understood yet. Several hypotheses are conceivable [61]. If recombination takes place during replication, it might occur during reverse transcription of the pregenomic RNA as a result of a jump of the viral polymerase from one molecule to another. But, unlike retroviruses such as HIV, the reverse transcription of HBV takes place inside the nucleocapsid and it is thought that only one pregenomic RNA and the viral polymerase are encapsidated at the same time. Recombinant strains could also arise from homologous recombination [3] between two cccDNA molecules from different HBV genotypes, or during the synthesis of the pregenomic RNA by a jump of the RNA polymerase from one cccDNA molecule to another.

Therefore, it is not yet known if recombination occurs when the HBV genome is present in a circular (as cccDNA or partially double-stranded DNA) or in a linear form (as pregenomic RNA). Homologous recombination would always lead to two recombination breakpoints, whereas recombination during the synthesis or the reverse transcription of the pregenomic RNA results in an *artificial* recombination breakpoint at the end of the pregenomic RNA (in the case that an odd number of breakpoints is introduced), and thus at the position in the circular genome where both ends of the minus strand are linked. See Figure 2 in [7] for the organization of the HBV genome.

2.1.2.6 Geographic distribution of (sub)genotypes

The HBV genotypes are distributed in distinct geographic regions (Figure 2.2). In Eastern Africa, genotype A predominates HBV infections with a prevalence of $93\,\%$ whereas in Western Asia, it is genotype D with a prevalence of $94.8\,\%$ [48]. In all other regions, a variety of genotypes can be found. In Europe and North America, all eight genotypes A-H have been identified, with A and D being the predominant genotypes in Europe. In East and South-East Asia, genotypes B and C are the most prevalent genotypes.

In some geographic regions, recombinant forms have become the predominant strain. For example, in Tibet, the dominant HBV strain is a recombinant form of genotypes C and D, defined as a subgenotype of genotype C [17]. One of the most wide-spread recombinant forms is a recombinant of genotypes B and C circulating in East Asia which is defined as subgenotype B2 [9, 96].

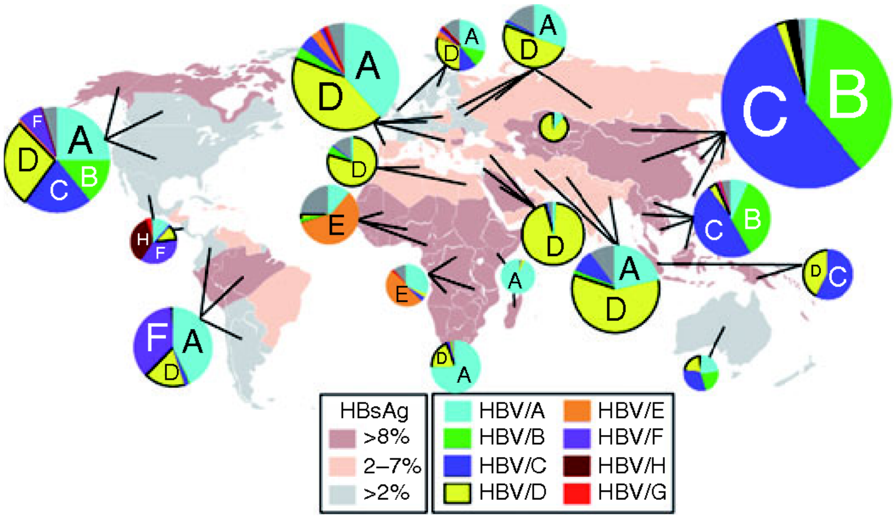


Figure 2.2: Geographic distribution of HBV genotypes. From [48].

2.1.2.7 Problematic genotype definition

Genotyping based on certain genomic regions instead of complete genomes Genotyping of HBV sequences (e.g. [15]) and the definition of new subgenotypes of HBV (e.g. [62, 55]) is often based on phylogenetic analysis of the S gene and/or the precore/core gene instead of the complete genome sequence. This had been suggested by Norder *et al.* [67] in 1992, as their results of a classification based on the S gene were consistent with the results of a previous classification on the basis of complete genome sequences. But, depending on the location of recombination breakpoints, genotyping methods that target only a certain genomic region may fail to detect recombinations [42]. This can lead to an incorrect prediction of recombinant sequences as pure (sub)genotypes. Or more complex recombinants may remain undetected.

Recombinants versus (sub)genotypes In contrast to the nomenclature for HIV, the definition of *circulating recombinant forms* in HBV is not common. New recombinant forms are usually defined as new subgenotypes of a known genotype, e.g. subgenotype D8 which is a recombinant of genotypes D and E detected in Niger [1], or the widespread subgenotype B2, a recombinant of genotypes B and C [96]. Or they are defined as a new genotype as in the above-mentioned case of genotype I.

Nomenclature The current classification system of HBV genotypes and subgenotypes has already been subject of debate [47, 48, 83]. For example, it was proposed that new (sub)genotypes should only be defined on the basis of complete genome sequences and that recombinant forms should be defined as new subgenotypes instead of new genotypes as it happened in the case of genotype I. Also, in contrast to HIV, there is no criterion for the number of epidemiologically unlinked sequences required to define a new genotype or subgenotype.

Purdy *et al.* [75] proposed a different classification system which includes the clustering of genotypes into three higher-order hierarchical groups: group I comprising genotypes A-E and G, group II comprising genotypes F and H, and a hypothetical group III. They also postulate genotype G being a recombinant instead of a pure genotype. This hypothesis is supported by Simmonds and Midgley [90] as well.

An accurate classification of HBV sequences into genotypes and recombinants is indispensable for future analysis. The definition of recombinants as (sub)genotypes, as, for example, in the case of subgenotype B2 or genotype I, may lead to several problems for recombination detection tools such as jpHMM. E.g. in the precore/core region of the HBV genome, it might be impossible for these tools to distinguish between genotype C and subgenotype B2, i.e. between genotypes C and B, if B2 is part of the analysis.

2.2 Methods for recombination detection

The accurate classification of viral genomes and the identification of recombinants, including precise breakpoint definitions, is important in many aspects, such as for epidemiological monitoring, as well as the design of potential vaccines and treatment strategies. For this challenging task, a wide variety of programs for recombination analysis in viruses has been developed during the last years [78].

2.2.1 Overview

The most widely used tool is Simplot [53], which has been applied to many viruses such as HIV-1 and HBV. For a query sequence it provides a graph reflecting the similarity of the sequence to a panel of reference sequences and predicts recombination breakpoints. Simplot is only available for Windows. RIP 3.0 (`http://www.hiv.lanl.gov/content/sequence/RIP/RIP.html`) is a program for detecting recombinations in HIV-1 sequences that was developed at the Los Alamos National Laboratory. It identifies recombination in a query sequence by calculating its similarity to a background alignment of HIV-1 sequences of different subtypes in a sliding window. Depending on how significantly better the 'best matching' background sequence is than the second best match, 'uncertainty regions' in the recombination prediction can be defined. RDP3 [56] is another program developed for Windows that is often used for recombination analysis in different viruses. It uses a range of recombination detection tools to identify recombinant sequences within a given set of aligned sequences. Besides the location of breakpoints, parental sequences of recombinants are determined among the given sequences. A very good visualization tool for locating recombination breakpoints (or breakpoint intervals) in a query sequence is provided by Recco [57]. It identifies the parental sequences within a given set of sequences and indicates robust sequence positions.

jpHMM [87, 109] is a method that we developed during my Diploma thesis [84] to compare a query sequence to a given multiple alignment of a sequence family divided into different subfamilies. It turned out to be a very accurate tool for recombination detection in HIV-1 and hepatitis C virus (HCV) genomes. Based on comparing single representative HIV-1 sequences, it has been shown that recombination breakpoints identified with jpHMM are far more accurate than breakpoints predicted with methods that are traditionally used such as Simplot and RDP [87].

2.2.2 jpHMM

In the following section, a short introduction of jpHMM will be given. A detailed description can be found in

A.-K. Schultz, M. Zhang, T. Leitner, C. Kuiken, B. Korber, B. Morgenstern, M. Stanke
A Jumping Profile Hidden Markov Model and Applications to Recombination Sites in HIV and HCV Genomes
BMC Bioinformatics 2006, **7:**265
doi:10.1186/1471-2105-7-265

2.2.2.1 Model

jpHMM is a probabilistic generalization of the jumping alignment algorithm (JALI) proposed by [92, 93] for protein classification and the detection of remote homologs. For each database sequence, JALI determines the similarity to a given protein family by a local alignment of the database sequence to a given multiple alignment of the sequence family. In contrast to other methods like profile hidden Markov models (profile HMMs) [21], JALI hereby does not take into account the amino acid composition of the single columns in the alignment but aligns each position of the query sequence locally to one sequence in the multiple alignment, the so-called *reference sequence*. The reference sequence can change within this alignment. Such a change of the reference sequence is called a *jump* and the resulting local alignment of the query sequence to the multiple sequence alignment a *jumping alignment*.

In a profile HMM, each column in the given multiple sequence alignment is modeled by a *match* state, that can *emit* the symbols of the underlying alphabet, e.g. nucleotides, with a certain probability. For each two successive columns, an *insert* state exists, allowing for insertions between the two columns, by emitting symbols of the given alphabet with a certain probability. Additionally, for each column a *mute delete* state exists, that does not emit any symbols, allowing for the deletion of the respective column. States of two successive columns are connected by *transitions*. To these transitions *transition probabilities* are assigned. Each query sequence is thought to be generated by a *path* through the model, beginning with a special *begin* state and ending with a special *end* state. The best known implementation of profile HMMs is HMMER [31, 21] and the theory behind profile HMMs is well described in [20].

In contrast to profile HMMs, JALI takes into account the *horizontal information* given in an alignment. By the pairwise alignment of the query sequence to a reference sequence, the relation of neighboring amino acids in a sequence as well as conserved patterns within a subfamily of sequences are taken into account. Additionally, by jumps within the alignment information about amino acids in other sequences at the respective position, i.e. *vertical information*, is considered. But, nevertheless and in contrast to profile HMMs, vertical information in terms of the conservation of alignment columns is missing. For example, *mismatches* of the query sequence to conserved columns are penalized in the same way as mismatches to variable columns in the alignment. jpHMM combines the advantages of both approaches: Assuming that the given sequence alignment is subdivided into different subfamilies, called *subtypes*, each subtype in the alignment is modeled as a profile HMM (described in [20]).

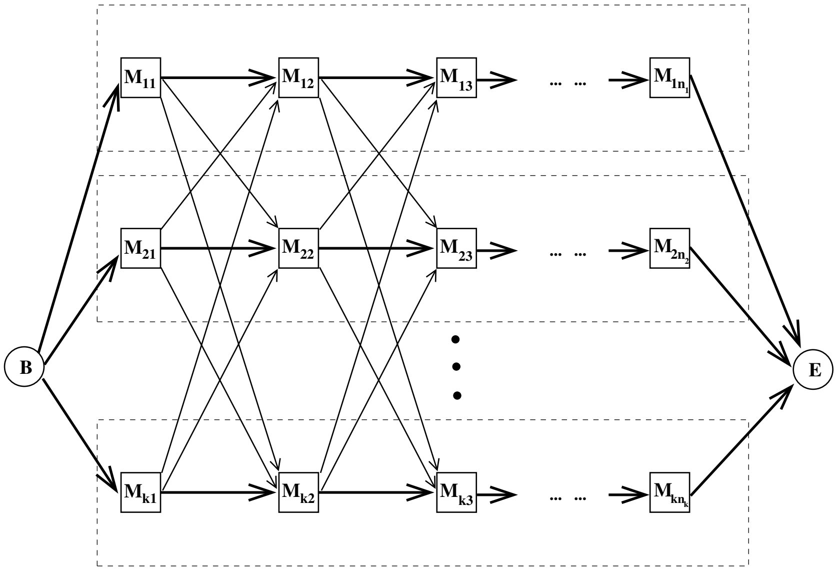


Figure 2.3: Simplified topology of a jpHMM. Each of the k subtypes is modeled as a profile HMM (dashed box). For clarity, only match states are shown. Transitions within a subtype and jumps between different subtypes are shown by arrows. From [87].

That is, each column in a subtype, that is a consensus column for this subtype, is modeled as a *match* state. A match state allows the alignment of a base in the query sequence to the corresponding column in the subtype with a certain probability. This probability is called *emission probability*, and is calculated based on the frequency of the observed nucleotides in the corresponding column. For each match state, an *insert* and a *delete* state exist allowing for the insertion of a nucleotide between the corresponding and the successive consensus column and the deletion of the corresponding consensus column, respectively. The *emission probabilities* of the insert states are also calculated on the basis of observed nucleotide frequencies. Delete states are *mute* states and do not emit any nucleotides. The states are connected by transitions as in standard profile HMMs, to which *transition probabilities* are assigned. In addition to these state transitions *within* a profile HMM, transitions, called *jumps*, *between* the different profile HMMs are allowed at almost any position in the alignment (Figure 2.3). Thus, the model can jump between states corresponding to different subtypes, depending on which subtype is locally most similar to the query sequence. The model that is achieved by the connection of the different profile HMMs by jumps is called a *jumping profile HMM* (jpHMM, Figure 2.4). In addition to the standard match, insert and delete states, the model has a begin and an end state and two single insert and delete states at both ends, respectively, allowing for local alignments (section 3.2.1). In the special case that each subtype consists of exactly one sequence, a jpHMM corresponds to JALI.

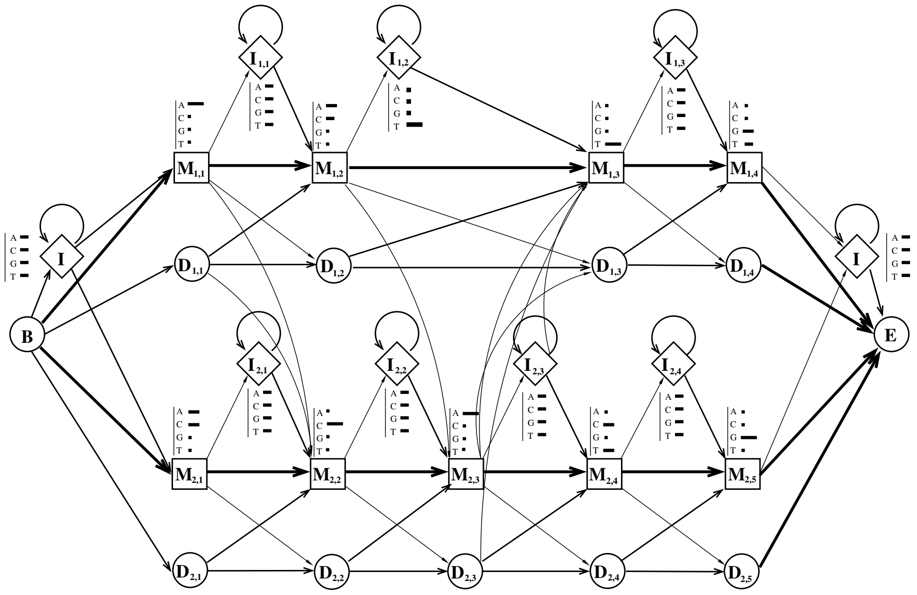


Figure 2.4: A toy example of a jpHMM for a multiple sequence alignment with two subtypes. The first subtype consists of four consensus columns, the second subtype consists of five consensus columns. With each match and insert state a vector is associated for the emission probability values corresponding to the nucleotides A, C, G and T. Fat arrows indicate high transition probabilities, thin arrows correspond to low probabilities. For clarity, some transitions are omitted. From [87].

2.2.2.2 Viterbi algorithm

Like in other HMMs, each sequence S is generated by the model with a certain probability $P(S)$. Usually, more than one path Q through the model, i.e. a sequence of states connected by transitions, beginning with the begin state and ending with the end state, can generate S.

The most probable path that generates S is called the *Viterbi* path and can be determined with the well-known Viterbi algorithm [101]. The Viterbi path maximizes the conditional probability $P(Q|S)$ which is equivalent to maximizing the joint probability $P(Q, S)$:

Let $S = s_1, \ldots, s_l$ be a query sequence. For each position $i = 1, \ldots, l$ of the query sequence S and for each state q in the model

$\delta_i(q)$ is defined as the probability of the prefix $s_1, \ldots, s_i$ of S and the most probable path through the model ending in state q and emitting $s_1, \ldots, s_i$.

$\delta_i(q)$ is called the *Viterbi variable* of position i and state q. Let $t_{q',q}$ be the probability of a transition from state q' to state q and e_{q,s_i} the probability of emitting nucleotide s_i in state q. The Viterbi variables for all sequence positions $i = 1, \ldots, l$ and for all states are calculated recursively:

$$\delta'_{i+1}(q) = \begin{cases} \max\limits_{q'} \delta'_{i+1}(q') t_{q',q}, & \text{if q is a delete state,} \\ e_{q,s_{i+1}} \max\limits_{q'} \delta'_i(q') t_{q',q}, & \text{otherwise.} \end{cases} \tag{2.1}$$

The probability of the most probable path through the model generating S is

$$\delta_l^* := \max_q \delta_l(q). \tag{2.2}$$

The most probable path Q^* through the model that generates S, and thus the jumping alignment of S to A, is determined by backtracking.

2.2.2.3 Recombination prediction

The jumping alignment of a query sequence S to the given multiple sequence alignment A determined with the Viterbi algorithm defines the recombination prediction for S: since each query sequence position is generated by exactly one state of the model and each state of the model only belongs to one profile HMM, each query sequence position is assigned to exactly one profile HMM, and thus to exactly one subtype. Positions of jumps between different subtypes define recombination breakpoints.

2.2.2.4 Runtime and beam-search algorithm

For a query sequence of length l, k subtypes in the alignment and n states in the model, the complexity of the Viterbi algorithm is $O(lkn)$ in time and $O(ln)$ in space. Obviously, the number of states in the model, n, depends on the number of subtypes, k. In general, for each column, in each subtype three states exist. Thus, for an alignment with m columns, the model has roughly $n = 3km$ states. Assuming that the length of a full-length query sequence and the number of columns in the alignment are almost equal, i.e. $m \approx l$, the complexity of the Viterbi algorithm can also be described by $O(l^2k^2)$ in time and $O(l^2k)$ in space. For large genomes, this may exceed the capacity of current computer hardware and especially require too much runtime [87].

To reduce the search space of the Viterbi algorithm and thus to speed-up the runtime, the *beam-search algorithm* was applied [54, 74]. This algorithm is based on the idea to restrict the search space of the Viterbi algorithm to *promising* paths and to exclude possible irrelevant paths. This is achieved by the definition of *active* states for each query sequence position. For each position s_i, a *modified* Viterbi variable $\delta'_i(q) \leq \delta_i(q)$ is calculated and stored only for those states q whose modified Viterbi variable is not much lower than the optimal local solution $\delta^*_i = \max_q \delta_i(q)$. These states are called active states for s_i and the set $\mathcal{A}_i$ of active states at position i is defined by

$$\mathcal{A}_i := \{q \,|\, \delta'_i(q) \geq \mathcal{B}\delta^*_i\}, 0 < \mathcal{B} \ll 1. \tag{2.3}$$

The modified Viterbi variable $\delta'_q(i+1)$ at step $i+1$ needs now only be calculated for states that can be reached by a transition from a state in $\mathcal{A}_i$, i.e. for *successor* states of states in $\mathcal{A}_i$:

$$\delta'_{i+1}(q) = \begin{cases} \max\limits_{q' \in \mathcal{A}_{i+1}} \delta'_{i+1}(q')t_{q',q}, & \text{if q is a delete state,} \\ e_{q,s_{i+1}} \max\limits_{q' \in \mathcal{A}_i} \delta'_i(q')t_{q',q}, & \text{otherwise.} \end{cases} \tag{2.4}$$

The Viterbi variables of *inactive* states are set to 0 and do not need to be saved. $\mathcal{B}$ is called the *beam-width*. It has been set to $\mathcal{B} = 10^{-20}$, which allows for a high accuracy of jpHMM as well as a relatively efficient algorithm. On a Linux PC with 3 GB RAM and 3.2 GHz the CPU time for the tested full-length HIV-1 genomic sequences was between 7.2 min (sequence of CRF 12, length = $8,760$ nt) and 13.6 min (sequence of CRF 11, length = $9,768$ nt). The memory the program required was between 1 GB and 1.5 GB.

For fragmental sequences, such a restriction of the Viterbi search space with the above beam-search algorithm is hardly possible. The reason is that a jpHMM is built for an alignment of full-length genomic sequences and each path through the model generating a query sequence must start in the begin state and end in the end state. To allow an alignment of short fragments that are located at the *end* of the genome at least one (long) path of deletes through the model up to the first match of the query sequence to the alignment must be *active* in the Viterbi algorithm. Since the probability of such a long path of deletes is usually very low compared to other paths through the model the beam-search restriction cannot be applied in this case because otherwise such a path would be inactive leading to incorrect alignments of such short sequences. Thus, depending on the length of the sequence, for short sequences, a much lower beam-width $\mathcal{B}$ is chosen allowing a larger search space of the Viterbi algorithm. For example, the beam-width for HIV-1 sequences of length ≤ 500 nt has been set to $\mathcal{B} = 10^{-45}$.

2.2.2.5 Parameters

A jpHMM has a large number of parameters. For each state in the model, the probabilities of the transitions to all successor states, i.e. transitions within a profile HMM and jumps to other profile HMMs, need to be specified. Additionally, for match and insert states, the *emission probabilities* of all four nucleotides need to be calculated. The emission and transition probabilities within a profile HMM are calculated on the basis of observed nucleotides and transitions in the given input alignment respectively. For a match state, the observed nucleotides in the corresponding column in the respective subtype are taken into account, for an insert state, the observed nucleotides in the columns between the corresponding and the successive consensus column.

Also, transitions can be observed in the given alignment. Each sequence in the alignment defines a unique path through the model. These paths give rise to observed transition frequencies. For example, two bases in a sequence aligned to two successive consensus columns define a transition from the match state corresponding to the first of these consensus columns to the successive match state. Pseudocounts are added to all observed frequencies to avoid probabilities equal to 0:

Let $\vec{n} = (n_1, \ldots, n_r)$ be a vector of observed nucleotides (or transitions) out of a certain state, with r being the number of emissions (or transitions) and n_i, $i = 1, \ldots, r$, the observed frequency of the ith emission (or transition). Let $\vec{\alpha} = (\alpha_1, \ldots, \alpha_r)$ be the respective pseudocounts. Then the probability of the ith emission (or transition) out of this state is calculated by

$$p_i = \frac{n_i + \alpha_i}{|\vec{n}| + |\vec{\alpha}|} \tag{2.5}$$

Jump probabilities cannot be calculated on the basis of observed frequencies, since jumps cannot be observed in the given multiple sequence alignment. Therefore, a fixed, empirically derived jump probability jp is used. Since each jump from a state to another subtype is considered as being equally likely, the probability of such a jump is jp/K, if jumps to K other subtypes are possible.

The estimation of the jpHMM parameters for HIV-1 is described in detail in [87] and [84]. The resulting parameters are:

Pseudocounts for transition probabilities The pseudocounts for the transition probabilities, α_{trans}, are taken from [104]:

$$\begin{aligned} \vec{\alpha}_{\text{trans}} &:= (\alpha_{\text{MM}}, \alpha_{\text{MI}}, \alpha_{\text{MD}}, \alpha_{\text{IM}}, \alpha_{\text{II}}, \alpha_{\text{DM}}, \alpha_{\text{DD}}) \\ &= (0.794, 0.95, 0.005, 0.333, 0.667, 0.278, 0.222) \end{aligned} \tag{2.6}$$

Each entry in the vector represents the pseudocount for a certain type of transition, described by the indices. For example, α_{MI} is the pseudocount for the calculation of the transition probability from a match (M) state to an insert (I) state and α_{DM} the pseudocount for the calculation of the transition probability from a delete (D) state to a match (M) state.

Pseudocounts for emission probabilities The pseudocounts for the calculation of the emission probabilities are estimated as described in [91]. The estimated pseudocounts for the emission probabilities of the match (M) and the insert (I) states are:

$$\begin{aligned} \text{Match states:}\quad \vec{\alpha}_{\text{em,M}} &:= (\alpha_{\text{M,A}}, \alpha_{\text{M,C}}, \alpha_{\text{M,G}}, \alpha_{\text{M,T}}) \\ &= (0.09, 0.05, 0.06, 0.05) \qquad (2.7) \\ \text{Insert states:}\quad \vec{\alpha}_{\text{em,I}} &:= (\alpha_{\text{I,A}}, \alpha_{\text{I,C}}, \alpha_{\text{I,G}}, \alpha_{\text{I,T}}) \\ &= (1.01, 1.01, 1.01, 1.01) \qquad (2.8) \end{aligned}$$

Here, the indices describe the type of state (M or I) and the nucleotide (A, C, G or T) that is emitted.

Jump probability The jump probability jp was empirically derived on the basis of recombination breakpoints of published CRFs (`http://www.hiv.lanl.gov/content/sequence/HIV/CRFs/CRFs.html`):

$$jp = 10^{-9} \tag{2.9}$$

2.2.2.6 Accuracy

jpHMM was evaluated on real virus genome sequences as well as simulated recombined genome sequences [87]. It has been demonstrated that jpHMM is far more accurate than competing methods such as Simplot [53] for phylogenetic breakpoint detection. Comparing single representatives, the recombination breakpoints identified by jpHMM were found to be significantly more accurate than breakpoints defined by methods that are traditionally used.

2.2.2.7 Availability

For HIV-1 genomes, jpHMM is available online at `http://jphmm.gobics.de` [109]. The user can paste or upload either full-length HIV-1 genomic sequences or fragments. In addition, a command line version of jpHMM can be downloaded, which can be used for any virus or other data, if a multiple sequence alignment subdivided into different subtypes is available. The parameters of the model are provided for HIV-1 and HCV.

The web server has been described in

M. Zhang, A.-K. Schultz, C. Calef, C. Kuiken, T. Leitner, B. Korber, B. Morgenstern, M. Stanke
jpHMM at GOBICS: a web server to detect genomic recombinations in HIV-1
Nucleic Acids Research 2006 34:W463-W465
doi:10.1093/nar/gkl255

Chapter 3

Improvements, extensions and modifications of jpHMM

jpHMM has been proven to be a very efficient and accurate tool for predicting recombinations in viral genomes. Nevertheless, it is important to develop and improve the model further. One main aspect is the reliability of the predicted recombination. In section 3.1, a method to verify the predicted parental subtypes and the location of recombination breakpoints, and, in case, to present alternative solutions, is described. A disadvantage of jpHMM (section 2.2.2) is the complexity of the Viterbi algorithm. In section 3.2, a method is described that restricts the search space of the Viterbi algorithm and thus the runtime of jpHMM. A modification of the original jpHMM architecture that reduces the number of jumps in the model to be linear in terms of the number of subtypes instead of being quadratic is presented in section 3.3. In section 3.4, a jpHMM for circular viral genomes is described. Recombination detection in circular genomes is a challenge due to several reasons. These reasons and possible solutions are demonstrated.

3.1 Uncertainty regions in recombination prediction and breakpoint intervals

The predicted recombination for a query sequence S is defined by the most probable path through the model generating S, the *Viterbi path*. But, since incorrect subtype assignment or recombination prediction may lead to wrong conclusions in epidemiological or vaccine research, it is necessary to introduce a measure of the reliability of the predicted recombination breakpoints and parental subtypes.

A suitable measure for this is the so-called *posterior probability* for each base of the query sequence S and each subtype in the given alignment. This quantity denotes that the base belongs to the subtype in the model. The posterior probabilities are calculated using the well-known *Forward* and *Backward algorithms*. Based on these probabilities, regions in the query sequence where the model is *uncertain* about the predicted parental subtype are tagged and an *interval* estimate for each predicted breakpoint, i.e. an interval where the breakpoint can be expected to be located, is defined. These regions are called *uncertainty regions* and *breakpoint intervals* respectively.

Posterior probabilities have also been used in other sequence analysis tools to represent the degree of confidence in the respective output. For example, in HMMER3, the latest version of the protein sequence analysis tool HMMER [31, 21], for each sequence alignment, posterior probabilities are calculated to decide which parts of the alignment are well-determined or not.

This part of the thesis is published in

A.-K. Schultz, M. Zhang, I. Bulla, T. Leitner, B. Korber, B. Morgenstern, M. Stanke.
jpHMM: Improving the reliability of recombination prediction in HIV-1
Nucleic Acids Research 2009 37:W647-W651
doi:10.1093/nar/gkp371

3.1.1 Forward and backward variables

As in other HMMs, a sequence $S = s_1, \ldots, s_l$ is generated by the model with a certain probability $P(S)$. In general, a sequence S can be generated by more than one path Q through the model. For each position $i = 1, \ldots, l$ of the query sequence S and for each state q in the model

$\alpha_i(q)$	is defined as the probability of emitting the prefix $s_1, \ldots s_i$ of S by any path through the model ending in state q at position i, $\alpha_i(q)$ is called the *Forward variable* of position i in S and state q
$\beta_i(q)$	is defined as the probability of emitting the suffix $s_{i+1}, \ldots s_l$ of S by any path through the model starting in state q at position i, $\beta_i(q)$ is called the *Backward variable* of position i in S and state q

Then, $P(S) = \sum_{q} \alpha_l(q)$ is the probability of emitting $s_1, \ldots, s_l$ by any path through the model.

Both types of variables are calculated with a dynamic programming algorithm, the well-known *Forward* and *Backward algorithm*, respectively. In jpHMM, the Forward variables are calculated along with the Viterbi algorithm and, to accelerate the computation and to restrict the search space as described in section 2.2.2.4 for the Viterbi algorithm, the Forward variables are also only calculated for those states that are defined as *active* states by the beam-search algorithm (Eq. (2.3)). That is, for each query sequence position s_i, a modified Forward variable $\alpha'_q(i) \leq \alpha_q(i)$ is calculated and stored only for states q in the set $\mathcal{A}_i$ of active states, determined with the beam-search algorithm. The modified Forward variable of an *inactive* state is set to 0 and must not be saved. For position $i+1$, the modified Forward variable $\alpha'_{i+1}(q)$ of a state q is then only calculated for states that are successors of states in $\mathcal{A}_i$ ($\mathcal{A}_{i+1}$ if q is a delete state):

$$\alpha'_{i+1}(q) = \begin{cases} \displaystyle\sum_{q' \in \mathcal{A}_{i+1}} \alpha'_{i+1}(q') t_{q',q}, & \text{if q is a delete state,} \\ e_{q,s_{i+1}} \displaystyle\sum_{q' \in \mathcal{A}_i} \alpha'_i(q') t_{q',q}, & \text{otherwise.} \end{cases} \tag{3.1}$$

The Backward variables are calculated in a separate algorithm and are only required for the calculation of the posterior probabilities (section 3.1.2). Due to the definition of the posterior probabilities (Def. (3.3)), the posterior probability of a certain state at a certain query sequence position is 0, if the Forward or the Backward variable of the respective position and state is 0. Therefore, the posterior probabilities can only be calculated for states that are active in the Forward as well as in the Backward algorithm. Thus, for each query sequence position the Backward variables are only calculated for those states that are defined as active states for this position in the Viterbi/Forward algorithm. Analogous to the Forward algorithm, for each position i, a modified Backward variable $\beta'_q(i) \leq \beta_q(i)$ is calculated only for states $q \in \mathcal{A}_i$, and the modified Backward variables of *inactive* states are set to 0. For position i, the modified Backward variable $\beta'_i(q)$ of a state q is then only calculated for states that are predecessors of states in $\mathcal{A}_{i+1}$ (predecessors of delete states in $\mathcal{A}_i$):

$$\beta'_i(q) = \sum_{\substack{q' \in \mathcal{A}_i, \\ q' \text{ is delete state}}} t_{q,q'} \beta'_i(q') + \sum_{\substack{q' \in \mathcal{A}_{i+1}, \\ q' \text{ is non-delete state}}} t_{q,q'} e_{q',s_{i+1}} \beta'_{i+1}(q') \tag{3.2}$$

3.1.2 Posterior probabilities

3.1.2.1 Posterior probability of a state

As for other HMMs, the *posterior probability* $P_{\text{post},i}(q)$ of position i in S and a *state* q is the probability that s_i is emitted by state q given that the query sequence S is emitted by a path through the model. That is the probability that the prefix $s_1, \dots, s_i$ is emitted by any path through the model ending in state q, i.e. $\alpha_i(q)$ multiplied by the probability that the suffix $s_{i+1}, \dots, s_l$ is emitted by any path through the model, starting in state q at position i, i.e. $\beta_i(q)$, divided by the probability of generating S, $P(S)$:

$$P_{\text{post},i}(q) = \frac{\alpha_i(q) \cdot \beta_i(q)}{P(S)} \tag{3.3}$$

Using *modified* Forward and Backward variables (Def. (3.1) and (3.2)) in jpHMM, the *modified* posterior probability $P'_{\text{post},i}(q)$ of position i in S and a *state* $q \in \mathcal{A}_i$ is defined by

$$P'_{\text{post},i}(q) = \frac{\alpha'_i(q) \cdot \beta'_i(q)}{\sum\limits_{q \in \mathcal{A}_i} \alpha'_i(q) \cdot \beta'_i(q)} \tag{3.4}$$

Obviously, the modified posterior probability of a certain state q at a certain query sequence position i is 0 if $\alpha'_i(q)$ or $\beta'_i(q)$ is 0. Therefore, for all states that are not active for a certain query sequence position, the modified posterior probability is 0, and thus the modified posterior probabilities must only be calculated for states in the respective set of active states.

3.1.2.2 Posterior probability of a subtype

We define the posterior probability $P_{\text{post},i}(\mathcal{S})$ of position i in S and a *subtype* $\mathcal{S}$ as the probability that s_i belongs to subtype $\mathcal{S}$, i.e. s_i is emitted by a state belonging to the profile HMM of $\mathcal{S}$ (profile HMM$(\mathcal{S})$), given that the query sequence S is emitted by a path through the model. Thus, the posterior probability of a position i in S and a subtype $\mathcal{S}$ is the sum of the posterior probabilities of all states belonging to the profile HMM of $\mathcal{S}$ at position i of S:

$$P_{\text{post},i}(\mathcal{S}) := \sum_{q \in \text{ profile HMM}(\mathcal{S})} P_{\text{post},i}(q) \tag{3.5}$$

3.1.3 Uncertainty regions and breakpoint intervals

For each query sequence position and each subtype in the given alignment, the posterior probability is calculated. Based on these probabilities, firstly *uncertainty regions* (UR) in the predicted recombination for the query sequence and secondly *interval estimates of breakpoints*, called *breakpoint intervals* (BPI), are defined (Workflow 3.1).

3.1.3.1 Uncertainty region

If at a certain position i of a query sequence S the posterior probability of the subtype predicted by jpHMM for this position is lower than a certain threshold $0 \ll t_{\text{UR}} < 1$ the prediction for this position is marked as *uncertain* (Figure 3.1 a). This classification accounts for the fact that there is a significant ($\geq 1 - t_{\text{UR}}$) probability that the predicted subtype is wrong according to the probabilistic model.

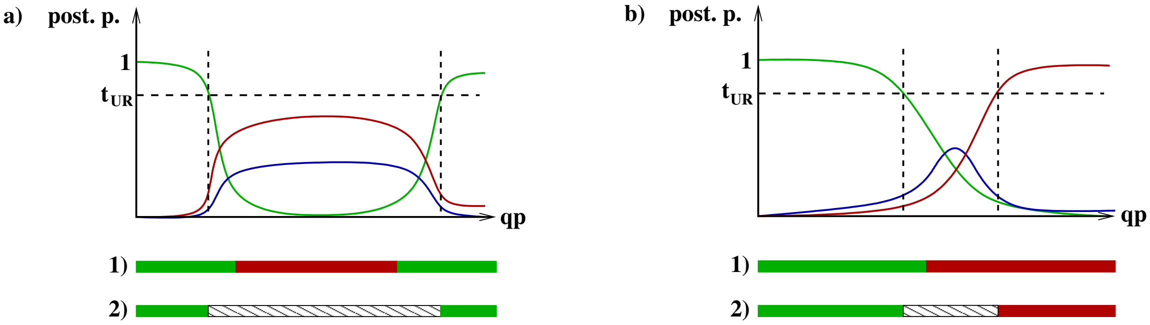


Figure 3.1: Two examples for uncertainty regions in predicted recombinations. In both figures, for each query sequence position (qp), the posterior probabilities (post. p.) of three subtypes are plotted. t_{UR} marks the posterior probability threshold for the definition of uncertainty regions. Vertical dashed lines define the extent of uncertainty regions. The first bar (1) below the plot of the posterior probabilities shows the original recombination prediction with precise breakpoint positions. The second bar (2) shows the predicted recombination including uncertainty regions (hatched regions). In a) an uncertainty region is defined because the posterior probability of the predicted subtype (red) is below t_{UR}. In b) the region around the predicted breakpoint is not defined as a breakpoint interval since the posterior probability of a third subtype (blue) is higher than the posterior probabilities of the subtypes predicted to the left (green) and to the right (red) of the breakpoint.

For uncertainty regions, no parental strain can confidently be determined. But both a text file with the posterior probabilities for all query sequence positions and all subtypes as well as a graph of the posterior probabilities are part of the new jpHMM output (see chapter 4, Implementation). Thus, information about which subtypes are most closely related in these regions is given.

3.1.3.2 Breakpoint interval

For each predicted breakpoint position (defined by the Viterbi path) the corresponding breakpoint interval is defined by the interval around the predicted breakpoint position, where the posterior probabilities of the two subtypes predicted to the left and the right of the breakpoint are lower than a certain threshold $0 \ll t_{\text{BPI}} < 1$, but higher than the posterior probabilities of all other subtypes (Figure 3.2). The maximum extent of such a breakpoint interval is limited by the position of the preceding and the successive predicted breakpoint (if one of the breakpoints does not exist the maximum extent is restricted by the corresponding sequence end). Therefore, if the posterior probability of the subtype predicted to the left (to the right, resp.) of the breakpoint does not reach the threshold t_{BPI} at any position within between the preceding and the current breakpoint (between the current and the successive breakpoint, resp.), the whole interval is defined as an uncertainty region. This also happens if the posterior probability of a third subtype is higher than the posterior probability of one of the two predicted subtypes in this region (Figure 3.1 b), to indicate the possibility of an undetected recombination segment. If the predicted breakpoint is located outside of the breakpoint interval defined by the posterior probabilities (Figure 3.2 b)) the breakpoint is extended to include the predicted breakpoint. The length of a predicted

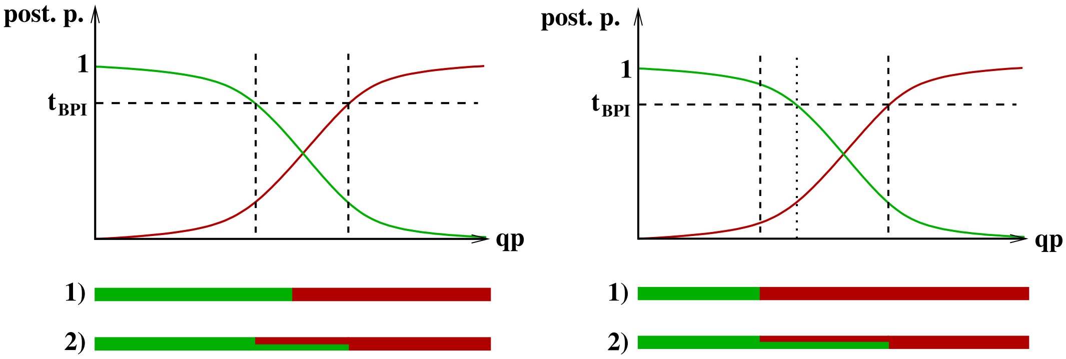


Figure 3.2: Two examples for breakpoint intervals in predicted recombinations. In both figures, for each query sequence position (qp), the posterior probabilities (post. p.) of two subtypes are plotted. t_{BPI} marks the posterior probability threshold for the definition of breakpoint intervals. Vertical dashed lines define the extent of breakpoint intervals. The first bar (1) below the plot of the posterior probabilities shows the original recombination prediction with precise breakpoint positions. The second bar (2) shows the predicted recombination including breakpoint intervals (two-color region). In a) a breakpoint interval around the predicted breakpoint is defined by the region where the posterior probability of the predicted subtypes (green and red) is below t_{BPI}. In b) the original left end (dotted line) of the breakpoint interval defined by the posterior probabilities of the predicted subtypes is moved to the left (dashed line) to include the predicted breakpoint position.

breakpoint interval indicates how precisely the breakpoint can be located reliably. A large interval, for example, is the consequence of the uncertainty of the model to locate the exact breakpoint position between two subtypes.

Regions that are initially defined as uncertainty regions (e.g. often close to predicted breakpoint positions) and secondly defined as breakpoint intervals, are regarded as breakpoint intervals and not as uncertainty regions. Due to the order of defining uncertainty regions and breakpoint intervals (Workflow 3.1), it is appropriate to define $t_{\text{BPI}} \leq t_{\text{UR}}$. The chosen thresholds are given in chapter 5, Results, section 5.3.

Workflow 3.1 (Definition of uncertainty regions and breakpoint intervals)
Let $S = s_1, \dots, s_l$ be a query sequence. Let $\mathcal{S}$ be the set of subtypes and $st = st[1, l], st_i \in \mathcal{S}, i \in [1, l]$, the predicted sequence of subtypes for S. A recombination breakpoint is usually located *between* two successive query sequence positions, e.g. i and $i + 1$, which is notated as $i/i + 1$. Here, a breakpoint b_j describes the breakpoint $b_j/b_j + 1$, i.e. b_j is the position to the left of the breakpoint. Let $\mathcal{B} = \{b_1, \dots, b_{s_k}\}$ be the set of all predicted recombination breakpoints in S.

1. **Definition of uncertainty regions**
2. **Definition of breakpoint intervals:**
   for each breakpoint $b_j/b_j + 1$ define the surrounding breakpoint interval:
   (a) **definition of the left boundary of the breakpoint interval:**
   let $st[b_j]$ be the subtype predicted to the left of the breakpoint.
   Define the position i_{left}, $b_{j-1} < i_{\text{left}} \leq b_j$ in the query sequence where the posterior probability of $st[b_j]$ reaches the threshold t_{BPI},
   i.e. $P_{\text{post},i_{\text{left}}}(st[b_j]) \geq t_{\text{BPI}}$, decreasing i_{left} and starting with $i_{\text{left}} = b_j$.
   (b) **definition of the right boundary of the breakpoint interval:**
   let $st[b_j + 1]$ be the subtype predicted to the right of the breakpoint.
   Define the position i_{right}, $b_j < i_{\text{right}} \leq b_{j+1}$ in the query sequence where the posterior probability of $st[b_j + 1]$ reaches the threshold t_{BPI},
   i.e. $P_{\text{post},i_{\text{right}}}(st[b_j + 1]) \geq t_{\text{BPI}}$, increasing i_{right} and starting with $i_{\text{right}} = b_j + 1$.

```
IF   (one of these positions $i_{\text{left}}$ or $i_{\text{right}}$ cannot be found)
     the region remains defined as uncertainty region.
ELSE
     check the posterior probabilities of all other subtypes within $[i_{\text{left}}, i_{\text{right}}]$ :
     IF   (a subtype $\mathcal{S}, \mathcal{S} \neq st[b_j]$ and $\mathcal{S} \neq st[b_j + 1]$, exists
          that has a higher posterior probability than the two predicted subtypes, i.e.
          $P_{\text{post},i}(S) \geq P_{\text{post},i}(st[b_j])$ and $P_{\text{post},i}(S) \geq P_{\text{post},i}(st[b_j + 1]))$
          the region is also defined as uncertainty region.
```

3.2 Restriction of the search space of the Viterbi algorithm

In the original jpHMM program, the beam-search algorithm was chosen as method to restrict the search space for the Viterbi algorithm. The average runtime of jpHMM including the beam-search algorithm is about ten minutes for nearly full-length HIV-1 sequences. For fragmental sequences, the runtime is much higher since the beam-search algorithm has almost no effect on the search space restriction (section 2.2.2, p. 21).

In the Los Alamos HIV sequence database [49], currently more than 2,300 full-length and more than 330,000 fragmental HIV-1 genomic sequences are available. There is a need to reclassify all the available HIV-1 sequences to enable a better understanding of the dynamics of the global HIV-1 epidemic and the role of recombination. Leading HIV experts from the Los Alamos National Laboratory already analyzed, in close collabpration with us, more than $9,400$ sequences from three epidemically important regions [108]. For 4.9% of these sequences, subtype assignments were different from the ones published in the original literature, demonstrating that a careful reclassification of the whole database is necessary. With the current jpHMM version, this would obviously take too much time, even if the program parallely runs on several computers. Therefore, a further acceleration of the program is necessary. For fragmental sequences, this can relatively easily be achieved by determining the location of the sequence(s) relative to the reference genome. This will be described in the following section 3.2.1. In section 3.2.2, an approach for the reduction of the runtime for full-length as well as fragmental sequences is presented. By a pre-alignment of the sequence(s) to the given multiple sequence alignment, the search space of the Viterbi algorithm can be restricted considerably.

3.2.1 Location of the input sequence relative to the reference genome

For fragmental sequences, a reduction of the runtime can be achieved by determining the location of the query sequence relative to the reference genome. For HIV sequences, this is done with the *HIV Sequence Locator* (`http://www.hiv.lanl.gov/content/sequence/LOCATE/locate.html`), a tool that finds the start and end position of the query sequence relative to the reference strain HXB2 (HIV reference genome, GenBank accession number K03455) [40]. The HIV Sequence Locator is implemented in the web server application of jpHMM.

Since the HXB2 sequence is included in the HIV-1 multiple sequence alignment, the start and end position of a query sequence based on HXB2 numbering can easily be mapped onto alignment columns. The alignment columns corresponding to the start and end position of the query sequence in the reference genome define the region in the multiple sequence alignment the query sequence can be aligned to. Such a local alignment is allowed by two special *delete* states at the beginning and the end of the HMM, respectively (Figure 2.4). The delete state immediately after the begin state allows for transitions to each match state in the model, the delete state immediately before the end state can be reached from each match state. Therefore, a transition from the begin state to each match state and a transition from each match state to the end state is possible.

In the Viterbi algorithm only states within the given columns are set to be *active*, i.e. the Viterbi path that defines the predicted recombination is the most probable path through this *local HMM*, beginning in the *begin* state and ending in the *end* state, generating the query sequence. Therefore, for short sequences, this local alignment can result in an immense reduction of runtime and memory, but for full-length sequences no improvement is achieved. Additionally, within the given start and end position of the query sequence based on HXB2 numbering, all alignments are possible, including long insertions or deletions. I.e. each query sequence position is allowed to be aligned to any column in the alignment (certainly taking into account the preceding and successive sequence positions). This is not necessary since the approximate position in the alignment can be determined for each query sequence position. The following heuristic approach is implemented to reduce the runtime and the memory of the Viterbi algorithm in jpHMM for query sequences of any length.

3.2.2 Definition of active states in the Viterbi algorithm

In general, each query sequence position can be aligned to each column in the given alignment, i.e. it can be emitted by each (non-mute) state in the model. If the query sequence and the alignment (or the respective region) have the same length, this leads to a complexity of the Viterbi algorithm of $O(l^2k^2)$ in time (section 2.2.2.4, p. 21) with l being the length of the query sequence and k the number of subtypes. Thus, the complexity of the Viterbi algorithm is quadratic in the length of the query sequence.

For sequences that share a certain degree of similarity with the sequences included in the multiple sequence alignment, the complexity of the Viterbi algorithm can be reduced considerably. By a pre-alignment of the query sequence to the multiple alignment (see the following subsection 3.2.2.1), it is possible to map each query sequence position to a certain column (or a certain region) in the alignment. This column (or region) defines the part of the alignment to which the respective query sequence position is allowed to be aligned with jpHMM. Thus, for each query sequence position, the search space of the Viterbi algorithm is restricted to states corresponding to the respective assigned column (or region). Depending on the size of the assigned region, the complexity of the Viterbi algorithm can even be reduced to be linear in the length of the query sequence, i.e. $O(lk^2)$.

3.2.2.1 Definition of active alignment columns

In order to reduce the search space of the Viterbi algorithm, the query sequence is compared to a set of selected sequences from the alignment. It is aligned to each of these sequences pairwisely so that each query sequence position is mapped to a certain set of alignment columns. These columns define a certain region (Workflow 3.5) in the alignment to which the respective sequence position is allowed to be aligned with jpHMM. Since most of the columns in the alignments we study are reasonably conserved, the size of these regions is very short, usually.

As pairwise alignment tool, the BLAST-like alignment tool (BLAT) [37] is chosen. It is shortly described in the following paragraph. Since only alignments of nucleotide sequences are taken into account in this thesis, this overview is focused on DNA sequences.

BLAT BLAT is a very fast and accurate tool for mRNA/DNA and cross-species protein alignments with an easy-to-use output. In many aspects, BLAT is very similar to the Basic Local Alignment Search Tool (BLAST) that was developed in 1990 by Altschul *et al.* [4, 5] as a method to search DNA and protein sequence databases for local similarities between sequences. BLAST is based on the idea that homologous sequences share (short) regions of very high similarity. For DNA sequences, a list of all contiguous K-mers (words) occurring in the query sequence is compiled. After removing "uninformative" words (for example, repeats [4]) the database sequences are scanned to identify word pairs (matches) in the query and the database sequence whose aligned score is greater than a certain threshold. Each such 'hit' is extended in both directions, allowing gaps in the alignment, until the score of the current alignment has dropped more than a certain value below the maximum score yet seen. Then, the BLAST output comprises all alignments with a score above a certain threshold.

In the main workflow, BLAT [37] is very similar to BLAST. In an initial step, the so-called *search stage*, regions in the two sequences that are likely to be homologous are identified. In a second *alignment stage*, the previously defined homologous regions are aligned, extended as far as possible allowing no mismatches, and stitched together. But in several points the two programs differ significantly. One of the main differences can be found in the *search stage*: in contrast to BLAST, that builds an index of the query sequence and scans linearly through the database, BLAT builds an index of all non-overlapping K-mers in the database and then scans linearly through the query sequence in order to find homologous regions. This is mainly responsible for the enormous speed-up of BLAT compared to other programs. Additionally, BLAT allows two alternative search methods: instead of requiring perfect matches of a database sequence with a K-mer to initiate an alignment, hits with one mismatch are allowed or multiple perfect matches near each other are required. Both alternatives reduce the runtime of the alignment stage significantly. The default option for nucleotide alignments is the "two perfect 11-mer match criterion". At DNA level, BLAT works well for sequences with a similarity greater than or equal to $90\,\%$. More divergent alignments might be missed but BLAT is able to align sequences that include large inserts.

We chose BLAT for the following reasons: Most parts of the HIV-1 as well as the HBV genome are very conserved and show a genetic divergence lower than $10\,\%$. Only few regions with a higher genetic variability are included. For example, the highest genetic divergence between different HIV-1 subtypes can be observed in the env and the gag region with a maximum divergence of $\sim 35\,\%$ and $14\,\%$ respectively. Genotypes of HBV are distinguished on the basis of a genetic divergence of at least $8\,\%$. Thus, in some regions in the HBV genome, the genetic divergence may also exceed $10\,\%$. In these regions, poor or even no alignments will be produced with BLAT. But, since most of the genomic regions show a much lower genetic divergence and as we are only interested in reliable alignments, only alignments in conserved regions are taken into account anyway. In variable regions, the whole region is assigned to the respective query sequence position (Workflow 3.5, step 4), and the beam-search algorithm is applied to restrict the search space of the Viterbi algorithm. Thus, BLAT is a suitable tool which will also be proven in section 5.4.2 (p. 96). Additionally, the BLAT strategy of building an index of the database, here the given alignment sequences, and scanning through the query sequence is beneficial for our application, especially when a great number of query sequences is examined. The program builds an index of the chosen alignment sequences only once, and then scans linearly through any number of query sequences.

Workflow for definition of active alignment columns The following workflow 3.5 describes how a query sequence is aligned to a certain set of sequences in the given multiple sequence alignment, and the following assignment of alignment columns to each query sequence position. An example for such a workflow is given in Figure 3.3. First, a few definitions will be introduced that will be helpful in the description of the workflow:

Definition 3.2 (Conserved alignment column) *A column in a multiple sequence alignment A is called a* conserved column*, if a certain nucleotide is observed in at least $C\,\%$ of the sequences in A.*

Definition 3.3 (Gap alignment column) *A column in a multiple sequence alignment A is called a* gap column*, if a gap is observed in at least $G\,\%$ of the sequences in A. So, a* non-gap *column is a column in which less than $G\,\%$ of the sequences have a gap.*

Definition 3.4 (Active alignment column) *A column in a multiple sequence alignment A is called an* active column*, if it is*

- *a conserved column (Def. 3.2) and*
- *a non-gap column (Def. 3.3) and*
- *has a certain number of adjacent conserved non-gap columns to the left and to the right.*

Workflow 3.5

Let $S = (s_1, \ldots, s_l)$ be a query sequence of length l and A the given multiple sequence alignment, subdivided into subtypes.

1. **Definition of *database* D for BLAT alignment:**
   N sequences are selected randomly from A, equally distributed on all subtypes. The *raw* sequences, i.e. without gaps, build the *database* D for the BLAT alignments. The subset of these sequences builds the *subalignment* A_D of the database sequences in A. For each database sequence, each position is mapped to the corresponding column in A_D.

2. **Definition of *active* columns in the subalignment A_D of A**:

   2.1. **definition of *conserved columns* in A_D**

   2.2. **definition of *non-gap columns* in A_D**

   2.3. **definition of *active columns* in A_D**

3. **Alignment of query sequence S to database D with BLAT**:
   S is aligned to each database sequence with BLAT. Thus, each base s_i is aligned to a certain number R of bases in the database sequences (BLAT allows gaps in the alignment, so the number R of aligned sequences can be lower than N). Since each base in a database sequence corresponds to a certain column in A_D, each base s_i is aligned to a certain number R of columns in A_D: Let

$$x(s_i) = (x_1, x_2, \ldots, x_R), \quad x_1 \leq x_2 \leq \ldots \leq x_R, \tag{3.6}$$

   be the sorted list of columns in A_D, s_i is aligned to.

4. **Definition of *active alignment column intervals* for each base in S:**
   for each query sequence position s_i,

$$I(s_i) = [I_l(s_i), I_r(s_i)], \tag{3.7}$$

denotes the interval of active alignment columns in A, i.e. the columns in A, s_i can be aligned to. Such an interval is called *active alignment column interval* and will be defined below:

4.1. **Definition of (temporary) active alignment column intervals:**
for all positions s_i that are aligned to at least $T\,\%$ of the N database sequences, the interval $I(s_i)$ of active alignment columns is defined by the interquartile range of the aligned columns:

- let Q_1 be the lower and Q_3 the upper quartile of the sorted list of aligned columns, $x(s_i) = (x_1, \ldots, x_R)$; then the active alignment column interval for s_i is

$$I(s_i) = [Q_1, Q_3]$$

For all other sequence positions, the active alignment column interval is defined by the active intervals of preceding and successive positions:

- let $j < i$ be the preceding, most right, and $k > i$ the successive, most left query sequence position that is aligned to $\geq T/100 \cdot N$ sequences; then

$$I(s_i) = [I_l(s_j), I_r(s_k)]$$

The threshold T has been introduced to avoid active alignment column intervals defined on the basis of only a few pairwise BLAT alignments.

4.2. **Extension of active alignment column intervals:**
each active alignment column interval has to include at least one *active* alignment column; if not, it is extended in both directions until this requirement is fulfilled. Additionally, each interval is extended by a certain number $K \geq 0$ of *active* alignment columns

$$I(s_i) = [I_l(s_i) - K, I_r(s_i) + K]$$

4.3. **Concatenation of two successive active alignment column intervals:**
if a gap between the active alignment column intervals of two successive query

sequence positions is observed, i.e. $I_l(s_i) > I_r(s_{i-1}) + 1$, these two intervals are merged:

$$I_l(s_i) = I_l(s_{i-1}) \quad \text{and} \quad I_r(s_{i-1}) = I_r(s_i)$$

5. **Adjustment of active alignment column intervals to the jpHMM:** each active alignment column interval is extended such that each subtype has at least M match states, i.e. M consensus columns, within the interval.

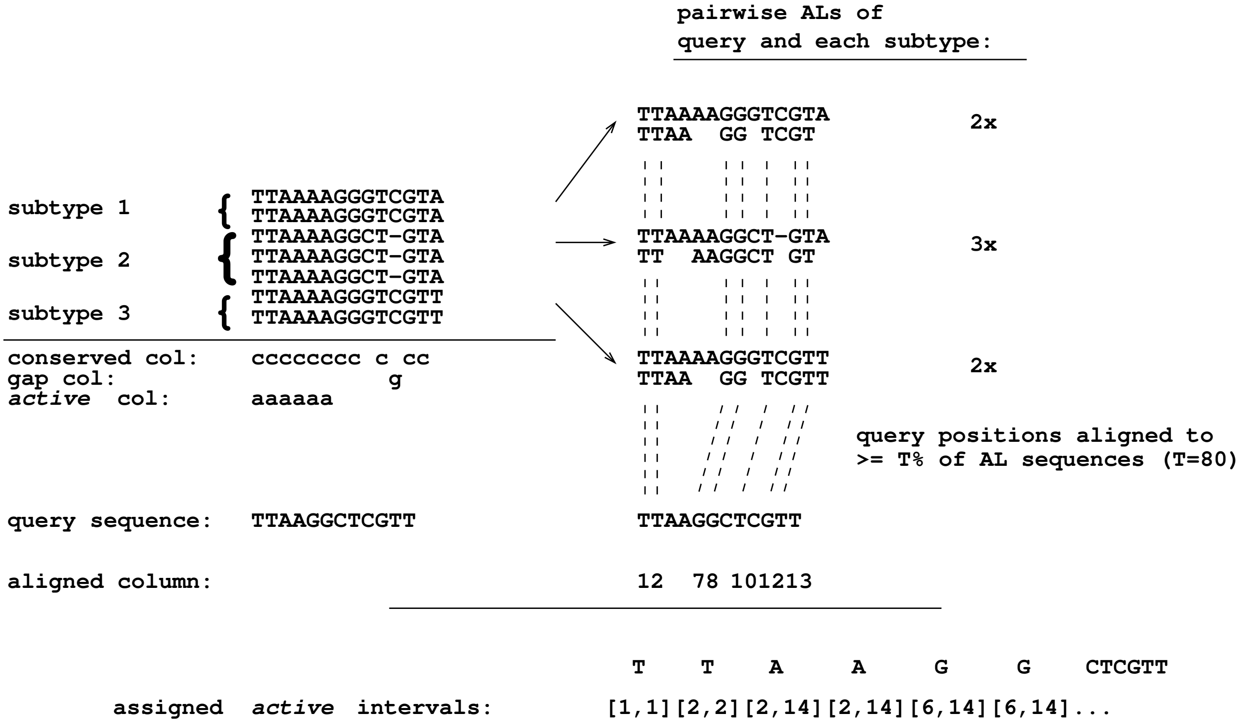


Figure 3.3: Definition of active columns in the pre-alignment with BLAT: In the left part of the figure in an alignment with three subtypes (each consisting of two or three identical sequences) conserved (c), gap (g) and *active* columns (col) are defined. For the given query sequence, in the right part a pairwise alignment (AL) to one sequence of each subtype is shown. For positions in the query sequence that are aligned to at least $T = 80\,\%$ of the alignment sequences, the aligned columns/intervals are determined (e.g. column 1 for position 1, column 7 for position 5, etc.). These intervals are extended such that they comprise at least a certain number (here 1) of *active* (see 'a' on the left) alignment columns. For positions that are not aligned to at least $T = 80\,\%$ of the alignment sequences, the *active* interval is determined by the *active* intervals for the preceding and successive position. The resulting *active* intervals are, for example, [1,1] for position 1, [2,2] for position 2, [2,14] for positions 3 and 4, etc..

3.2.2.2 Definition of active states in the Viterbi algorithm

For each query sequence position s_i, the active alignment column interval $I(s_i)$ is the region in the multiple sequence alignment s_i can be aligned to with jpHMM. This interval is active for all subtypes in the alignment. To align s_i to a column in the interval $I(s_i)$, s_i must be emitted by a state in the model corresponding to a column within this interval. Therefore, for each s_i the Viterbi path is forced to pass a state that is assigned to a column in $I(s_i)$.

Let $S = s_1, \ldots, s_l$ be a query sequence of length l and $I(s_i) := [I_l(s_i), I_r(s_i)], i = 1, \ldots, l$, the active alignment column interval for s_i. Let q be a state in the model and $col(q)$ the column in the multiple sequence alignment state q is assigned to.

Analogous to the beam-search algorithm (section 2.2.2, p. 21), for each query sequence position s_i, a modified Viterbi variable $\delta'_q(i) \leq \delta_q(i)$ is calculated and stored only for a subset $\mathcal{A}_i$ of states, namely those states q that are assigned to columns in the multiple sequence alignment lying within I_{s_i}. These states are called *active* states, and the set of active states for a certain position s_i is given by

$$\mathcal{A}_i := \{q \mid col(q) \in I(s_i) = [I_l(s_i), I_r(s_i)]\} . \tag{3.8}$$

The modified Viterbi variable of an *inactive* state is set to 0 and must not be saved. So, the modified Viterbi variable $\delta'_{i+1}(q)$ of a state q in step $i+1$ is only calculated for states $q \in \mathcal{A}_{i+1}$ that are successors of states in $\mathcal{A}_i$, or in $\mathcal{A}_{i+1}$ for a mute state q, respectively:

$$\delta'_{i+1}(q) = \begin{cases} \max\limits_{q' \in \mathcal{A}_{i+1}} \{\delta'_{i+1}(q') t_{q',q}\}, & \text{if q is a delete state,} \\ e_{q,s_{i+1}} \max\limits_{q' \in \mathcal{A}_i} \{\delta'_i(q') t_{q',q}\}, & \text{otherwise.} \end{cases} \tag{3.9}$$

Combination of BLAT and beam-search algorithm In certain cases the active alignment column interval for a query sequence position can be very large, leading to a slowdown of the runtime of the Viterbi algorithm and an increase in memory. Possible reasons are:

- variable regions in the multiple sequence alignment
- (sequencing) errors in the given multiple sequence alignment or the query sequence
- incorrect or missing BLAT alignments, e.g. in case of inserts, deletions or repeats in the query sequence
- sequences of other species or other non-matching sequences, uploaded erroneously

These aspects require a further restriction of the search space of the Viterbi algorithm based on the BLAT alignment. Therefore, the beam-search algorithm, described in section 2.2.2, is applied to the active alignment column intervals defined with BLAT:

Let $\mathcal{B}$ be the beam width, $0 < \mathcal{B} \ll 1$. The set of active states $\mathcal{A}_i$ at query sequence position s_i now comprises those states q with $col(q) \in I_{s_i}$, whose modified Viterbi value is not much lower than the optimal local solution

$$\delta_i^* = \max_q \delta_i'(q). \tag{3.10}$$

Thus, the set $\mathcal{A}_i$ of active states of step i is determined by

$$\mathcal{A}_i := \{q \,|\, col(q) \in I(s_i) = [I_l(s_i), I_r(s_i)] \text{ and } \delta_i'(q) \geq \mathcal{B}\delta_i^*\}, \quad 0 < \mathcal{B} \ll 1. \tag{3.11}$$

The modified Viterbi variable $\delta_q'(i+1)$ at step $i+1$ is now calculated by recursion (3.9), with $\mathcal{A}_i$ defined in (3.11). Let

$$B_i := \left[\min_{q \in \mathcal{A}_i} col(q), \max_{q \in \mathcal{A}_i} col(q)\right] \tag{3.12}$$

be the interval of columns in the multiple sequence alignment A for position s_i defined by the most left and the most right active state in $\mathcal{A}_i$. Within this interval, not all columns must be active. This depends on the modified Viterbi variables of the corresponding states. Also, the application of the beam-search algorithm to the active alignment column interval $I(s_i)$ can have varying degrees of influence on the different subtypes, resulting in active intervals of different lengths for different subtypes.

Problems The beam-search condition can have such a strong impact on the length of an interval $I(s_i)$, that no transition from a state in $\mathcal{A}_i$ to a state assigned to a column within the interval $I(s_{i+1})$ for the successive position s_{i+1} is possible. This is demonstrated by an example in Figure 3.4: if s_i is emitted by a state in $\mathcal{A}_i$ and $B(s_i) = [B_l(s_i), B_r(s_i)]$ is the interval in the alignment defined by the most left and the most right active state in $\mathcal{A}_i$, s_{i+1} can only be emitted by a successor state of a state in $\mathcal{A}_i$, i.e. a state assigned to a column within the interval $[B_l(s_i), B_r(s_i) + \max_{\mathcal{S}} k_{\mathcal{S}} + 1]$, with $k_{\mathcal{S}}$ being the number of non-consensus columns between $B_r(s_i)$ and the successive consensus column in subtype $\mathcal{S}$. The reason is:

1. each *insert* state is a successor of itself. So, if the *insert* state assigned to column $B_l(s_i)$ is active for position s_i, it can also be active for position s_{i+1} as a successor of itself. Thus, $B_l(s_i)$ can be the most left active column for a state emitting s_{i+1}.

2. a *match* state that is a successor of a state assigned to a column c, is assigned to the successive consensus column in one of the subtypes, which is at least $c+1$. If in a certain subtype $\mathcal{S}$ column c is followed by $k_\mathcal{S}$ non-consensus columns, the successive consensus column in this subtype is $c+k_\mathcal{S}+1$. Therefore, all *emitting* (match and insert) states that can be active at position s_{i+1} must be assigned to a column lower or equal to $B_r(s_i)+\max\limits_{\mathcal{S}} k_\mathcal{S}+1$.

Delete states do not emit any symbols. The Viterbi variable of a delete state at step i is calculated from the Viterbi variables of predecessors at the same step i (Equation (3.9)).

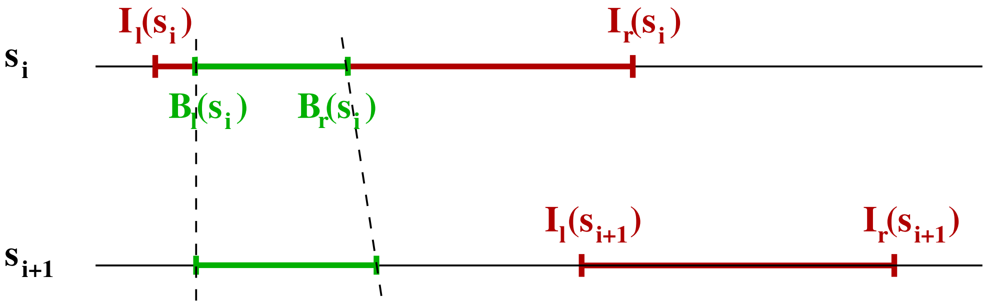


Figure 3.4: Example for non-overlapping alignment intervals determined with BLAT and the beam-search algorithm: For two successive query sequence positions s_i and s_{i+1}, intervals of active columns (red) are marked in the given alignment (represented by a vertical line). $I(s_i) := [I_l(s_i), I_r(s_i)]$ and $I(s_{i+1}) := [I_l(s_{i+1}), I_r(s_{i+1})]$ are the *active alignment column intervals* for s_i and s_{i+1}, resp., determined with BLAT (see Workflow 3.5). $B(s_i) := [B_l(s_i), B_r(s_i)]$ is the interval of active alignment columns within $I(s_i)$ resulting from the application of the beam-search algorithm. Only successors of states assigned to a column in $B(s_i)$ can be active for s_{i+1}, demonstrated by dotted lines. The interval of corresponding alignment columns is marked in green in the second line. Obviously, it does not overlap with $I(s_{i+1})$.

If the intervals $[B_l(s_i), B_r(s_i)+\max\limits_{\mathcal{S}} k_\mathcal{S}+1]$ and $[I_l(s_{i+1}), I_r(s_{i+1})]$ do not overlap (Figure 3.4), a transition from a state corresponding to a column in $B(s_i)$ to a state in $I(s_{i+1})$ is not possible. Therefore, in recursion (3.9) for all states q assigned to a column in the interval $I(s_{i+1})$ ($col(q) \in I(s_{i+1})$), the transition probability $t_{q',q}$ for all $q' \in \mathcal{A}_i$ is 0, and consequently

$$\mathcal{A}_{i+1} = \emptyset. \tag{3.13}$$

Since each path through the model generating a query sequence must start in the *begin* state and end in the *end* state, in this case no path can be found that emits the query sequence, and thus neither an alignment of the query sequence to the given multiple sequence alignment nor a recombination prediction is possible. Therefore, if such a "gap" between the active alignment intervals of two successive query sequence positions occurs due to the beam-search algorithm, the Viterbi variables are calculated based on the active alignment intervals of which the left boundary is determined by the beam-search condition (Algorithm 3.6, Recursion 2.2.). The right boundary is still determined by the right boundary of $I(s_i)$, if this interval is located to the right of the interval determined by the beam-search condition. Otherwise, the beam-search algorithm is applied disregarding any right boundary.

Algorithm 3.6 (Viterbi including beam-search algorithm on BLAT intervals)
Let A be a multiple sequence alignment with m columns and $S = s_1, \ldots, s_l$ a query sequence of length l. Let $0 < \mathcal{B} \ll 1$ be the beam-width in the beam-search algorithm.

1. Initialization:

   1. Let B be the *begin* state with $\delta'_0(B) := 1 \quad (\Rightarrow \quad \delta^*_0 = 1)$;

      and $I_0 := [I_l(s_1), I_r(s_1)]$ the active alignment column interval for *deletes* in front of s_1
   2. Viterbi recursion for *delete* states q with $col(q) \in I_0$,

      sorted by the assigned columns in A, starting with the left-most state:

      $$\delta'_0(q) = \max_{q' : q' = B \text{ or } col(q') \in I_0} \{\delta'_0(q') t_{q',q}\}$$

      $$\Rightarrow \mathcal{A}_0 := \{B\} \cup \{q \mid q \text{ is delete state with } col(q) \in I_0 \text{ and } \delta'_0(q) \geq \mathcal{B}\delta^*_0\}$$

2. Recursion:

```
FOR $i = 1, \dots, l$,
    1. Let $I(s_i) = [I_l(s_i), I_r(s_i)]$ be the active alignment column interval for $s_i$,
       and $B(s_{i-1}) = [B_l(s_{i-1}), B_r(s_{i-1})] = [\min_{q \in \mathcal{A}_{i-1}} col(q), \max_{q \in \mathcal{A}_{i-1}} col(q)]$
       and $succ_{\mathcal{S}}(B_r(s_{i-1}))$ the successive consensus column of $B_r(s_{i-1})$ in subtype $\mathcal{S}$ and
       $B_r^* := \max_{\text{subtype } \mathcal{S}} succ_{\{\mathcal{S}, B_r(s_{i-1})\}}$ the maximum of these columns;
        $\Rightarrow$ each successor of a state $q \in \mathcal{A}_{i-1}$ is assigned to a column in $B^* := [B_l(s_{i-1}), B_r^*]$
    2. IF $(B^* \cap I(s_i) == \emptyset$ (the intervals do not overlap) )
        $I_l(s_i) = B_l(s_{i-1})$ and $I_r(s_i) = \begin{cases} I_r(s_i), & \text{if } I_r(s_i) \geq B_l(s_{i-1}) \\ m, & \text{otherwise} \end{cases}$
    3. Viterbi recursion for match and insert states $q$ with $col(q) \in I(s_i)$ :
        $\delta'_i(q) = e_{q,s_i} \max_{q' \in \mathcal{A}_{i-1}} \{\delta'_{i-1}(q') t_{q',q}\}$
          $\delta^*_i = \max_q \delta'_i(q)$
       $\Rightarrow$ temporary set of active states $\mathcal{A}_i = \{q \mid col(q) \in I(s_i)$ and $\delta'_i(q) \geq \mathcal{B}\delta^*_i\}$
    4. Viterbi recursion for delete states $q$ with $col(q) \in I(s_i)$,
       sorted by the assigned columns in $A$, starting with the most left state:
        $\delta'_i(q) = \max_{q' \in \mathcal{A}_i} \{\delta'_i(q') t_{q',q}\}$   (since $q' \in \mathcal{A}_i$ and $t_{q',q} \leq 0$   $\delta^*_i$ does not change)
        IF $(\delta'_i(q) \geq \mathcal{B}\delta^*_i)$
            $\mathcal{A}_i = \mathcal{A}_i \cup \{q\}$
```

3.2.2.3 Forward and Backward variables and posterior probabilities

As described in sections 3.1.1 and 3.1.2, the Forward and Backward variables, and thus the posterior probabilities, are only calculated for states q in the respective set $\mathcal{A}_i$ of active states, here defined by the beam-search algorithm on the basis of the active alignment columns intervals defined with BLAT (Algorithm 3.6).

3.3 A modified jpHMM architecture with mute jump states

The jpHMM architecture is obviously very complex which is a consequence of the large number of transitions, i.e. jumps, between different profile HMMs. Usually, for each column, a jump from each profile HMM to any other profile HMM in the model is possible. Thus, the number of jumps in each column is quadratic in terms of the number of profile HMMs, i.e. the number of subtypes (Figure 3.5).

In the current applications of jpHMM (to HIV-1 and HBV), the number of subtypes is very limited (14 and 8, resp., see section 5.1) and therefore recombination prediction with jpHMM can be carried out with an appropriate amount of memory and time in these cases. But, it is also conceivable to apply jpHMM to larger sets of subtypes. Possible applications are recombination detection with regard to subgenotypes (e.g. up to now 35 HBV subgenotypes have been identified) or CRFs, and chimera detection in 16S rRNA sequences. Comparative analysis of 16S rRNA is the 'gold-standard' for bacterial classification and identification ([25], reviewed in [105]) but it can be hampered by chimeric artifacts of different species generated during PCR amplification [51, 39] when mixed bacterial populations are studied, for example in metagenomic projects. To avoid distortions in bacteria population studies and false identification of chimeric sequences as novel taxa [28] it is important to detect these chimeric sequences. For this purpose, a subdivision of known 16S rRNA sequences into subtypes, i.e. into different phyla, families etc., is necessary. Depending on the examined taxonomic rank, the number of subtypes can be very large, even comprising several hundreds. Such a large number of subtypes would result in millions of possible jumps (depending on the length of the sequences) in the current model and the storage of all these jumps alone would already require several hundred megabyte. Therefore, a modification of the current jpHMM architecture to reduce the number of jumps in the model is introduced.

3.3.1 Number of jumps in a (simple) jpHMM

For clarity, insert and delete states are neglected in this section and it is assumed that each column in each subtype is a consensus column and that the model only consists of match states. Then, from each match state, a jump to the match state in the successive column of all subtypes is possible. Therefore, in a jpHMM with k subtypes for each column, $k-1$ jumps (plus a transition within each profile HMM) exist, resulting in $k \cdot (k-1)$ jumps per column. In Figure 3.5, all possible jumps and transitions between two successive columns are plotted. For an alignment with l columns, this results in $(l-1) \cdot k \cdot (k-1)$ jumps ($l-1$ since no jumps are possible for the last column).

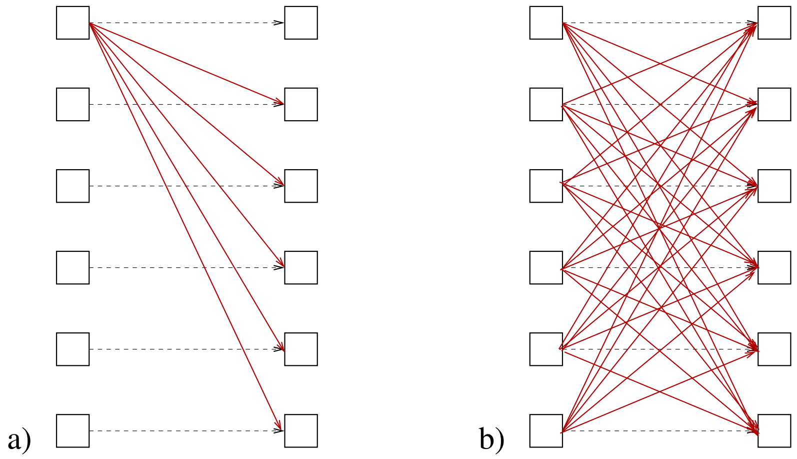


Figure 3.5: Jumps (in red) between two columns in a jpHMM with six subtypes consisting only of match states and consensus columns. In a), all jumps for one match state are shown and in b) the jumps for the match states of all subtypes. Broken arrows mark transitions within profile HMMs.

3.3.2 A modified (simple) jpHMM with a reduced number of jumps

The number of jumps can be reduced enormously by the introduction of an additional *mute jump state* for each column. This mute jump state is labeled D_M. A mute jump state D_M is assigned to each column in the alignment . Replacing the original jumps by jumps from each match state in a certain column to the corresponding mute jump state, and from the mute jump state to all match states in the successive column (Figure 3.6), the number of jumps per column in a jpHMM with k subtypes is reduced to $2k$. Thus, the number of jumps in a jpHMM for an alignment with l columns is reduced to $2k(l-1)$. Obviously, this model also allows "jumps" from one subtype to the same subtype via D_M. But, since the transition probabilities within each profile HMM are usually much higher than the jump probabilities these "jumps" do not pose a problem. In this section, this new model with a linear number of jumps in terms of the number of subtypes is named *jpHMM_linear* and, for comparison, the original model *jpHMM_orig*.

In Table 3.1, a comparison of the number of jumps per columns is shown for different numbers of subtypes. The chosen numbers correspond to the number of subtypes in current jpHMM applications like HBV (8 subtypes) and HIV-1 (14 subtypes). 100 subtypes could be an appropriate number of subtypes for an application of jpHMM to chimera detection in 16S rRNA. For example, for a 16S rRNA alignment with 100 subfamilies, the original model contains $9,900$ jumps per column, whereas the number of jumps in the new model is only a fraction of this number, namely 200 jumps per column, which is about 2% of the original number of jumps. For an alignment with $1,500$ columns (which is the length of

16S rRNA sequences), this means that the number of jumps in the whole model can be reduced from 15 million to only $300,000$.

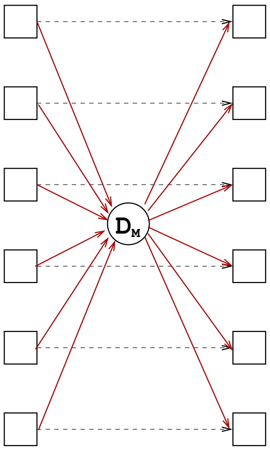


Figure 3.6: Jumps (in red) between two columns in the new model (jpHMM_linear) with six subtypes consisting only of match states and consensus columns. The introduction of a mute jump state D_M reduces the number of jumps in a jpHMM from $k \cdot (k-1)$ to $2k$ per column if k is the number of subtypes in the model. Broken arrows mark transitions within profile HMMs.

Table 3.1: Comparison of the number of jumps per column in the original model (jpHMM_orig, column 3) and the new model (jpHMM_linear, column 4) for different example species (column 1) with different numbers k (column 2) of subtypes (ST).

		Number of jumps per column in	
Species	# ST	jpHMM_orig	jpHMM_linear
e.g.	k	$k(k-1)$	$2k$
HBV	8	56	16
HIV-1	14	182	28
16S rRNA	100	9,900	200

3.3.3 A modified jpHMM including all state types

In the previous sections, a simple model only consisting of match states has been examined. Now it will be described how jumps between other state types in the original model can be modeled in the new model (jpHMM_linear) using mute jump states. In the original model, in addition to the jumps from match to match states jumps from delete and insert states to match states and jumps from match to delete states are possible [87]. Jumps from insert to delete states and vice versa were not allowed to reduce the number of transitions within the model and to follow the Plan 7 architecture of HMMER [31, 21] .

3.3.3.1 Jumps from insert and delete states to match states

Jumps from delete and insert states to match states can be modeled by a jump from each delete and each insert state in a certain column to the corresponding mute jump state D_M which is a predecessor of the match states in the successive column. Therefore, for each column, $2k$ jumps from the delete and insert states to D_M are necessary in addition to the k jumps from the match states to D_M. This results in $3k + k$ jumps per column, $3k$ jumps from match, insert and delete states to D_M and k jumps from D_M to the successive match states.

3.3.3.2 Jumps from match to delete states

Jumps from match to delete states can also be modeled by the introduced mute jump state D_M but if a jump from the mute jump state D_M to a delete state is allowed implicitly jumps from insert to delete states are allowed. In principle, such transitions between insert and delete states are also possible, but in order to model only transitions and jumps that were possible in the original model the introduction of a second mute jump state is necessary to model jumps from match to delete states. This mute jump state is labeled D_D. A mute jump state D_D is assigned to each column in the alignment. From each match state a jump to D_D in the corresponding column and a jump from D_D to all delete states in the successive column is allowed. Thus, additionally, for each column k jumps from the match states to D_D and k jumps from D_D to the delete states in the successive column (Figure 3.7) are necessary to model jumps from match to delete states.

3.3.3.3 Number of jumps in the modified jpHMM including all state types

In total, the new model (jpHMM_linear) with k subtypes contains $4k$ jumps via D_M and $2k$ jumps via D_D per column. In the original model, the number of jumps is composed of $3k(k-1)$ jumps to match states and $k(k-1)$ jumps to delete states per column. Thus, the number of jumps per column in the new model is $6k$ compared to $4k(k-1)$ jumps per column in the original model.

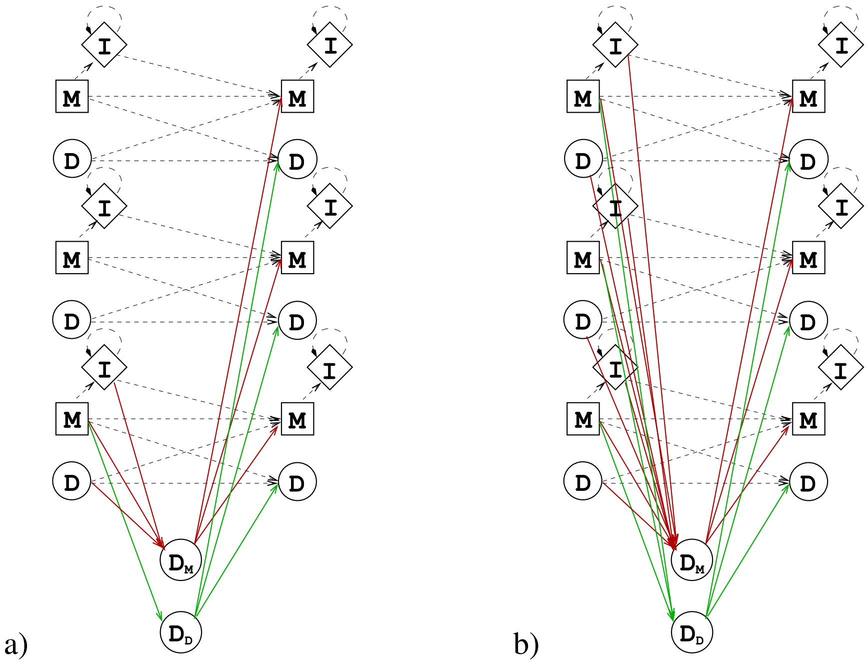


Figure 3.7: Jumps between two successive columns in the new model (jpHMM_linear) with three subtypes including all state types. In a), all possible jumps from one subtype to all other subtypes are shown. Jumps regarding D_M, i.e. jumps from all state types to match states, are shown in red. Jumps regarding D_D, i.e. jumps from match to delete states, are shown in green. In b), all possible jumps between all subtypes are shown. Broken arrows mark transitions within profile HMMs.

3.3.4 Memory required for storing the jumps in a jpHMM

In the current C++ implementation of a jpHMM, for each jump (and transition), a pointer to the state where the jump is initiated, a pointer to the state the jump is directed to, and the probability of this jump have to be stored. The storage needed for a pointer is 4 bytes. Using double precision numbers, 8 bytes are needed to store the transition probability. Thus, each jump requires 16 bytes. On the basis of the number of jumps calculated for different example data (Table 3.1) the memory required to store all jumps (including jumps regarding match, insert and delete states) in the original (jpHMM_orig) and the new model (jpHMM_linear) is compared in Table 3.2. As one can see, in the original model about 900 MB are needed to store only the jumps. With the introduction of mute jump states the required memory can be reduced by 98% to only 18.3 MB in the new model.

Table 3.2: Comparison of the memory (in MB) required to store the jumps in a model only consisting of match states (column 4 and 6) and to store all jumps for all state types (column 5 and 7) in the original model (jpHMM_orig, column 4 and 5) with the corresponding memory required in the new model (jpHMM_linear, column 6 and 7). This comparison is provided for different example species (column 1) with different numbers k (column 2) of subtypes (ST) and length l (column 3). Additionally, the required memory for jumps in both models is compared by indicating the amount of reduction (in %, column 8) achieved with the new model compared to the original model.

Species	# ST	Length	Memory required (in MB) for jumps in		
	k	l	jpHMM_orig	jpHMM_linear	Reduction
HBV	8	3,200	10.93	3.12	71.45 %
HIV-1	14	10,000	111.07	17.09	84.61 %
16S rRNA	100	1,500	905.77	18.30	97.98 %

3.3.5 Complexity of the Viterbi algorithm

The Viterbi algorithm of a jpHMM has a complexity of $O(lkn)$ in time, if l is the length of the query sequence, k the number of subtypes and n the number of states in the model. As described in section 2.2.2 (p. 21), for a query sequence and an alignment of equal length, namely l, the Viterbi algorithm has a complexity of $O(l^2k^2)$ in time.

Explicitly, the Viterbi variable of a certain state q at a certain position of a query sequence is calculated by an iteration over all predecessor states of q (section 2.2.2.2, p. 20). The number of predecessors differs for different state types: A *match* state has three predecessors in its corresponding profile HMM and three predecessors (match, insert and delete state) in each of the other subtypes. Thus, a match state has $3k$ predecessors. An *insert* and a *delete* state has two predecessors in the corresponding profile HMM (match and insert or match and delete, resp.). Additionally, a delete state has one predecessor (match) in each of the other subtypes. Thus, in total, $(4k + 3)l$ iterations are needed to calculate the Viterbi variables for all states at a certain query sequence position, and $(4k^2 + 3k)l^2$ iterations for the calculation of the whole Viterbi table.

Obviously this number is reduced enormously by the introduction of mute jump states. In addition to the predecessors in the corresponding profile HMM described above, in the new model (jpHMM_linear), each *match* state has one mute D_M jump state as predecessor and a *delete* state has a mute D_D jump state as predecessor (Figure 3.7). Each mute D_M jump state has three and each mute D_D jump state has one predecessor in each subtype.

Thus, the number of iterations needed to calculate the Viterbi variables of all states for a certain query sequence position is $13kl$, and the number of iterations the Viterbi algorithm comprises in total is $13kl^2$. So, the complexity of the Viterbi algorithm in time can be reduced to $O(kl^2)$ in the new model. In Table 3.3, a comparison of the number of iterations in the Viterbi algorithm in the original model (jpHMM_orig) and the new model (jpHMM_linear) is given for different species with different number of subtypes. As it can be seen, for a model with eight genotypes, the number of iterations is more than halved using the new model. For a model with 14 subtypes, it is reduced to a quarter of the number of iterations in the original model, and for a model with 100 subtypes, it is even reduced to about 3.2% of the original amount. An explicit comparison of the runtime of the two models for several alignments with different lengths and different number of subtypes is carried out in the Results chapter (section 5.5, p. 101).

Table 3.3: Comparison of the number of iterations in the Viterbi algorithm for each query sequence position (qp) in the original model (jpHMM_orig, column 3) and the new model (jpHMM_linear, column 4) for different example species (column 1) with different numbers k (column 2) of subtypes (ST).

		Number of iterations in Viterbi for each qp	
Species	# ST k	jpHMM_orig $(4k^2+3k)$	jpHMM_linear $13k$
HBV	8	280	104
HIV-1	14	826	182
16S rRNA	100	$40,300$	$1,300$

3.3.6 Jumps in case of non-consensus columns

In the previous sections, we assumed that each column in the alignment is a consensus column which means that each column in the alignment is modeled as a match state in each subtype. For this case, all jumps in the original model can be modeled in the new model by introducing mute jump states D_M and D_D. But this is different for alignments including non-consensus columns. In this case, not every column in the alignment is represented in every subtype as a match state and the picture of jumps is more complicated. In Figure 3.8, an example jpHMM with six subtypes and match states modeled at different columns is given. For the states in the first column, all possible jumps are shown, using different colors for different subtypes.

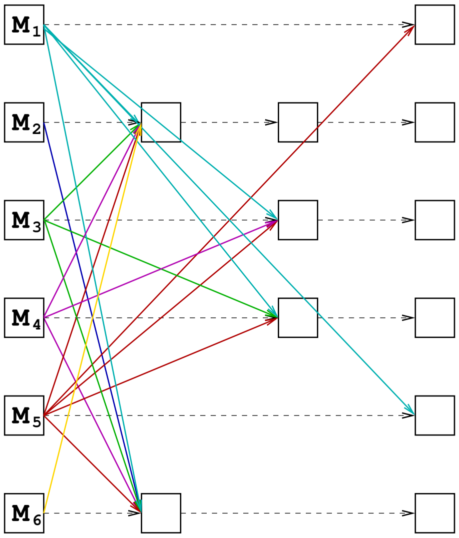


Figure 3.8: Jumps in the original model with six subtypes are shown in the case of gaps in the alignment. Only match states are shown. Broken arrows mark transitions within profile HMMs. For the state in the first column, all possible jumps are depicted. For different subtypes different colors have been chosen.

Originally, several rules have been imposed for jumps between subtypes to reduce the number of possible transitions in the model as well as to ensure that no insertions or deletions are introduced during a jump without using insert or delete states [87]. Besides the rule that jumps are only allowed from match, insert and delete states to match states, and from match states to delete states, the following two rules are valid: From a state in column i in subtype $\mathcal{S}_u$, named $T_{u,i}$, a jump to another subtype $\mathcal{S}_v$

r1 is possible to the *left-most* state in $\mathcal{S}_v$ that is *strictly to the right* of $T_{u,i}$

r2 is not possible to a state in $\mathcal{S}_v$ that is to the right of the successive consensus column of i in $\mathcal{S}_u$ (this is usually $i+1$ in case of non-gap columns, i.e. to the right of $T_{u,i}$).

In case of non-consensus columns, in different subtypes this can result in only a few possible jumps for certain subtypes as it can be seen in Figure 3.8: For example, from state M_2 in the second subtype only *one* jump, namely a jump to the sixth subtype, is possible. For jumps from this state to all other subtypes, rule r2 is not satisfied.

3.3.6.1 Adaption to a jpHMM with a modified architecture

In the previous sections, jumps in the new model were modeled by a jump to one of the mute jump states D_M or D_D followed by a jump from the mute jump state to a state in the successive column. In case that not every column in the alignment is a consensus column in every subtype, with this imprecise definition only the jumps modeled in Figure 3.9 are possible. In this figure, the blue arrow labeled with a question mark marks a jump that is possible in the original model but not in the new model (jpHMM_linear). Thus, to model jumps between subtypes in case of a model containing non-consensus columns in different subtypes the following rule is imposed:

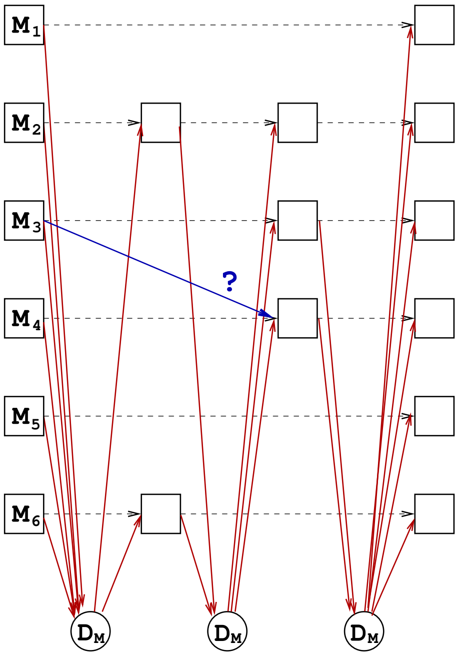


Figure 3.9: Jumps in the new model (jpHMM_linear) in case of gaps, if from the mute jump state D_M only jumps to the next column are possible. Only match states are shown. Broken arrows mark transitions within profile HMMs. Red arrows show all possible jumps. The blue arrow labeled with a question mark marks a jump that is possible in the original model but not in the new model in its current form.

r3 for each column i, from the corresponding mute jump states D_M and D_D a jump is possible to the *left-most* state in each subtype that is *strictly to the right* of i.

Thus, in the new model, a jump from each subtype (in case that this column is represented by a match state in this subtype) to *all* other subtypes is possible for each column. In Figure 3.10, all jumps corresponding to the first column that obey rule r3 are shown with red arrows. The blue arrows, labeled with an exclamation mark, do not depict real jumps, they exemplary point out jumps modeled by the mute jump state D_M that were not possible in the original model. As described above in Figure 3.8, in the original model from state M_2 in the second subtype, only *one* jump to the sixth subtype is possible due to rule r2.

Now, in the new model, four additional jumps, namely to the first, the third, the fourth and the fifth subtype, are possible.

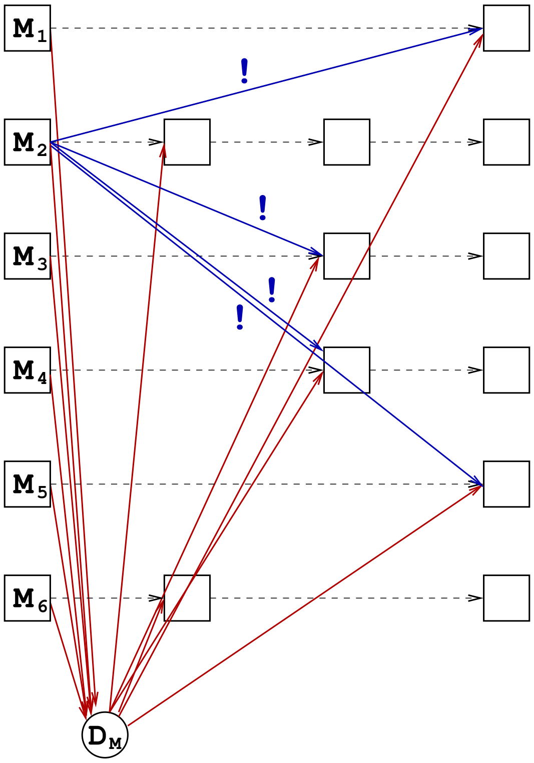


Figure 3.10: Jumps in the new model (jpHMM_linear) in case of gaps. Only match states are shown. Broken arrows mark transitions within profile HMMs. Red arrows show all possible jumps. Blue arrows labeled with an exclamation mark show jumps that are possible (indirectly) in the new model but not in the original model.

It is possible to build a model that models exactly the jumps which are allowed in the original model, but not with an appropriate amount of jump states. For example, for state M_1 in the first subtype, a jump to all other subtypes is possible in contrast to M_2 in the second subtype where only one jump to the sixth subtype is possible (Figure 3.8). These jumps must be distinguished which is only possible with different types of jump states. The number of these additional jump states depends on the length of the respective gap. But, since the aim of this project is to reduce the complexity of the model by decreasing the number of jumps, such a model is not desirable.

In the Results chapter (section 5.5, p. 101), the new model (jpHMM_linear) is compared to the original model (jpHMM_orig) in terms of the location of predicted breakpoint positions and especially in terms of runtime.

3.4 A jpHMM for recombination prediction in circular genomes

The genomes of most viruses such as HIV and HCV are linear but a multitude of viruses with a circular genome (like many bacteria) exists. This includes RNA as well as DNA viruses. Besides HBV (partially double-stranded DNA), examples are the hepatitis delta virus (HDV) (single-stranded RNA) and the human papillomavirus (HPV) (double-stranded DNA). Naturally, sequences of viruses with a circular genome are uploaded to GenBank [8] in a *linear* form. But, in contrast to real linear genomes, in these sequences, dependencies between the nucleotides at the 5' and the 3' end of the sequences exist. Such dependencies cannot be modeled with a *linear* model like jpHMM since they require a transition from the end of the model back to the beginning. Allowing such transitions in jpHMM, endless paths through the model are possible. Thus, breakpoints at or near the 5' or 3' end of the sequence cannot be detected with jpHMM either since they require a jump from the subtype predicted for the last sequence position back to the subtype predicted for the first position, i.e. a jump back to the beginning of the model. Also, such jumps cannot be modeled by the Viterbi algorithm due to its structure: In the case that the most probable path through the model generating a certain sequence has been determined and the predicted parental subtypes for the first and the last sequence position are different, the cost for a jump between the two subtypes should be added by multiplying the probability of the most probable path with the jump probability. But in this way, the probability of the path is decreased and the structure of the Viterbi algorithm does not allow an answer to the upcoming question whether the former most probable path is still the most probable path or if the probability of other paths is higher now. Additionally, using jpHMM in its current form to predict recombinations in viruses with a circular genome, can have the following implications:

Breakpoints close to the sequence ends are not detected A recombination breakpoint close to the chosen origin for sequence coordinates (see Figure 3.11) is not detected with jpHMM if the probability of a jump to the correct subtype is too low compared to a small number (depending on the distance of the true breakpoint to the chosen origin) of mismatches to the incorrect subtype. As a consequence,

1. the breakpoint is implicitly predicted at the chosen origin for the sequence coordinates. This, of course, only holds for nearly full-length genomes.
2. as parental subtype for the segment between the real breakpoint and the chosen origin for the sequence coordinates the subtype of the preceding segment is (incorrectly)

predicted.

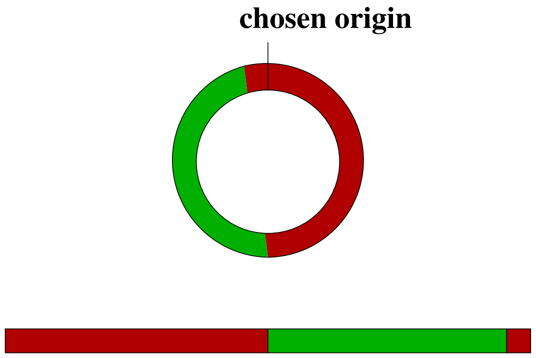


Figure 3.11: Example for the location of breakpoints in the annotated linear sequence (bottom) depending on the chosen origin for sequence coordinates in the circular genome (top). A recombination of two subtypes (green and red) with two breakpoints is shown. One breakpoint is close to the chosen origin for sequence coordinates and thus close to the end of the linearized sequence.

Erroneously predicted recombination segments at the sequence ends In circular genomes, usually, two breakpoints are necessary to introduce a recombination segment in the recombination prediction. But, a recombination segment at the end of a linearized sequence requires the introduction of only one breakpoint. I.e. for the cost of only one jump, a recombination segment can be introduced. Therefore,

1. for circular genomes, the cost of introducing a recombination segment strongly depends on the chosen origin for the sequence coordinates leading to a bias towards recombination segments at the sequence ends;
2. for a relatively low cost, depending on the similarity of the subtypes, a recombination segment can be introduced erroneously, which would not happen if the model was forced to make a second jump back to the first subtype, i.e. to add the cost for a second jump.

To summarize, on the one hand, the application of jpHMM to linearized sequences of circular genomes can lead to a shift of the real breakpoint to the chosen origin of the sequence coordinates to avoid the cost for a jump. On the other hand it can lead to the introduction of a recombination segment at the 5' or 3' end of the sequence, since only one breakpoint is required for a recombination segment at the end of the genome instead of two. Therefore, it is not possible to apply jpHMM in its current form to viruses with a circular genome. This, of course, only holds for full-length sequences. The 5' and the 3' end of a fragmental sequence do not correspond to the same position in the circular genome and, consequently, no dependencies between both sequence ends must be taken into account in the recombination prediction.

In the following section, an extension of the model is presented that allows for recombination detection in viruses with circular genomes. To our knowledge, this is the first approach for recombination detection in circular genomes that explicitly takes into account the circularity of the genome.

3.4.1 Extension of query sequence and multiple sequence alignment

To overcome the problems of an accurate recombination prediction at the 5' and 3' end of full-length sequences mentioned above, each full-length query sequence is extended: The prefix of a certain length of the query sequence is copied and concatenated to the end of the query sequence, and, analogously, the suffix of the original sequence is copied and concatenated to the beginning of the sequence (Figure 3.12). Naturally, instead of an extension of the query sequence at the 5' and the 3' end of the sequence, a (longer) extension of the query sequence at only one end of the sequence is conceivable. But to simplify the post-processing, i.e. the determination of the recombination prediction for the original query sequence, an extension at both ends of the sequence is preferred.

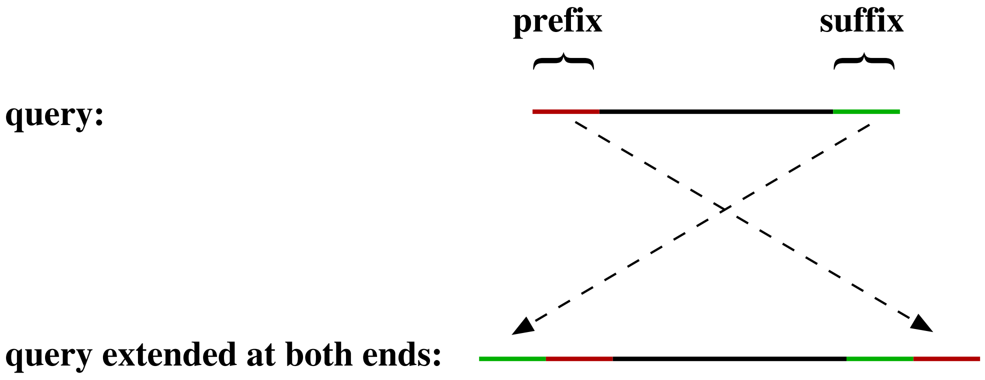


Figure 3.12: For circular genomes, each full-length query sequence is extended by copying and concatenating the prefix of a certain length of the query sequence (red) to the end of the query, and, analogously, copying and concatenating the suffix of the original sequence (green) to the beginning of the sequence. Both processes are demonstrated by dashed arrows.

By this extension of the query sequence, depending on the length of the copied regions, breakpoints close to the original 5' and 3' end of the sequence must be explicitly predicted and the cost for introducing recombination segments in the original sequence does not depend on the chosen origin for the sequence coordinates. Thus, a recombination prediction for such extended sequences is not biased against recombination breakpoints near the chosen origin.

To enable an alignment of an extended, full-length query sequence to the given multiple sequence alignment, it is necessary to extend the multiple alignment at the 5' and the 3' end as well. Even for full-length sequences with a different chosen origin for the sequence coordinates than that of the sequences in the alignment (EcoR1 cleavage site for HBV, section 5.1.2.2), such an extension is sufficient. It is necessary to just cut the sequence at the respective genomic position and concatenate both parts in reverse order before applying jpHMM. But for fragmental sequences, such a reordering is not possible since the 5' and the 3' end of the sequence do not correspond to the same position in the genome. Therefore, the given multiple sequence alignment is extended in the following way: It is duplicated, i.e. an identical copy of the alignment is concatenated to its end, and a prefix of the alignment, that has at least the length of the extended region in the query sequences, is copied and concatenated to the end of the duplicated alignment (Figure 3.13). A jpHMM built on the basis of this extended alignment allows the alignment of fragmental as well as extended, full-length sequences to the multiple sequence alignment, regardless of the chosen origin for the sequences' coordinates.

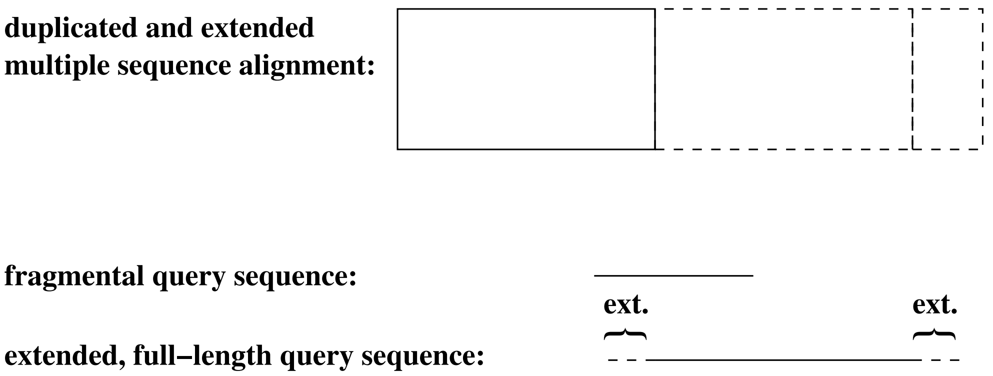

duplicated and extended multiple sequence alignment:
fragmental query sequence:
ext.
ext.
extended, full–length query sequence:

Figure 3.13: For circular genomes, the given multiple sequence alignment (roughly sketched by the black rectangle) is duplicated and extended by copying a prefix of the alignment to the end (dashed rectangles). By this extension, alignments of fragmental as well as extended, full-length query sequences independently from any chosen origin for sequence coordinates are allowed.

3.4.2 Pre-alignment of extended query to extended alignment with BLAT

Applying jpHMM to an extended query sequence and to an extended multiple sequence alignment in its original form, i.e. with no further restrictions of the search space than the beam-search algorithm, would result in an unnecessary waste of runtime and memory, since each query sequence can be nearly completely aligned to two different regions in the given multiple sequence alignment. For this reason, it is useful to define the location of the given extended query sequence in the multiple sequence alignment before jpHMM is applied. As described in section 3.2.2, for this purpose each (extended) query sequence is initially aligned to the extended multiple sequence alignment with BLAT, resulting in an interval of active alignment columns for each sequence position.

Problem Fragmental sequences can be aligned to two regions in the extended multiple sequence alignment. In general, BLAT produces more than one pairwise alignment of a query sequence and a database sequence, which have different scores. Usually, the alignment with the highest score is chosen. But, if, for example, for half of the database sequences the alignment of the query sequence to the first part of the extended alignment has the highest score, and for the other half the alignment to the second part of the extended alignment, the resulting intervals of active alignment columns comprise both parts and are thus very large. This problem is solved by forcing the algorithm to use always the most left alignment of the query sequence and the database sequence, if the respective score differs from the highest score only by a certain value. The chosen thresholds are given in chapter 5, section 5.7.

3.4.3 Recombination prediction with jpHMM based on pre-alignment

For circular genomes, the jpHMM is built on the extended multiple sequence alignment. The active alignment columns defined with BLAT for each query sequence position represent the *active* states in the jpHMM that are allowed to emit the corresponding query sequence position. Based on these active states, the Viterbi path of the (extended) query sequence through the model is determined with the Viterbi algorithm, including the beam-search algorithm as described in Algorithm 3.6. So, for nearly full-length query sequences, a recombination prediction is made for the extended query sequence (for fragments the recombination is predicted for the original query sequence). This *extended recombination prediction* is cut at the original start and end position of the query sequence such that the output comprises only the predicted recombination for the original query sequence.

If different subtypes are predicted at both ends of the sequence, the chosen origin for the sequence coordinates represents a breakpoint.

3.4.4 Uncertainty regions and breakpoint intervals in circular genomes

For each position in the (extended) query sequence, the posterior probabilities of all subtypes are calculated with the Forward and Backward algorithm (section 3.1.2), on the basis of the active states defined by the BLAT alignments and the beam-search algorithm. Based on these probabilities, uncertainty regions in the predicted recombination (for the original, not extended query sequence) and breakpoint intervals are determined as described in section 3.1.3, but with one difference at both ends of the sequence:

In linear genomes, the extent of a breakpoint interval around a certain predicted breakpoint is limited by the preceding and successive breakpoint, which can be, if necessary, the first or last position, respectively, in the query sequence. In linearized, full-length sequences of circular genomes, the 5' and 3' end represent an artificial origin for the sequence coordinates that should not restrict the extent of a breakpoint interval. For example, for the left-most predicted breakpoint in the given linearized sequence, the preceding breakpoint can either be the chosen origin for the sequence coordinates if the two subtypes predicted at the sequence ends differ, or, due to the original circularity of the genome, the right-most predicted breakpoint in the given linearized sequence if the two subtypes agree. Therefore, for the definition of a breakpoint interval around one of the breakpoints predicted most closely to one of the sequence ends, it is necessary to take into account the behavior of the posterior probabilities at the other end of the sequence as well if the predicted subtypes at both sequence ends agree.

Thus, it can happen that regions in a query sequence that are defined as uncertainty regions in linear genomes, are predicted as breakpoint intervals in circular genomes. For example, if the posterior probability of the subtype predicted to the left of the left-most predicted breakpoint is lower than the threshold t_{BPI} at all positions upstream of this breakpoint, the whole region is marked as uncertain in linear genomes. In circular genomes, this region is defined as a breakpoint interval, if the posterior probability of the respective subtype reaches the threshold at any position at the other end of the sequence, downstream of the right-most predicted breakpoint.

Additionally, breakpoints defined by the chosen origin for the sequence coordinates if the predicted subtypes at both ends of the sequence differ, have be taken into account for the definition of breakpoint intervals.

3.4.5 Workflow for the recombination prediction in circular genomes

The following workflow gives an overview of the recombination prediction with jpHMM in circular genomes, described in detail in the previous sections.

Workflow 3.7

1. **Extension of the given multiple sequence alignment**
   by duplication and concatenation of its prefix to the end of the alignment (section 3.4.1)

2. **Extension of each nearly full-length query sequence**
   by concatenation of its prefix and suffix to the end and the beginning of the sequence respectively (section 3.4.1)

3. **Alignment of each (extended) query sequence to the extended alignment with BLAT**

4. **Definition of active alignment column intervals for each sequence position**

5. **Definition of active states in the model for each sequence position**

6. **Recombination prediction for each (extended) query sequence**
   on the basis of the active states defined for each query sequence position

7. **Determination of uncertainty regions and breakpoint intervals**
   on the basis of the calculated posterior probabilities taking into account the circularity of full-length sequences (section 3.4.3)

8. **Graphical visualization and output of the predicted recombination including uncertainty regions and breakpoint intervals in a circular form**
   using Circos [44], a software package for visualizing data and information in a circular layout (e.g. Figure 4.2). For further information, e.g. alternative parental subtypes in uncertainty regions, the posterior probabilities are plotted and the location of genes in the reference genome is presented.

Chapter 4

Implementation

The main jpHMM source code is written in C++ [94] comprising several classes such as for reading the input alignment and query sequences, generating a profile HMM for each subtype, combining the profile HMMs to one jpHMM, searching the Viterbi path and calculating the posterior probabilities. The program was compiled under Linux with the g++ (gcc, version 4.4) compiler. For the application of jpHMM to query sequences pre-aligned with BLAT [37], a Perl script was written that uses the BLAT output to determine active alignment columns for each query sequence position (section 3.2.2, p. 37). Uncertainty regions and breakpoint intervals in the predicted recombination are determined with a Perl script evaluating the posterior probabilities of the subtypes. The posterior probabilities of all subtypes at each query sequence position are plotted with a script written in the programming language R [77] using the 'seqinr' package [14]. The predicted recombination for HBV genomes is visualized in a circular form using the software package Circos [44]. All scripts are called consecutively in a pipeline in a Perl script.

4.1 Revision of the source code

The original jpHMM source code was already implemented during my diploma thesis [84]. Due to constant modifications and extensions of jpHMM the source code became difficult to read or extend, especially for other users. Additionally, features of object-oriented programming have hardly been taken into account. Therefore, during this thesis main parts of the original jpHMM source code have been revised to provide an object-oriented C++ program that is easy to be used and to be modified by other users. Furthermore, some bugs in the code could be detected and fixed. Especially the Viterbi algorithm has been implemented in a different way, which can lead to slightly different recombination prediction results than those of the original jpHMM version. The reason is that several paths through the model generating the query sequence can have the same probability. In this case, the selection of one of these paths as Viterbi path only depends on the order of the calculation of the Viterbi variables in the Viterbi recursion. Changing the order in which the Viterbi variables of the states are calculated can slightly change the results of the original recombination prediction. But both results are correct.

4.2 Program workflow

1. **jpHMM: main** C++ **program:**
    (a) For circular genomes, a Perl script is called that extends the given multiple sequence alignment and each full-length query sequence as described in section 3.4.1, p. 59.
    (b) A jpHMM is generated on the basis of the (extended) multiple sequence alignment subdivided into several subtypes using the given pseudocounts for the estimation of the emission and transition probabilities (section 2.2.2, p. 18).
    (c) Optionally, a Perl script is called for defining active alignment columns for each query sequence position by pre-aligning each query sequence to the given multiple sequence alignment with BLAT (section 3.2.2, p. 37).
    (d) For each given query sequence, the Viterbi path through the model and thus its recombination of the given subtypes is determined (section 2.2.2, p. 21, section 3.2.2, p. 42, and section 3.4, p. 61).
    (e) The posterior probabilities (section 3.1.2, p. 30) are calculated using the Forward and Backward algorithm (section 3.1.1, p. 28).
2. **Uncertainty regions and breakpoint intervals:**
A Perl script determines uncertainty regions and breakpoint intervals in the predicted recombination on the basis of the posterior probabilities (section 3.1.3, p. 31).
3. **Graphical output:**
    (a) The posterior probabilities are plotted with a R script (Fig. 4.1).
    (b) For circular genomes, the predicted recombination including uncertainty regions and breakpoint intervals is visualized in a circular form with a Perl script (Fig. 4.2).

4.3 Input

1. **for the main jpHMM** C++ **program:**
   As input the main jpHMM program requires a file containing the query sequence(s) in FASTA [72] format and a file containing the multiple sequence alignment subdivided into subtypes in a FASTA-like format: Each subtype must be labeled with ">>" followed by the name of the subtype. For each subtype, the assigned sequences are then stored in the successive lines satisfying the FASTA format. Additionally, two files containing the pseudocounts for the calculation of the emission and the transition probabilities (section 2.2.2.5, p. 23) are required and certain parameters (for all of them default values are defined) can be specified:

   - the jump probability jp (default is 10^{-9} for HIV and 10^{-7} for HBV),
   - the beam-width $\mathcal{B}$ defining the degree of restriction of the Viterbi search space ($\mathcal{B} = 10^{-20}$ is the default for full-length sequences),
   - the type of virus (e.g. HIV or HBV) determining the input files (e.g. the pseudocounts) and parameters (e.g. the jump probability) if no further specifications are given, and the output format (e.g. in a circular form for HBV),
   - the type of genome (circular (e.g. for HBV) or non-circular (e.g. for HIV)),
   - the program version (the original jpHMM version (default) or the version using BLAT for a pre-alignment of the query sequence to the multiple alignment),
   - the kind of jumps (jumps are possible between all subtypes as in the original jpHMM version (default) or mute jump states are introduced to reduce the number of jumps being linear in terms of the number of subtypes in the model).

2. **for the Perl script extending circular genomes:**
   A query sequence file in FASTA and an alignment file in FASTA-like format (described above) are required. The length of the segment (default is 500) each query sequence is extended by at both ends can be specified. The alignment is duplicated and further extended by a certain factor $0 < e \leq 1$ (default is 0.25) that can also be specified by the user (section 3.4.1, p. 59). This script is called by the main jpHMM program.

3. **for the Perl script defining active alignment columns:**
   As input this script requires the path of the directory containing the BLAT program, the alignment and the query sequence file as described above, and the name of the

output file for the active alignment columns determined. If circular genomes are examined this must be specified. This script is called by the main jpHMM program.

4. **for the Perl script determining uncertainty regions and breakpoint intervals:** This script evaluates the output of the main jpHMM program. As input files it requires the ones containing the predicted recombination and the posterior probabilities. As parameters the posterior probability thresholds for defining uncertainty regions and breakpoint intervals can be specified (the default thresholds are $t_{\text{UR}} = t_{\text{BPI}} = 0.99$).

5. **for the scripts generating the plots:**
   (a) The R script for plotting the posterior probabilities requires the file including the posterior probabilities and a text file with a list of the predicted subtypes (see [84]). Both files are output files of the main jpHMM program.
   (b) The Perl script for visualizing circular genomes requires the path of the directory containing the parameter files for the Circos program, the original and the predicted recombination file including uncertainty regions and breakpoint intervals, the file specifying the alignment columns each query sequence position is aligned to with the Viterbi path, a file mapping each alignment column to the position in the virus reference sequence, and the name of the output directory for the files generated with the Circos program.

4.4 Output

The predicted recombination for a query sequence is given in text as well as GFF formats (`http://www.sanger.ac.uk/resources/software/gff/`) that have been described in my diploma thesis [84]. The new, extended jpHMM output comprises the following additional files:

Posterior probabilities of the subtypes The posterior probabilities of all subtypes at each query sequence position are given as a matrix where the rows describe the query sequence positions and the columns the subtypes. Additionally, the posterior probabilities of the two *insert* states at the beginning and the end of the model (called *5'*- and *3'-Insertion*) are given. As an example, an excerpt of the posterior probabilities file for an example HIV sequence is shown:

```
#example: 5'-Insertion A1 A2 B C D F1 F2 G H J K 01_AE O CPZ 3'-Insertion
1 8.90756e-152 1.83172e-148 5.71244e-144 4.23826e-153 [...] 0
```

```
1 1.26923e-151 3.02008e-148 6.0755e-144 8.23412e-153 [...] 0
1 1.9677e-151 5.22868e-148 6.74151e-144 1.59489e-152 [...] 0
1 5.25153e-149 1.50665e-146 3.32052e-141 2.46842e-150 [...] 0
1 1.3418e-148 3.68265e-146 8.51565e-141 6.31376e-150 [...] 0
1 1.34493e-148 3.78236e-146 8.51866e-141 6.34861e-150 [...] 0
1 1.35068e-148 3.96152e-146 8.52417e-141 6.41342e-150 [...] 0
1 1.36128e-148 4.28799e-146 8.53429e-141 6.53294e-150 [...] 0
```

The posterior probabilities are plotted with a R script using the ’seqinr’ package. For each subtype, the posterior probabilities at all query sequence positions are plotted in a different color. In Figure 4.1, the posterior probabilities are plotted for an example HIV-1 sequence.

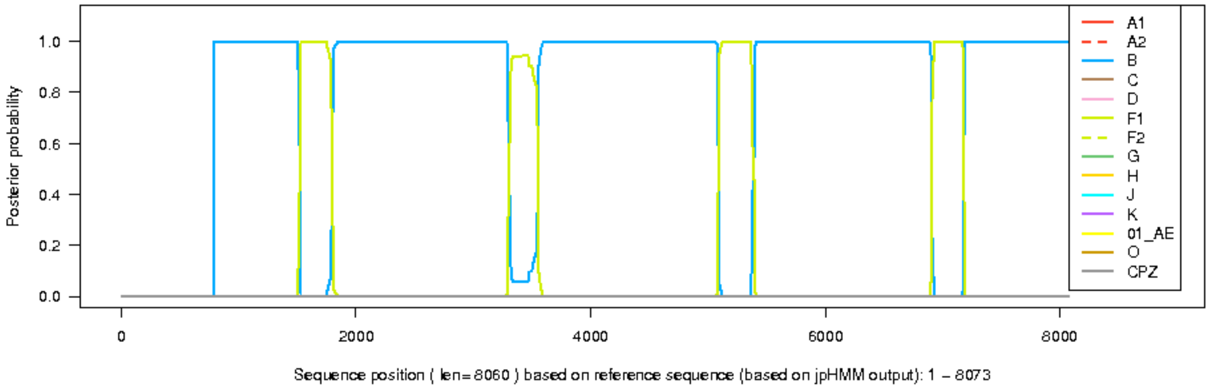


Figure 4.1: Posterior probabilities for an example HIV-1 sequence.

As additional information for each query sequence position the subtype with the highest posterior probability is determined:

```
>example
1   789 5’-Insertion
790 1521    B
1522    1796    F1
1797    3305    B
3306    3548    F1
3549    5088    B
5089    5387    F1
5388    6897    B
6898    7164    F1
7165    8060    B
```

Uncertainty regions and breakpoint intervals The predicted recombination including uncertainty regions and breakpoint intervals is given in text format. As an example, the recombination prediction including uncertainty regions and breakpoint intervals for a threshold of $t_{\text{UR}} = t_{\text{BPI}} = 0.99$ is shown for an example HIV sequence:

```
>example
1   789 N/A
790 1500    B
1501    1530    B/F1
1531    1749    F1
1750    1832    F1/B
1833    3286    B
3287    3589    ?/B
3590    5076    B
5077    5116    B/F1
5117    5367    F1
5368    5404    F1/B
5405    6882    B
6883    6918    B/F1
6919    7152    F1
7153    7171    F1/B
7172    8060    B
```

Additionally, the uncertainty regions and breakpoint intervals are given in a separate text file:

```
>example
# uncertainty regions:
3287    3589    ?/B
# breakpoint intervals:
1501    1530    B/F1
1750    1832    F1/B
5077    5116    B/F1
5368    5404    F1/B
6883    6918    B/F1
7153    7171    F1/B
```

Alignment of the query sequence to the given multiple alignment For each query sequence, the alignment to the given multiple sequence alignment, defined by the Viterbi path, is provided by indicating the aligned column for each query sequence position. This column corresponds to the state in the Viterbi path that emits the respective query sequence position. As an example, an excerpt of this file for an example HIV sequence is shown:

```
>example
501 502 503 504 505 506 507 508 509 510 511 512 513 514 [...] 13314
```

Graphical output for circular genomes For species with a circular genome, the predicted recombination is visualized in a circular form with a Perl script using the software package Circos [44] (Fig. 4.2).

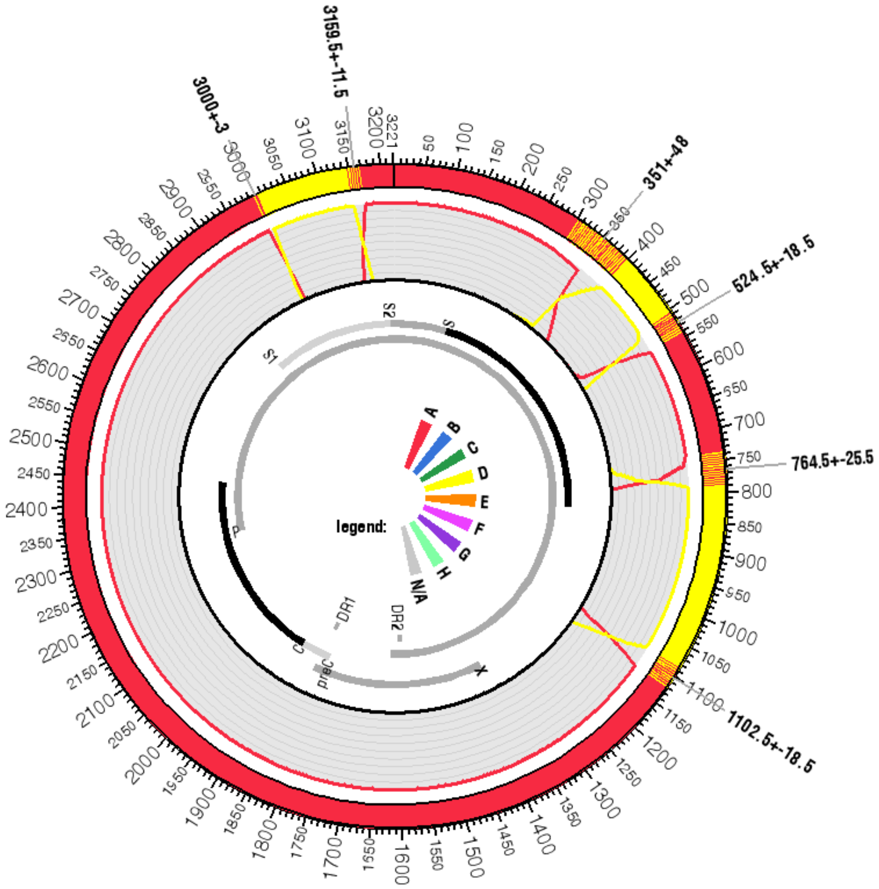


Figure 4.2: Circos plot of the recombination for a sample HBV sequence predicted with jpHMM. At the outer circle the predicted recombination is plotted based on reference sequence numbering. Striped regions mark breakpoint intervals between two subtypes. The exact position and length of each breakpoint interval is given by bold numbers. In the following circle the posterior probabilities of all subtypes are plotted for each query sequence position. A legend for the colors of the subtypes is given in the center of the plot. Additionally, the position of genes in the reference genome are given. DR1 and DR2 mark special regions that are important in the HBV replication cycle.

4.5 Runtime

The jpHMM runtime is evaluated in the Results chapter 5 by comparing the runtime of different jpHMM versions developed in this thesis.

4.6 Availability

jpHMM is available online as a webserver [109] at `http://jphmm.gobics.de/` and as a command-line tool. The use of the command-line version is described above in sections 4.2 - 4.4. The program, including all C++, Perl and R scripts described above, and a detailed documentation can be downloaded from the web page.

4.6.1 Webserver

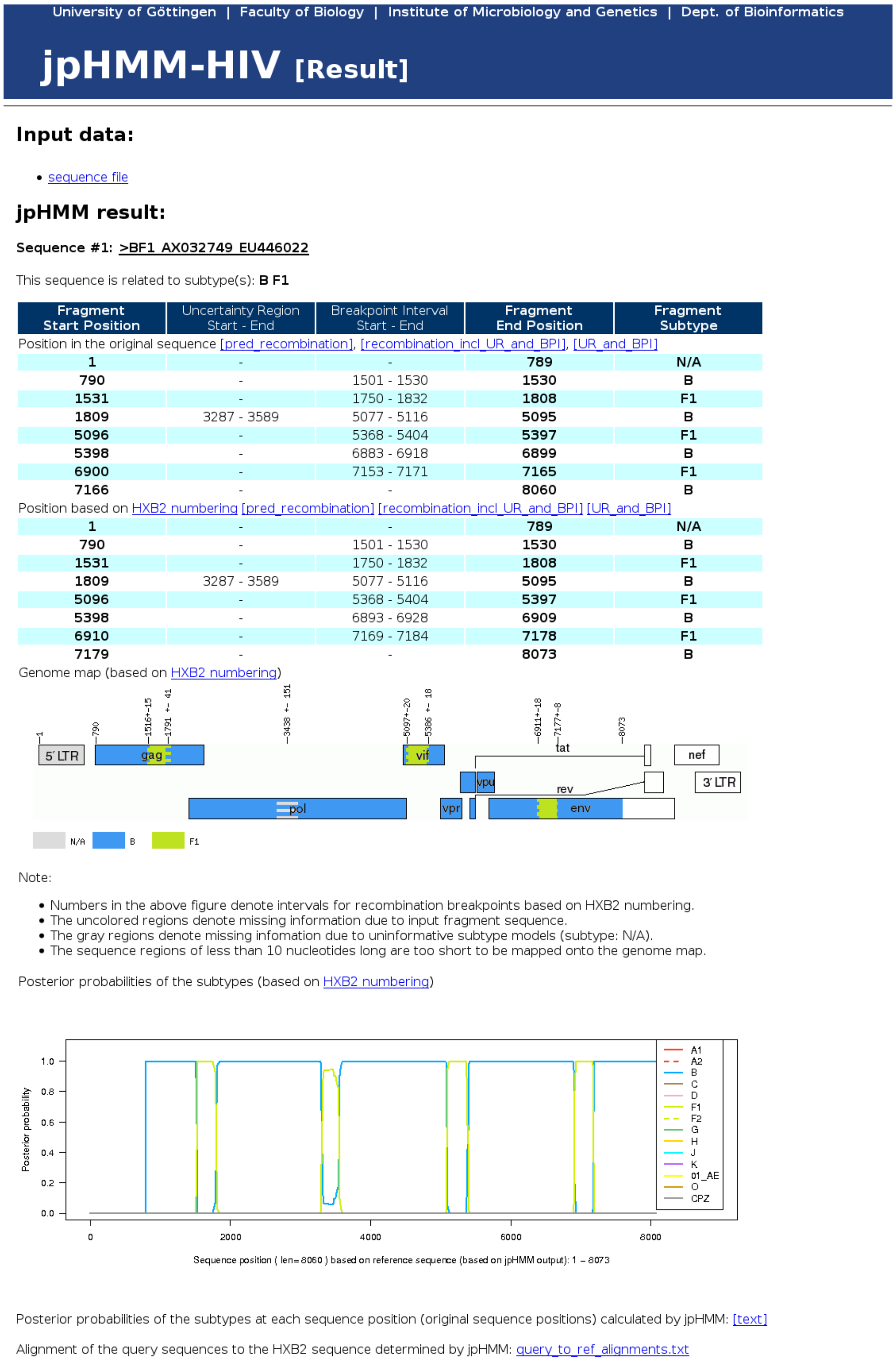
The webserver was originally implemented by Ming Zhang. During this thesis it was modified to incorporate information about uncertainty regions and breakpoint intervals [85]. The new webserver output for a sample HIV-1 sequence is shown in Figure 4.3. Additionally, the webserver is now available for HBV and provides an output in circular form (Fig. 4.2).

4.6.1.1 Input and output

As input, up to five full-length genomic sequences or fragments in FASTA format are accepted at a time. The jpHMM results are stored on the server for seven days, and a hyperlink to them is sent to the user by e-mail. For each input sequence, the predicted recombination, now including information about uncertainty regions and breakpoint intervals, is provided in text format and the posterior probabilities of the subtypes are plotted. The thresholds specified for the definition of uncertainty regions and breakpoint intervals are the default thresholds $t_{\text{UR}} = t_{\text{BPI}} = 0.99$. The predicted recombination pattern and the location of the sequence within the reference genome are represented graphically, for HIV-1 in a linear and for HBV in a circular form. Uncertainty regions and breakpoint intervals are marked by an interfingering of two colors, white for uncertainty regions and the color(s) of the predicted subtype(s). All jpHMM output files such as the predicted recombination including precise breakpoint positions as well as uncertainty regions and breakpoint intervals, and the posterior probabilities are provided for download.

4.6.1.2 Management of enquiries

Webserver enquiries are now managed using the resource management system *Sun Grid Engine* (SGE) [26]. Submitted jobs are queued and worked off one after the other depending on the available resources. This prevents the server from crashing if too many jobs are submitted at the same time.



University of Göttingen | Faculty of Biology | Institute of Microbiology and Genetics | Dept. of Bioinformatics

jpHMM-HIV [Result]

Input data:

- sequence file

jpHMM result:

Sequence #1: >BF1_AX032749_EU446022

This sequence is related to subtype(s): **B F1**

Fragment Start Position	Uncertainty Region Start - End	Breakpoint Interval Start - End	Fragment End Position	Fragment Subtype
Position in the original sequence [pred_recombination], [recombination_incl_UR_and_BPI], [UR_and_BPI]				
1	-	-	**789**	**N/A**
790	-	1501 - 1530	**1530**	**B**
1531	-	1750 - 1832	**1808**	**F1**
1809	3287 - 3589	5077 - 5116	**5095**	**B**
5096	-	5368 - 5404	**5397**	**F1**
5398	-	6883 - 6918	**6899**	**B**
6900	-	7153 - 7171	**7165**	**F1**
7166	-	-	**8060**	**B**
Position based on HXB2 numbering [pred_recombination] [recombination_incl_UR_and_BPI] [UR_and_BPI]				
1	-	-	**789**	**N/A**
790	-	1501 - 1530	**1530**	**B**
1531	-	1750 - 1832	**1808**	**F1**
1809	3287 - 3589	5077 - 5116	**5095**	**B**
5096	-	5368 - 5404	**5397**	**F1**
5398	-	6893 - 6928	**6909**	**B**
6910	-	7169 - 7184	**7178**	**F1**
7179	-	-	**8073**	**B**

Genome map (based on HXB2 numbering)

Note:

- Numbers in the above figure denote intervals for recombination breakpoints based on HXB2 numbering.
- The uncolored regions denote missing information due to input fragment sequence.
- The gray regions denote missing infomation due to uninformative subtype models (subtype: N/A).
- The sequence regions of less than 10 nucleotides long are too short to be mapped onto the genome map.

Posterior probabilities of the subtypes (based on HXB2 numbering)

Posterior probability
1.0
0.8
0.6
0.4
0.2
0.0
0
2000
4000
6000
8000
Sequence position (len= 8060) based on reference sequence (based on jpHMM output): 1 – 8073
A1
A2
B
C
D
F1
F2
G
H
J
K
01_AE
O
CPZ

Posterior probabilities of the subtypes at each sequence position (original sequence positions) calculated by jpHMM: [text]

Alignment of the query sequences to the HXB2 sequence determined by jpHMM: query_to_ref_alignments.txt

Figure 4.3: Screenshot of the new webserver output for an example HIV-1 recombinant including uncertainty regions and breakpoint intervals.

Chapter 5

Results and discussion

In this chapter, the methods developed in chapter 3 are applied to HIV-1 and HBV sequences and the results are discussed. First, the datasets (section 5.1) and the jpHMM parameters (section 5.2) that are used are described. Then, the accuracy of the jpHMM recombination prediction including uncertainty regions and breakpoint intervals is assessed on the basis of HIV-1 sequences (section 5.3). In section 5.4, the accuracy of jpHMM based on pre-defined active alignment column intervals is evaluated. It is also shown that BLAT is a suitable program for defining such active alignment column intervals to restrict the search space of the Viterbi algorithm in jpHMM. The accuracy of the modified jpHMM including mute jump states is compared to the accuracy of the original jpHMM in section 5.5. The runtime of all jpHMM versions developed in this thesis is analyzed in section 5.6. In section 5.7, first, the circular jpHMM version is applied to HBV sequences and its accuracy is evaluated. Then, several criteria for classifying recombinant forms of HBV are proposed.

5.1 Data

The HIV-1 and HBV datasets that are used in this thesis are presented in the following subsections. This includes the background alignments as well as training and test datasets.

5.1.1 HIV

All HIV-1 sequences studied in this thesis were downloaded from the HIV Sequence Database from the Los Alamos National Laboratory (LANL, `http://www.hiv.lanl.gov/`).

5.1.1.1 Multiple sequence alignment

The input multiple sequence alignment for jpHMM is built on the basis of the so-called "2007 multiple sequence alignment" from LANL [45]. This alignment was published in 2007 and includes near full-length sequences of all HIV-1 (sub-) subtypes A1, A2, B, C, D, F1, F2, G, H, J and K, of all CRFs, from group N and O and CPZ (chimpanzee) sequences. It was generated automatically using HMMER [31, 21] and edited manually afterwards. For the jpHMM input alignment, only the sequences of the (sub-) subtypes, CRF01_AE, group O and CPZ in this alignment are chosen. CRF01_AE is included in the alignment because it contains the only information of subtype E. Disadvantageously, some of the sequences in the alignment are not complete at the sequence ends. Therefore, the multiple sequence alignment is frayed and less informative at both ends, which can lead to problems in the recombination prediction in these regions, which will be discussed later.

5.1.1.2 Training data

Since the parameters of the jpHMM have already been determined during my diploma thesis [84] (described in [87]), no training datasets are needed.

5.1.1.3 Test data

In lack of real recombinant sequences with exactly known breakpoint positions, as test dataset for HIV-1 40 semi-artificial near full-length inter-subtype HIV-1 recombinant sequences with artificially introduced breakpoints were created. Each of the test sequences is a recombination of two 'real-world' parental sequences from two different HIV-1 (sub-) subtypes or circulating recombinant forms. To simulate the case with previously unobserved sequences that also differ by mutations from the known sequences, the parental sequences of all test sequences are not contained in the multiple sequence alignment we use to build the model. Since for the (sub-) subtypes A2, F2, H, J and K only a few sequences are available, of which all are included in the multiple sequence alignment, only every possible pair of the subtypes A1, B, C, D, F1, G and CRF_01 was chosen as parental subtypes.

The parental sequence pairs were used in three different datasets D1_HIV, D2_HIV and D3_HIV, differing by the position of artificially introduced recombination breakpoints. Hereby, the positions of breakpoints are given relatively to the HIV reference genome, called HXB2 sequence (GenBank [8] accession number K03455), that has a length of 9719 nt.

Dataset D1_HIV Segments of length 1000 nt from one subtype are interrupted by segments of length 1000 nt from another subtype. Introducing a breakpoint at every 1000th position based on HXB2 numbering results in nine recombination breakpoints at positions $1000, 2000, 3000, \ldots, 8000, 9000$.

Dataset D2_HIV Alternating long segments of length 1500 nt from one subtype are interrupted by short segments of length 500 nt from another subtype. So, the breakpoint positions based on HXB2 numbering are $1500, 2000, 3500, 4000, \ldots, 8000$ and 9500. Due to the length of some parental sequences in seven test sequences the breakpoint at position 9500 is missing.

Dataset D3_HIV Alternating long segments of length 1500 nt from one subtype are interrupted by short segments of length 300 nt from another subtype, resulting in sequences with 10 recombination breakpoints at positions $1500, 1800, 3300, 3600, \ldots, 8700$ and 9000.

So, in total, 120 artificial recombinant sequences were evaluated, each having eight to ten recombination breakpoints.

5.1.2 HBV

The HBV sequences studied here comprise all nearly full-length ($\geq$ 3000 nt) HBV genomic sequences available in December 2009 from GenBank [8]. As presented in chapter 2, HBV sequences are classified into different genotypes instead of subtypes (for HIV). Therefore, we use this notation as well.

5.1.2.1 Reference genome

For a consistent representation of sequence positions, e.g. positions of breakpoints, in the HBV genome, all sequence position numbers are given relative to a reference strain. As reference genome the sequence with the GenBank accession number AM282986 was chosen. AM282986 is a well-annotated sequence [71] that belongs to genotype A. With a length of 3221 nt, genotype A is one of the genotypes with the largest number of nucleotides. Using a sequence of a shorter genotype as reference genome could, for example, result in accumulating breakpoint positions relative to the reference genome if the evaluated sequence has a large insertion compared to the reference genome. A sequence of genotype G, that even has a length of 3248 nt, was not chosen as reference genome since this larger size of genotype G, compared to all other genotypes, results from an insertion that is only present in this genotype.

5.1.2.2 Multiple sequence alignment

Sequences for the HBV multiple sequence alignment that is used to build the jpHMM have been chosen from all sequences that were published as pure genotypes. The published genotype composition of the sequences has been rechecked with jpHMM, based on an initial multiple sequence alignment of clearly identified pure genotype sequences. For this test, a high jump probability ($jp = 10^{-3}$) was used to ensure that the sequences really represent pure genotypes. From this set of verified pure genotype sequences, certain sequences have been selected for the multiple alignment, taking into account the following aspects:

1. only full-length genomic sequences were chosen for each genotype
2. the global variety of each genotype should be represented
3. identical or nearly identical sequences should not be contained in the alignment
4. the number of sequences in the alignment is limited due to restrictions of multiple sequence alignment methods

These aspects led to the decision that for each genotype about 50 representative sequences should be chosen, which would lead to about 400 sequences in the whole alignment.

The sequences for each of the eight HBV genotypes A-H were clustered with CD-HIT-EST, a very fast and widely used program of the CD-HIT Suite [50, 33] for clustering DNA sequence datasets: on the basis of a certain sequence identity threshold, the input sequences are clustered and a database of representative sequences is generated. As for genotypes H and G less than 50 sequences were available, only $100\,\%$ identical sequences have been removed from the dataset, resulting in 16 representative sequences for genotype G and 24 representative sequences for genotype H. Sequences of genotype C were clustered using a sequence identity threshold of $97\,\%$, since a great many of sequences was available. To all other genotypes a threshold of $99\,\%$ was applied to achieve about 50 clusters, i.e. 50 representative sequences. For genotypes with more than 50 clusters, 50 clusters were chosen randomly. For genotype F 49 clusters were built. The representative sequences of all eight genotypes were aligned with Muscle [22]. The genotype classification of the aligned sequences was rechecked by using each aligned sequence as query sequence for jpHMM with the given multiple sequence alignment as input alignment (removing the respective query sequence from the alignment) and a very high jump probability of 10^{-3}.

Later, one sequence of genotype F was removed from the given multiple sequence alignment because of a long deletion in the sequence that had a large influence to the parameter estimation. Thus, the final HBV alignment contains 339 sequences.

5.1.2.3 Training data

As for HIV-1, semi-artificial recombinants from real pure genotype HBV genomic sequences with artificially introduced breakpoints were created. Each of these recombinants is a recombination of two 'real-world' parental sequences from two different HBV genotypes. As parental sequences, sequences from the multiple sequence alignment were chosen, since these sequences are confirmed pure genotype sequences satisfying a certain minimum of pairwise distance. To simulate previously unobserved sequences, for each recombinant the two respective parental sequences were removed from the alignment that is used to build the jpHMM in the respective evaluation.

Dataset D0_HBV For each pair of genotypes A-H 10 semi-artificial recombinants were created and the corresponding $2 \cdot 10$ parental sequences were chosen randomly from the given set of sequences. The number of artificially introduced breakpoints n_{bp}, $n_{bp} \in \{0, \ldots, 4\}$, as well as the breakpoint positions were chosen randomly. Therefore, also pure genotype sequences ($n_{bp} = 0$) and recombinants with segments of length as low as 1 were possible. In total, 280 semi-artificial recombinant sequences with known breakpoint positions were created. (Later, one sequence of genotype F was removed from the given multiple sequence alignment because of a long deletion in the sequence that had a large influence to the parameter estimation. Artificial recombinants including this F sequence were removed from the set of training sequences, resulting in 276 sequences.)

5.1.2.4 Test data

Semi-artificial test recombinants were created from the sequences in the multiple sequence alignment as described in the previous subsection for the training data (section 5.1.2.3) only the difference is is that the breakpoints were introduced at fixed positions relative to the HBV reference genome as it was also done for the sequences in the HIV-1 test dataset (section 5.1.1.3).

Dataset D1_HBV Segments of length 1000 nt from one genotype are interrupted by segments of length 1000 nt from another genotype resulting in three recombination breakpoints at positions $1000, 2000$ and 3000.

Dataset D2_HBV Alternating short segments of length 500 nt from one genotype are interrupted by long segments of length 1500 nt from another genotype. Thus, the breakpoints are located at positions $500, 2000$ and 2500 in the reference genome.

Dataset D3_HBV Alternating short segments of length 300 nt from one genotype are interrupted by long segments of length 1500 nt from another genotype. This also results in sequences with three recombination breakpoints at positions $300, 1800$ and 2100 in the reference genome.

In total, 840 artificial recombinant sequences were evaluated. In case that the test sequences are treated as circular sequences (section 5.7), each test sequence contains a fourth breakpoint at the sequence end.

5.1.2.5 Real-world HBV genomic sequences

To evaluate the genetic diversity of all published real-world HBV strains all nearly full-length (≥ 3000 nt) HBV genomic sequences available in GenBank in December 2009 were downloaded. After removing eight sequences with a long stretch ($\geq$ 50 nt) of non-specified nucleotides (N's) this dataset comprised 2918 sequences.

5.2 jpHMM parameter estimation

In this section, the estimation of the jpHMM parameters for the application to HBV sequences is described. For HIV-1, the original parameters are used.

5.2.1 HIV-1

As parameters for the jpHMM for recombination prediction in HIV-1 the parameters given in section 2.2.2 (p. 23) and described in [87] are used.

5.2.2 HBV

For the jpHMM for HBV, the pseudocounts for the emission probabilities of the insert states were set to 1, since they do not have much influence on the results of the recombination prediction. As pseudocounts for the transition probabilities the same values as for HIV-1 (see (2.6), p. 24) were chosen (described in [104]).

The jump probability jp and the pseudocounts $\vec{\alpha}$ for the emission probabilities of the match states were estimated on the basis of HBV training sequences (dataset D0_HBV, section 5.1.2.3). They were estimated jointly, assuming identical pseudocounts α for the emission probabilities of all nucleotides. The tested pairs of parameters are

$$\Theta := (jp, \alpha) \in \{10^{-5}, \ldots, 10^{-10}\} \times \{0.5, 0.1, 0.09, \ldots, 0.01, 0.009, \ldots 0.001\}. \quad (5.1)$$

As training data, the dataset D0_HBV (section 5.1.2.3, p. 79), that consists of 276 semi-artificial recombinant sequences with breakpoints introduced at random positions, was used. The parameter pair $\Theta^* = (jp^*, \alpha^*)$ that maximized the accuracy of predicted breakpoint intervals and genotypes in these training sequences was chosen as the optimal pair of parameters for the application of jpHMM to HBV sequences.

There are several possibilities to measure the accuracy of recombination prediction. For example, the **accuracy of predicted breakpoint intervals** can be measured by the number of predicted breakpoint intervals that contain a real breakpoint (*specificity*, Def. 5.1) and by the number of real breakpoints that can be detected with the predicted breakpoint intervals (*sensitivity*, Def. 5.2). To measure the **accuracy of predicted genotypes**, it can, for example, be taken into account if the genotypes are predicted correctly, regardless of the predicted order (*set of predicted genotypes*, Def. 5.7), or if the genotypes are predicted in the correct order (*predicted recombination pattern*, Def. 5.8). It is also possible to count the number of sequence positions that are assigned to the correct genotype (Def. 5.9).

These five criteria were taken into account to assess the accuracy of the recombination prediction for the training sequences for different pairs of parameters (see (5.1)). On the basis of the obtained values, the optimal pair of parameters was determined.

5.2.2.1 Definitions

Let $\Theta = (jp, \alpha)$ be a pair of parameters (see (5.1)).

Definition 5.1 (Specificity of breakpoint intervals)

1. *The* specificity *of predicted breakpoint intervals is defined as the ratio of the number of predicted breakpoint intervals (BPI) that contain a real breakpoint (BP) having the same preceding and successive genotype to the total number of predicted breakpoint intervals:*
$$\text{spec} := \frac{\#\text{ predicted BPIs that contain a real BP}}{\#\text{ predicted BPIs}} \tag{5.2}$$

2. spec_i^{Θ} *denotes the* specificity of breakpoint intervals of fixed length i *around predicted breakpoints for the parameter pair* Θ*.*

3. *The* average specificity *of breakpoint intervals of fixed lengths* $k_1, \ldots, k_n$ *for* Θ *is defined by*
$$\text{av\_spec}_{k_1,\ldots,k_n}^{\Theta} := \frac{\text{spec}_{k_1}^{\Theta} + \ldots + \text{spec}_{k_n}^{\Theta}}{n} \tag{5.3}$$

Definition 5.2 (Sensitivity of breakpoint intervals)

1. *The* sensitivity *of predicted breakpoint intervals is defined as the ratio of the number of real breakpoints (BP) that are located in a predicted breakpoint interval(BPI) having the same preceding and successive genotype to the total number of real breakpoints:*
$$\text{sens} := \frac{\#\text{ real BPs located in a predicted BPI}}{\#\text{ real BPs}} \tag{5.4}$$

2. sens_i^{Θ} *denotes the* sensitivity of breakpoint intervals of fixed length i *around predicted breakpoints for the parameter pair* Θ*.*

3. *The* average sensitivity *of breakpoint intervals of fixed lengths* $k_1, \ldots, k_n$ *for* Θ *is defined by*
$$\text{av\_sens}_{k_1,\ldots,k_n}^{\Theta} := \frac{\text{sens}_{k_1}^{\Theta} + \ldots + \text{sens}_{k_n}^{\Theta}}{n} \tag{5.5}$$

Example 5.3

For example, the average specificity and sensitivity of breakpoint intervals of fixed length

10, 20, etc. up to length 100, are

$$\text{av\_spec}^{\Theta}_{10,20,\ldots,100} = \frac{\text{spec}^{\Theta}_{10} + \text{spec}^{\Theta}_{20} + \ldots + \text{spec}^{\Theta}_{100}}{10} \tag{5.6}$$

and

$$\text{av\_sens}^{\Theta}_{10,20,\ldots,100} = \frac{\text{sens}^{\Theta}_{10} + \text{sens}^{\Theta}_{20} + \ldots + \text{sens}^{\Theta}_{100}}{10} \quad \text{respectively.} \tag{5.7}$$

Definition 5.4 (Set of predicted genotypes)
The set of predicted genotypes *for a query sequence is the unique set of genotypes predicted for at least one base in the sequence regardless of the order of the predicted genotypes.*

Definition 5.5 (Predicted recombination pattern)
The predicted recombination pattern *of a query sequence is defined as the sequence of predicted genotypes, not taking into account the location of predicted breakpoints or breakpoint intervals.*

Example 5.6 (Example for predicted recombination patterns and set of genotypes)
Let $S1$ and $S2$ be query sequences with the following predicted recombinations:

```
>S1                    >S2
   1  500 A               1 1500 A
 501 1000 B            1501 2000 B
1001 1500 A            2001 2500 C
1501 2000 C            2501 3200 A
2001 2500 A
2501 3000 B
3001 3200 A
```

1. The set of predicted genotypes is identical for both sequences, namely ABC.
2. The predicted recombination pattern of S_1 is ABACABA, of S_2 ABCA.

Definition 5.7 (Accuracy of set of predicted genotypes)
The accuracy $\text{acc}_{\text{ST\_set}}$ of sets of predicted genotypes for a certain pair Θ of parameters *is defined as the ratio of the number of sequences with a correctly predicted set of genotypes to the total number of sequences:*

$$\text{acc}^{\Theta}_{\text{ST\_set}} := \frac{\#\text{ sequences with correctly predicted set of genotypes}}{\#\text{ sequences}} \tag{5.8}$$

Definition 5.8 (Accuracy of predicted recombination pattern)
The accuracy $\text{acc}_{\text{recomb\_pattern}}$ of predicted recombination patterns for a certain pair Θ of parameters *is defined as the ratio of the number of sequences with a correctly predicted recombination pattern to the total number of sequences:*

$$\text{acc}^{\Theta}_{\text{recomb\_pattern}} := \frac{\#\text{ sequences with correctly predicted recombination pattern}}{\#\text{ sequences}} \tag{5.9}$$

Definition 5.9 (Accuracy of predicted genotypes at each sequence position))
The accuracy $\text{acc}_{\text{pred\_ST}}$ of predicted genotypes for a certain pair Θ of parameters *is defined by the ratio of the number of sequence positions, located outside of uncertainty regions (UR) and breakpoint intervals (BPI), that are assigned to the correct genotype to the total number of sequence positions:*

$$\text{acc}^{\Theta}_{\text{pred\_ST}} := \frac{\#\text{ sequence positions } \notin \{\text{UR, BPI}\}\text{ assigned to correct genotype}}{\#\text{ sequence positions}} \tag{5.10}$$

5.2.2.2 Parameter estimation

For each parameter pair $\Theta = (jp, \alpha)$, the five scores

$$sc^{\Theta}_1 := \text{acc}^{\Theta}_{\text{ST\_set}} \tag{5.11}$$
$$sc^{\Theta}_2 := \text{acc}^{\Theta}_{\text{recomb\_pattern}} \tag{5.12}$$
$$sc^{\Theta}_3 := \text{av\_spec}^{\Theta}_{10,20,\ldots,100} \tag{5.13}$$
$$sc^{\Theta}_4 := \text{av\_sens}^{\Theta}_{10,20,\ldots,100} \tag{5.14}$$
$$sc^{\Theta}_5 := \text{acc}^{\Theta}_{\text{pred\_ST}} \tag{5.15}$$

were determined. Thus, for each $i \in \{1, \ldots, 5\}$, a set of scores was obtained. Let

$$sc_i := (sc^{\Theta_1}_i, \ldots, sc^{\Theta_m}_i), \quad sc^{\Theta_1}_i \leq \ldots \leq sc^{\Theta_m}_i, \tag{5.16}$$

be the sorted list of the scores for each $i \in \{1, \ldots, 5\}$ with $\Theta_1, \ldots, \Theta_m$ being the tested pairs of parameters. Then, for a certain pair of parameters Θ, the *rank* of sc^{Θ}_i in sc_i,

$$\text{rank}_{sc_i}(sc^{\Theta}_i), \tag{5.17}$$

defines the position of the score sc^{Θ}_i in the sorted list sc_i of scores (5.16) determined for i, $i \in \{1, \ldots, 5\}$ (see (5.11) to (5.15)).

The score sc^{Θ} of the parameter pair $\Theta = (jp, \alpha)$ is then defined by the sum of the five ranks determined for Θ:

$$sc^{\Theta} := \sum_{i=1}^{i=5} \text{rank}(sc_i^{\Theta}) \tag{5.18}$$

The pair $\Theta^* = (jp^*, \alpha^*)$ of parameters that maximizes this score was chosen as parameter pair for the jpHMM for recombination detection in HBV genomes:

Definition 5.10 (Optimal pair of parameters)

$$\Theta^* = (jp^*, \alpha^*) = \arg \max_{\Theta=(jp,\alpha)} sc^{\Theta} \tag{5.19}$$

5.2.2.3 Results

As the training sequences contain breakpoints at randomly chosen positions, also very short recombination segments are included in the sequences. Thus, high jump probabilities and very small pseudo counts gave the best results. But, a jpHMM using such parameters tends to produce a lot of jumps on the basis of only one or a few matches resulting in short predicted recombination segments. Therefore, in a first step, all pairs of parameters with a score not much lower than the score of the best pair of parameters were accepted as possible jpHMM parameters.

In the next step, for all these remaining pairs of parameters, the jpHMM recombination predictions for real-world HBV recombinants were compared. Pairs of parameters leading to obviously incorrect jumps (e.g. a jump to genotype F for only a few bases within a recombinant of genotypes A and B) were removed from the set of possible parameter pairs. Additionally, the results of the recombination prediction were compared to the recombination patterns of 24 independent recombinant forms published in [90]. This study is one of the first studies that defines independent recombinant forms among HBV sequences on the basis of all publicly available full-length HBV sequences (in October 2004).

After this extensive evaluation the following pair of parameters was chosen as optimal pair of parameters for the application of jpHMM to recombination detection in HBV:

$$\Theta^* = (jp^*, \alpha^*) = (10^{-7}, 0.009) \tag{5.20}$$

5.3 Uncertainty regions and breakpoint intervals

As customary, and in lack of real recombinant sequences with exactly known breakpoint positions, the accuracy of the jpHMM recombination prediction including uncertainty regions and breakpoint intervals (section 3.1, p. 27) is evaluated on recombinant sequences of real HIV-1 sequences with artificially introduced breakpoints. It is measured by the accuracy of the predicted recombination patterns (section 5.3.1), the sensitivity of the predicted breakpoint intervals (section 5.3.2) and the accuracy of the predicted parental subtypes at positions outside uncertainty regions and breakpoint intervals (section 5.3.3). As test data the datasets D1_HIV, D2_HIV and D3_HIV, described in section 5.1.1 (p. 76) are used. The accuracy of the jpHMM recombination prediction for HBV is evaluated in section 5.7 when a jpHMM for species with circular genomes like HBV is presented.

The major results for HIV-1 of this section have previously been published in

A.-K. Schultz, M. Zhang, I. Bulla, T. Leitner, B. Korber, B. Morgenstern, M. Stanke.
jpHMM: Improving the reliability of recombination prediction in HIV-1
Nucleic Acids Research 2009 37:W647-W651
doi:10.1093/nar/gkp371

5.3.1 Accuracy of predicted recombination patterns

In a first step, the accuracy of the recombination patterns predicted by jpHMM is evaluated. In this case, the *predicted recombination pattern* is the *sequence of predicted subtypes*, not taking into account the location of predicted breakpoints or breakpoint intervals. Therefore, the recombination pattern of a sequence can be predicted correctly, even if not all breakpoints are *detected* (Def. 5.11). For example, if the predicted position of a breakpoint interval is shifted compared to the true breakpoint position, so that the breakpoint interval does not contain the true breakpoint, the predicted recombination pattern is still correct. For the three datasets D1_HIV, D2_HIV and D3_HIV, the following results were obtained:

Dataset D1_HIV For 39 of the 40 sequences, the recombination pattern was predicted correctly. In one of the 40 sequences one recombinant segment was not identified, i.e. one jump from one subtype to the second subtype and back was missing.

Dataset D2_HIV For 33 of the 40 sequences, one short segment (500 nt) at the sequence end was not assigned to the correct subtype, since jpHMM was not able to assign any subtype in this region. This is a consequence of the 'frayed' ends of the multiple sequence alignment. Sequence positions that are located within or near these genomic regions often cannot be assigned to any subtype by jpHMM. Apart from these 33 breakpoints, the predicted recombination pattern was correct.

Dataset D3_HIV For 32 of the 40 sequences, the recombination pattern was predicted correctly. In the remaining eight sequences one short segment (300 nt) was not assigned to the correct subtype. Three of these eight segments could not be identified as a recombinant segment but were predicted to have the same subtype as their neighbor segments. The other five segments were classified as uncertainty regions. A possible reason for this incorrect classification is that for the currently used jump probability of 10^{-9}, jpHMM is often not able to detect short recombinant segments. Higher jump probabilities might solve this problem but would also lead to a higher rate of false positive recombinant segments, i.e. regions that are incorrectly predicted as recombinant segments.

5.3.2 Sensitivity of predicted breakpoint intervals

The accuracy of the predicted breakpoint intervals is measured in terms of the sensitivity of the predicted breakpoint intervals, i.e. the number of real breakpoints *detected* by the predicted breakpoint intervals. The following definition holds:

Definition 5.11 *A breakpoint is defined as* detected *when the predicted breakpoint interval contains the breakpoint and, when the two subtypes to the left and to the right of the breakpoint interval are predicted correctly.*

Different thresholds t_{BPI} are used to define breakpoint intervals based on the posterior probabilities:

$$t_{\text{BPI}} \in \{0.75, 0.85, 0.9, 0.95, 0.99, 0.9999\} \tag{5.21}$$

For each of these thresholds, the number of detected breakpoints is determined. The accuracy of the predicted breakpoint intervals is compared to the results of a naïve approach, explained below in subsection 5.3.4. In Table 5.1, for each threshold given in (5.21), the percentage of real breakpoints detected with the corresponding breakpoint intervals is presented (column 4). As additional information the average (column 2), the minimum (column 3) and the maximum length (column 3) of the predicted breakpoint intervals are given.

Table 5.1: Comparison of the accuracy of predicted breakpoint intervals and breakpoint intervals of fixed length for the three tested datasets. The table for dataset D1_HIV (1000nt/1000 nt) was originally published in [86]. In column 1 the threshold t_{BPI} for the posterior probabilities to define breakpoint intervals (BPI) is given. The average length of the predicted breakpoint intervals defined by t_{BPI} is given in column 2, the minimum (Min.) and maximum length (Max.) in column 3. The percentage of real breakpoints (BPs) detected with the predicted breakpoint intervals (using the posterior probabilities (P_{post})) and with the naïve approach (using breakpoint intervals of fixed length around each predicted breakpoint (Fixed BPI length)) are shown in column 4 and 5. The length of the breakpoint intervals of fixed length in the naïve approach corresponds to the average length of the predicted breakpoint intervals. For each threshold, the highest value is marked in bold face.

Dataset D1_HIV (1000/1000 nt)

Threshold	BPI length		Percentage of detected BPs using	
t_{BPI}	Average	Min. / Max.	P_{post}	Fixed BPI length
0.75	16.12	0 / 113	54.17	**59.44**
0.85	22.46	0 / 121	**68.06**	66.67
0.90	26.89	2 / 135	**74.72**	69.72
0.95	34.05	2 / 202	**81.11**	75.56
0.99	48.58	5 / 233	**92.50**	82.78
0.9999	84.77	11 / 492	**98.06**	92.50

Dataset D2_HIV (1500/500 nt)

Threshold	BPI length		Percentage of detected BPs using	
t_{BPI}	Average	Min. / Max.	P_{post}	Fixed BPI length
0.75	14.56	0 / 84	**47.88**	45.61
0.85	20.10	0 / 109	**59.21**	55.24
0.90	23.73	2 / 115	**65.16**	59.49
0.95	30.11	2 / 202	**72.52**	64.59
0.99	43.73	5 / 233	**82.72**	69.97
0.9999	76.30	11 / 268	**88.10**	80.45

Dataset D3_HIV (1500/300 nt)

Threshold	BPI length		Percentage of detected BPs using	
t_{BPI}	Average	Min. / Max.	P_{post}	Fixed BPI length
0.75	13.75	1 / 105	**53.75**	49.50
0.85	18.85	1 / 111	**67.25**	59.25
0.90	22.29	1 / 114	**73.25**	63.50
0.95	28.36	1 / 123	**80.00**	69.25
0.99	41.24	1 / 143	**87.50**	80.50
0.9999	71.40	5 / 245	**91.75**	89.75

5.3.3 Accuracy of predicted subtypes

Another measure for the accuracy of predicted uncertainty regions and breakpoint intervals is the accuracy of predicted subtypes at positions outside uncertainty regions and breakpoint intervals. In Table 5.2, for all tested thresholds and using $t_{\text{UR}} := t_{\text{BPI}}$ as posterior probability threshold for the uncertainty regions, the percentage of positions outside uncertainty regions and breakpoint intervals that were assigned to a subtype and classified correctly (column 2) is presented. Additionally, the percentage of *unclassified* positions at the sequence ends (column 3) is determined. Unclassified positions are positions that are emitted by one of the two special insert states at the beginning and the end of the model and therefore are not assigned to any subtype. For each dataset, the number of unclassified positions is the same for all tested thresholds (e.g. $6.74\,\%$ for dataset D1_HIV).

On the basis of the percentage of unclassified and correctly classified positions, the percentage of positions that were classified incorrectly is calculated (column 4). This percentage is compared to the percentage of incorrectly predicted sequence positions with the original jpHMM version using the Viterbi algorithm and point estimates of breakpoints only (Table 5.3). For dataset D1_HIV, the percentage of incorrectly predicted positions is only $0.58 - 0.82\,\%$ compared to $1.51\,\%$ for the original jpHMM version. For dataset D2_HIV and D3_HIV, these percentages are $0.66 - 0.99\,\%$ compared to $1.55\,\%$ and $0.75 - 1.08\,\%$ compared to 1.97%, respectively.

5.3.4 Comparison to a naïve approach

The most obvious and naïve method to define breakpoint intervals around predicted breakpoint positions, if no further information is provided, is to define a symmetric interval of fixed length, centered around the predicted breakpoint position. In Table 5.1, column 5, the accuracy of such breakpoint intervals of fixed lengths is given and compared to the accuracy of breakpoint intervals defined by the posterior probabilities. For a direct comparison, the fixed breakpoint interval length corresponds to the average length of the predicted breakpoint intervals defined by the posterior probabilities (column 2), rounded to the nearest even number. The results show that for all posterior probability thresholds evaluated (except $t_{\text{BPI}} = 0.75$ for dataset D1_HIV), especially for high thresholds, the percentage of breakpoints detected with breakpoint intervals defined by the posterior probabilities (column 4) is much higher than the percentage achieved with the naïve approach, i.e. using breakpoint intervals of fixed length (column 5). For the default threshold $t_{\text{BPI}}^{\text{df}} = 0.99$, the results are again summarized in Table 5.4. For dataset D1_HIV, the sensitivity of the extended jpHMM version predicting uncertainty regions in the recombination prediction and

Table 5.2: For different posterior probability thresholds, $t_{\text{UR}} = t_{\text{BPI}}$, the percentage of sequence positions located outside of breakpoint intervals and uncertainty regions that are classified correctly, unclassified and classified incorrectly is given. Unclassified positions are positions at both ends of the sequences that were not assigned to any subtype by the Viterbi path. For each dataset, the number of unclassified positions is the same for all tested thresholds.

dataset D1_HIV (1000/1000 nt)

Threshold	Percentage of positions		
$t_{\text{UR}} = t_{\text{BPI}}$	Classified correctly	Unclassified	Classified incorrectly
0.75	92.44	6.74	0.82
0.85	92.56	6.74	0.70
0.90	92.61	6.74	0.65
0.95	92.65	6.74	0.61
0.99	92.68	6.74	0.58
0.9999	92.54	6.74	0.72

dataset D2_HIV (1500/500 nt)

Threshold	Percentage of positions		
$t_{\text{UR}} = t_{\text{BPI}}$	Classified correctly	Unclassified	Classified incorrectly
0.75	92.27	6.74	0.99
0.85	92.38	6.74	0.88
0.90	92.42	6.74	0.84
0.95	92.49	6.74	0.77
0.99	92.59	6.74	0.67
0.9999	92.60	6.74	0.66

dataset D3_HIV (1500/300 nt)

Threshold	Percentage of positions		
$t_{\text{UR}} = t_{\text{BPI}}$	Classified correctly	Unclassified	Classified incorrectly
0.75	92.15	6.77	1.08
0.85	92.24	6.77	0.99
0.90	92.34	6.77	0.89
0.95	92.45	6.77	0.78
0.99	92.48	6.77	0.75
0.9999	92.37	6.77	0.86

Table 5.3: Comparison of the percentage of sequence positions classified incorrectly with jpHMM including uncertainty regions (UR) and breakpoint intervals (BPI) in the recombination prediction (columns 2 and 3) and with the original jpHMM version using point estimates of breakpoints only (Orig. jpHMM, column 4). For the recombination prediction including uncertainty regions, different posterior probability thresholds have been used and the results are shown in Table 5.2. Here, only the minimum (Min., column 2) and maximum (Max., column 3) of the achieved percentages are given for the comparison.

Dataset	Percentage of positions classified incorrectly based on		
	jpHMM incl. UR/BPI		Orig. jpHMM
	Min.	Max.	
D1_HIV	0.58	0.82	1.51
D2_HIV	0.66	0.99	1.55
D3_HIV	0.75	1.08	1.97

interval estimates of breakpoints is up to 9.72 percentage points higher than the sensitivity of the naïve method. For dataset D2_HIV, the sensitivity is even increased by 12.75 percentage points, and for dataset D3_HIV by 7.0 percentage points.

Table 5.4: Summary of the comparison of the accuracy of predicted breakpoint intervals (BPI) for the default posterior probability threshold $t_{\mathrm{BPI}} = 0.99$ (P_{post}) and breakpoint intervals of the corresponding fixed length (Fixed BPI length) for the three tested datasets.

Dataset	Percentage of detected BP using	
	P_{post}	Fixed BPI length
D1_HIV	92.50	82.78
D2_HIV	82.72	69.97
D3_HIV	87.50	80.50

5.3.5 Default posterior probability threshold for breakpoint intervals

Obviously, the higher the thresholds for the posterior probabilities the higher the accuracy of the predicted breakpoint intervals. But this increase is accompanied by an enormous enlargement of the predicted breakpoint intervals. Weighing up the size of the predicted breakpoint intervals against their accuracy,

$$t_{\text{BPI}}^{\text{df}} = 0.99$$

is chosen as default threshold $t_{\text{BPI}}^{\text{df}}$ for the definition of breakpoint intervals. For this threshold, 92.50% of the real breakpoints in dataset D1_HIV were able to be detected (Table 5.1, column 4). The average length of the corresponding predicted breakpoint intervals is 48.58 nt, with a minimum length of 5 nt and a maximum length of 233 nt. In dataset D2_HIV only 82.72% (Table 5.1, column 4) of the real breakpoints were able to be detected (with an average breakpoint interval length of 43.73 nt). But, as described above in subsection 5.3.1, $9.35\,\%$ (33 out of 353) of the real breakpoints were not able to be detected, because they were located within an unclassified region. So, only $7.93\,\%$ of the real breakpoints were predicted at incorrect positions in the genome. The average length of predicted breakpoint intervals in dataset D3_HIV for $t_{\text{BPI}} = 0.99$, is 41.24 nt. $87.50\,\%$ (Table 5.1, column 4) of the real breakpoints were able to be detected.

5.3.6 Discussion

The results show that the definition of uncertainty regions and breakpoint intervals strongly improves the reliability of the recombination prediction with jpHMM. Breakpoint intervals defined on the basis of posterior probabilities are much more accurate than breakpoint intervals of fixed length since their length depends on how precisely jpHMM can locate the breakpoint reliably. A large interval is the consequence of the uncertainty of the model to locate the exact breakpoint position between two subtypes. For example, breakpoints located in conserved genomic regions cannot be determined precisely by the model. Such a breakpoint could be predicted at any position within this region. Predicting breakpoint intervals on the basis of posterior probabilities allows the user to see which breakpoints can be located relatively precisely or which breakpoints are approximate.

The definition of uncertainty regions increases the reliability of predicted subtypes outside of these regions. For the default posterior probability threshold, less than $0.8\,\%$ of the positions located outside of breakpoint intervals and uncertainty regions were assigned to the incorrect subtype. For uncertainty regions, no parental strain can confidently be determined. But the graph of the posterior probabilities provides information about which subtypes are most closely related in these regions.

5.4 Restriction of the search space of the Viterbi algorithm

In section 3.2 (p. 35), a restriction of the search space of the Viterbi algorithm in jpHMM was presented in addition to the beam-search algorithm. As a first and very simple approach the determination of the start and end position of each query sequence in the reference genome (section 3.2.1, p. 35) was proposed. These positions define the region in the multiple sequence alignment the query sequence can be aligned to. This restriction of the search space of the Viterbi algorithm results in a reduction of the runtime of jpHMM, but only for fragmental sequences. Therefore, and since the results are identical to those of the original jpHMM without knowing the genome position this approach is not further evaluated in this section.

The focus of this section is on the second approach, proposed in section 3.2.2 (p. 36), namely the restriction of the Viterbi search space achieved by a pre-alignment of each query sequence to the given multiple sequence alignment: A set of sequences is selected from the multiple alignment and the query sequence is aligned to each of these sequences pairwisely using the alignment program BLAT [37]. Thus, each query sequence position is mapped to a certain set of columns in the alignment. These columns define a certain region in the alignment, called active alignment column interval, to which the respective sequence position can be aligned. Then, each sequence position can only be emitted by a state corresponding to a column within the active alignment column interval assigned to the respective sequence position.

In the following subsections, first, the parameters for the definition of active alignment column intervals are defined. Second, the accuracy of the BLAT alignments for HIV-1 and HBV sequences is evaluated and it is shown that BLAT is a suitable program for defining active alignment column intervals to restrict the search space of the Viterbi algorithm in jpHMM. Third, the accuracy of jpHMM based on these pre-defined active alignment column intervals is evaluated.

The runtime of jpHMM on the basis of pre-defined active alignment column intervals is analyzed in section 5.6 (p. 105) in which the runtime of all jpHMM versions developed in this thesis is compared.

5.4.1 Parameters

The workflow 3.5 in section 3.2.2 (p. 38) describes how active alignment column intervals are determined for each query sequence position. In this workflow, several parameters were described that will be specified in the following:

- **Definition of active columns in the given multiple sequence alignment:**
  According to Def. 3.4 (p. 39), an alignment column is an active column if it is a conserved (Def. 3.2, p. 38) and non-gap column (Def. 3.3, p. 38) with a certain number of adjacent conserved non-gap columns to the left and to the right. A column is a conserved column if a certain nucleotide is observed in this column in at least $C = 80\%$ of the sequences in the alignment. For a non-gap column, less than $G = 20\%$ of the sequences are allowed to show a gap. The number of required adjacent conserved non-gap columns to the left and to the right is 2.

- **Definition of active alignment column intervals:**
  $N \approx 100$ sequences were selected randomly from the alignment to build the database D for the BLAT alignments. An active alignment column interval is only assigned to a certain query sequence position if this position is aligned to at least $T = 50\%$ of the database sequences.

- **Extension of active alignment column intervals determined with BLAT:**
  Each active alignment interval determined with BLAT is extended by $K = 3$ active alignment columns if possible. At sequence ends such an extension is not required.

The parameters described in the first two items are estimated on the given HIV-1 alignment (section 5.1.1, p. 75) my maximizing the number of correctly aligned sequence positions in the test described below in section 5.4.2 (p. 96). For the extension of the active alignment column intervals determined with BLAT (item 3), several extension parameters K, $K = 0, \ldots, 3$, have been tested. For $K = 3$, the number of correctly aligned sequence positions was 100% on the basis of the given HBV alignment. This percentage could not be achieved for the HIV-1 sequences as it will also be described below.

5.4.2 Accuracy of pre-defined active alignment column intervals

Firstly, the accuracy of the pre-alignment of query sequences to the given multiple sequence alignment with BLAT was evaluated on the sequences in the multiple alignment. Each sequence in the multiple alignment, called query sequence here, was aligned to each selected database sequence with BLAT. For the sake of fairness, query sequences contained in the database were temporarily removed from the database. On the basis of these pairwise alignments, active alignment column intervals were defined for each query sequence position, as described in section 3.2.2 (p. 37).

The accuracy of the pre-alignment with BLAT was measured by the number of *correctly aligned* query sequence positions, i.e. query sequence positions for which the assigned active alignment column interval contains the original column of the position in the multiple sequence alignment.

Definition 5.12 (Correctly aligned sequence position)
A position of a sequence in the multiple sequence alignment is called correctly aligned *with the workflow 3.2.2.1 described in section 3.2.2 (p. 37) if the assigned active alignment column interval contains the original column of the sequence position in the alignment.*

Definition 5.13 (Accuracy of active alignment column intervals)
The accuracy of active alignment column intervals*,* acc_{AACI} *determined with the workflow 3.2.2.1 described in section 3.2.2 (p. 37), is given by the percentage of correctly aligned sequence positions:*

$$\text{acc}_{\text{AACI}} := \frac{\#\ \text{correctly aligned sequence positions}}{\#\ \text{sequence positions}} \tag{5.22}$$

The accuracy of the active alignment column intervals determined with BLAT was evaluated for the sequences in the given HIV-1 and HBV alignment. For the chosen parameters (section 5.4.1, p. 95), the assigned active alignment column interval contained the original position in the alignment (Table 5.5) for 100% of the HBV sequence positions and 99.95% of the HIV-1 sequence positions. In Table 5.5, for each virus type, the average, the minimum and the maximum length of the active alignment column intervals is given.

Table 5.5: Accuracy of the active alignment column intervals determined with BLAT for HIV-1 and HBV sequences. Additionally, for each virus type, the average and the minimum (Min.) and maximum (Max.) length of the active alignment column intervals is given.

HIV-1			HBV		
Accuracy	Interval length		Accuracy	Interval length	
	Average	Min. / Max.		Average	Min. / Max.
99.95%	81	7 / 1551	100 $\%$	24	6 / 79

The intervals for the HIV-1 sequences are larger than those for the HBV sequences since the HIV-1 alignment contains large regions of insertions or deletions where no active alignment column can be found. For example, both ends of the alignment are frayed with a large number of gaps so that a proper alignment is not possible. The minimum length of the intervals must be seven nucleotides due to the criterion that each active alignment column interval is extended by three active alignment columns. In HBV, the minimum length is determined by the positions at the sequence ends where such an extension is not possible. This is not the case for HIV-1 due to the large regions of gaps at the sequence ends.

The 0.05% of the HIV-1 sequence positions that were not able to be correctly aligned are located either in insert regions or are part of repeats of which one corresponds to an insertion in the alignment. Due to the criterion for active alignment columns (conserved and non-gap), an alignment for these positions is not possible. Besides, these positions showed up regions that were poorly aligned in the original alignment. Editing these regions in the original HIV-1 alignment could improve the results of the jpHMM recombination prediction for HIV-1.

5.4.3 Accuracy of jpHMM on the basis of pre-defined active alignment column intervals

For each query sequence position, the assigned active alignment column interval defines the columns in the alignment to which this sequence position is allowed to be aligned to. That is, the corresponding states in the model are allowed to emit the respective query sequence position. To guarantee that each active alignment column interval includes active states of all subtypes each active alignment column interval is extended such that each subtype contains at least a certain number M of consensus columns within this interval. A small number M of required consensus columns for each subtype strongly reduces the runtime of the Viterbi algorithm whereas larger values may guarantee better alignments. Taking into account both aspects M was set to 3.

On the basis of these pre-defined active alignment column intervals and with the beam-search algorithm (section 3.2.2.2, p. 42) as further restriction of the Viterbi search space, jpHMM was applied to all HIV-1 and HBV datasets described in section 5.1 (p. 75). Here, the HBV sequences were treated as linear genomes. The results of the recombination prediction were compared to the results of the original jpHMM version. For all sequences, the results of both methods were identical.

5.4.4 Discussion

The accuracy of the active alignment column intervals shows that BLAT is a suitable program for pre-aligning HIV-1 and HBV sequences to restrict the search space of the Viterbi algorithm in jpHMM. Additionally, it is a very fast program with an output that is easy to evaluate and incorporate in the jpHMM pipeline. Since the recombination prediction results of a jpHMM on the basis of these active alignment column intervals are identical to the results of the original model and as the runtime of the program is strongly reduced (section 5.6, p. 105), such a restriction of the Viterbi search space is recommended.

5.5 A modified jpHMM architecture with mute jump states

In section 3.3 (p. 47), a modification of the original jpHMM architecture (jpHMM_linear) was presented. By the introduction of mute jump states in the model, the number of jumps in the model was reduced to be linear in terms of the number of subtypes instead of being quadratic. Such a modification of the jpHMM architecture was necessary to allow the application of jpHMM to species with a large number of subtypes. A possible application is, for example, the detection of chimera in 16S rRNA sequences which will be a future project. In this section, the modified jpHMM including mute jump states will be applied to the HIV-1 and HBV sequences examined in this thesis. It will be shown that the accuracy of this new jpHMM version is comparable to the accuracy of the original jpHMM.

The runtime of jpHMM with a modified architecture (jpHMM_linear) and in combination with a pre-alignment of each query sequence to the given multiple sequence alignment is compared to the runtime of the original jpHMM in section 5.6. In that section, the runtime of all jpHMM versions developed in this thesis is evaluated.

Additionally, the runtime of jpHMM for an example alignment subdivided into a large number of subtypes is evaluated to demonstrate the necessity of the reduction of the number of jumps in the model for such applications.

5.5.1 Accuracy of the modified jpHMM with mute jump states

To compare the accuracy of the original jpHMM with that of the newly developed jpHMM, jpHMM_linear, a few modifications in the original model are necessary. In the original jpHMM, from each match state two types of jumps to another subtype are possible, namely a jump to the match state and a jump to the delete state in the successive column (section 2.2.2, Fig. 2.4). The ratio of the probabilities of these two jumps is determined by the ratio of the probabilities of the respective transitions within the subtype the jump leads to. For example, for the match state $M_{k,i}$ in column i in subtype $\mathcal{S}_k$, let t_{MM} be the transition probability from $M_{k,i}$ to the match state $M_{k,i+1}$ in the successive column $i+1$ and t_{MD} the transition probability from $M_{k,i}$ to the respective delete state $D_{k,i+1}$. Then, the ratio of the probability of a jump from $M_{k,i}$ to the match state $M_{l,i+1}$ in column $i+1$ in subtype $\mathcal{S}_l$ to the probability of a jump from $M_{k,i}$ to the respective delete state $D_{l,i+1}$ corresponds to the ratio of t_{MM} to t_{DD}. This can lead to the fact that jumps from a match state in a subtype to match states in two different subtypes are not equally likely. Such jump probabilities cannot be modeled in the modified jpHMM since from a match state jumps are only possible to the two mute jump states D_M and D_D which allow for jumps to match and delete states, resp.. Thus, to compare the two jpHMM versions, in both models, the jump probabilities from match states to match and delete states are set to be equal. This choice of jump probabilities will be discussed later again. But despite this adaption of the original model, the transition and jump probabilities in both models do not have to be equal for all columns. The reasons are

- the unequal number of jumps out of a certain state in both models,
- the fact that in the new model, the probability of a jump from one subtype to another is the product of the probability of a jump to a mute jump state and the probability of a jump from the mute jump state to the other subtype,
- the condition that the probabilities of all transitions and jumps out of a state must sum up to one.

Here, this will not be explained more detailed, since these facts do not have a big influence on the results (shown below). Additionally, as described in section 3.3.6 (p. 53), in regions where not all subtypes have consensus columns, in the original model, less jumps are possible than in the new model where in each column, a jump from each subtype to any other subtype is possible.

The accuracy of the new jpHMM, jpHMM_linear, was evaluated on the HIV-1 and HBV datasets described in section 5.1 (p. 75) and compared to the original jpHMM version (with parameters adapted to the new version). In total, 120 HIV-1 sequences (1113 breakpoints) and $1,116$ HBV sequences (4303 breakpoints) were evaluated. For HBV, the jpHMM version for circular genomes (section 3.4, p. 57) was applied.

For all tested datasets, the sensitivity and specificity of breakpoint intervals predicted with the original model based on the posterior probabilities of the subtypes were able to be reached with the new jpHMM. Thus, all breakpoints identified with the original jpHMM with parameters adapted to the new version (explained above) could be detected with the new jpHMM. But, some of them were predicted at different positions compared to the positions predicted with the original model. In total, the predicted position of 43 breakpoints was shifted. One reason for this shift is that in the original model, jumps at positions in the genome regions where some subtypes do not contain consensus columns are favoured due to a higher jump probability as explained above. This is not the case for the new model where at each position a jump to all subtypes is possible. Thus, for seven of the predicted breakpoints a shift of the breakpoint position to such non-consensus regions could be observed in the original model compared to the new model. The other shifted breakpoints are located in conserved regions, where a breakpoint cannot be predicted properly. The reason for this shift is that the probabilities of transitions and jumps out of a state are not identical in both models for all positions, as described above. The largest breakpoint shift comprised 50 positions and was located within a variable genome region with many insertions and deletions. Most of the shifts only comprised one or two sequence positions.

5.5.2 Accuracy of the modified jpHMM with mute jump states in combination with a pre-alignment with BLAT

The modified jpHMM containing mute jump states was applied to HBV using the circular version of the program. This circular jpHMM version is based on a pre-alignment of each query sequence to the given multiple sequence alignment with BLAT. Thus, the results of the previous subsection 5.5.1 show that the pre-definition of active alignment column intervals with BLAT to reduce the search space of the Viterbi algorithm is also applicable to the modified jpHMM architecture. This was also confirmed by its application to HIV-1. The recombination prediction results achieved by the new jpHMM combined with a restriction of the Viterbi search space on the basis of active alignment column intervals were identical to those of the new jpHMM without any further search space restriction.

5.5.3 Discussion

On the basis of the evaluated sequences, the new jpHMM with a modified architecture containing mute jump states appears to be a promising approach for datasets with a large number of subtypes as well as for the given HIV-1 and HBV datasets. The accuracy of the original and the new model is comparable, and the runtime is decreased enormously (section 5.6). A disputed point is the ratio of the jump probability from a match state to the mute jump state D_M which allows for jumps to match states and the probability of a jump to the mute jump state D_D which allows for jumps to delete states. Setting this ratio to one as it was done in this evaluation decreases the sensitivity and specificity of predicted breakpoint intervals by up to nine percentage points compared to the accuracy of the original model. Additionally, transitions between match states usually have a higher probability than transitions from match to delete states. Thus, choosing equally high probabilities for these two types jumps is not the best choice. One possibility is to adapt this ratio to the ratio of the respective transition probabilities within the respective subtype. But this would also result in unequal jump probabilities for different subtypes in the same column. The estimation of the jump probabilities for different types of jumps is subject of future analyses and the results must be carefully compared to the results of the original model. Probably, a modification of the jump probabilities in the original model so that the jump probabilities are equal for all subtypes will give better results with respect to the accuracy of predicted breakpoints.

One main difference between the two models is that in the new model jumps to all subtypes are allowed at any column. In case of non-consensus columns in certain subtypes, this is not possible in the original model. It is very possible that once a suitable ratio of the probabilities of jumps to match and delete states is found, the new model is more accurate in predicting breakpoint positions than the original model.

5.6 Comparison of the runtime of different jpHMM versions

In this section, the runtime of the newly developed jpHMM versions, i.e. the restriction of the Viterbi search space by a pre-alignment with BLAT (section 3.2, p. 36) and the modification of the jpHMM architecture (section 3.3, p. 47), is compared to that of the original jpHMM. First, the runtime is compared for the HIV-1 and HBV datasets presented in this thesis (section 5.1, p. 75). Then, in the second part, the runtime for datasets with a large number of subtypes is compared to show that a modification of the jpHMM architecture is necessary to apply jpHMM to such large datasets in the future.

5.6.1 Comparison of the runtime of different jpHMM versions

The runtime of the original jpHMM and the two approaches for restricting the search space of the Viterbi algorithm and reducing the complexity of the model were compared for HIV-1 as well as HBV sequences. The compared methods are again

1. the original jpHMM (section 2.2.2),
2. a jpHMM on the basis of predefined active states determined with BLAT (section 3.2.2),
3. a jpHMM with a modified architecture such that the number of jumps per column is linear in terms of the number of subtypes in the model (section 3.3) ,
4. the jpHMM with a modified architecture (3) on the basis of predefined active states determined with BLAT (combination of 2 and 3).

In all four approaches, the beam-search algorithm (section 2.2.2.4) was used as further restriction of the Viterbi search space. The results of the comparison are given in Table 5.6.

For HIV-1, the average jpHMM runtime for all sequences included in the three datasets D1, D2 and D3 (section 5.1.1.3) was determined. Thus, the evaluation comprised 120 nearly full-length ($\sim 9,500$ nt) HIV-1 genomes. As input alignment, the alignment described in section 5.1.1.1 was taken. It contains 14 subtypes with a length of about $10,000$ nt For HBV, the circular version of jpHMM was used. This version first extends the given multiple sequence alignment and query sequences as described in section 3.4. As test data, all sequences included in the datasets described in section 5.1.2.3 and 5.1.2.4 were taken, comprising 1116 sequences. The extended sequences had an average length of $4,200$ nt.

The input alignment is the extended HBV alignment described in section 5.1.2.2. It has a length of $7,371$ nt and consists of eight genotypes.

For HBV, only the runtime of the jpHMM versions on the basis of a pre-alignment with BLAT is compared, since such an alignment is necessary for applying jpHMM to circular genomes. As the original jpHMM version, the reimplemented jpHMM version including only the beam-search algorithm was used.

First, it can be seen in Table 5.6 that the average runtime of jpHMM for full-length HIV-1 sequences can be reduced from ≥ 10 minutes to about seven minutes due to the reimplementation of the source code. This runtime can be reduced by more than half by a restriction of the Viterbi search space by a pre-alignment with BLAT. For the jpHMM with a modified architecture, this restriction of the Viterbi search space does not have such a large effect, the runtime can be reduced by 98 seconds. But, compared to the original jpHMM, the average runtime for full-length HIV-1 sequences can be reduced by two-thirds to only 141 seconds. For HBV sequences, the modification of the jpHMM architecture does not have a big influence on the runtime of the program which may be the result of the small number of subtypes and the relatively short length of the sequences.

Table 5.6: Comparison of the runtime of four different jpHMM versions: 1.) the original jpHMM (jpHMM_orig), 2.) a jpHMM on the basis of predefined active states determined with BLAT (jpHMM_blat), 3.) a jpHMM with a modified architecture such that the number of jumps per column is linear in terms of the number of subtypes (jpHMM_linear), and 4.) a jpHMM with a modified architecture on the basis of predefined active states determined with BLAT (jpHMM_linear_blat). For each approach, the average runtime for all test sequences is given. For HBV, the jpHMM version for circular genomes is used. Thus, for HBV, only the jpHMM versions on the basis of a pre-alignment with BLAT are compared.

	# Subtypes	Runtime of			
		jpHMM_orig	jpHMM_blat	jpHMM_linear	jpHMM_linear_blat
HIV-1	14	7m18s	3m26s	3m59s	2m21s
HBV	8	-	1m21s	-	1m9s

5.6.2 Runtime for datasets with a large number of subtypes

In this section, the necessity of a modification of the model architecture for the application of jpHMM to species with a large number of subtypes like bacteria is shown on the basis of the runtime of the different jpHMM versions. In lack of well-defined alignments for species with a large number of subtypes, such as for 16S rRNA sequences, semi-artificial datasets with a large number of subtypes were created. For this purpose, the given HBV alignment was further subdivided into a larger number of subtypes. Additionally, only a certain part of the alignment with a length of $1,500$ nt was chosen. This length corresponds to the length of 16S rRNA sequences.

On the basis of this alignment, it will be shown that the modification of the jpHMM architecture to include mute jump states is suitable for the application of jpHMM to species with a large number of subtypes. In Table 5.7, the runtime of the different jpHMM versions is shown for such an alignment divided into $n = 35, 70$ and 100 subtypes. The compared methods are the original jpHMM (jpHMM_orig), the jpHMM with a restriction of the Viterbi search space with BLAT (jpHMM_blat) and the jpHMM with a linear number of jumps in terms of the number of subtypes (jpHMM_linear).

Table 5.7: Comparison of the average runtime of the original jpHMM (jpHMM_orig), of a jpHMM on the basis of a pre-alignment with BLAT to restrict the Viterbi search space (jpHMM_blat) and of a jpHMM with a linear number of jumps in terms of the number of subtypes (jpHMM_linear) for sequences of length 1,500 nt and different numbers ($n = 35, 70, 100$) of subtypes.

# Subtypes	Runtime of		
	jpHMM_orig	jpHMM_blat	jpHMM_linear
35	4m47s	1m37s	49s
70	21m10s	6m55s	1m54s
100	45m38s	13m25s	3m06s

As it can be seen in Table 5.7, the runtime of the original jpHMM increases immensely with a growing number of subtypes. For an alignment consisting of 100 subtypes, the average runtime for a sequence of length $1,500$ nt is more than 45 minutes. While the definition of active alignment column intervals based on a pre-alignment with BLAT already reduces the original runtime to a third, the effect of modifying the jpHMM architecture such that the number of jumps is linear instead of quadratic in terms of the number of subtypes is even much higher. The average runtime of 45 minutes can be reduced to only three minutes.

5.6.3 Discussion

The results show that a reduction of the runtime of jpHMM is necessary to apply the program to species with many subgroups. 45 minutes as average runtime for sequences of length $1,500$ nt is far too much to evaluate large datasets. As both modifications, the definition of active alignment columns as well as the modification of the jpHMM architecture, provide an accuracy that is comparable or even identical to that of the original method, the application of both (combined) modifications is highly recommended for future projects.

5.7 A jpHMM for circular genomes and its application to HBV

In section 3.4, a method for the application of jpHMM to recombination prediction in viruses with circular genomes was developed. The main idea behind this approach was to extend full-length query sequences at both sequence ends to allow a recombination prediction that takes into account dependencies between nucleotides at the 5' and the 3' end of the linearized sequences of circular genomes and is not biased against recombination breakpoints in these regions.

The hepatitis B virus is such a virus with a circular genome. This virus replicates via a pregenomic RNA, that is present in a linear form, using reverse transcription. The mechanism of recombination during replication has not been completely understood yet. Several hypotheses are possible as described in section 2.1.2 (p. 12). The main question is whether the recombination takes place when the genome is present in its linear or in its circular form. Applying the circular version of jpHMM to HBV sequences implies that we assume that recombination in HBV takes place when the genome is present in its circular form. This hypothesis was chosen to avoid the problems mentioned in section 3.4 (p. 57) for the application of a linear model to circular genomes, such as being biased towards recombination segments at the sequence ends and implicitly predicting breakpoints at the chosen origin for the sequence coordinates. In the case that this hypothesis is not true, the circular approach is not appropriate. Small recombinant segments at one end of the linear genome may not be detected due to the fact that in the circular version two breakpoints instead of one are required for the detection of such a segment. On the basis of the results of the recombination prediction, this will be discussed again at the end of this section.

First, the use of BLAT (section 3.2.2, p. 37) for aligning extended query sequences to the extended input alignment is discussed. Then, the accuracy of the circular jpHMM version for predicting recombinations in HBV sequences is evaluated on the basis of semi-artificial recombinant sequences. With this knowledge, the circular jpHMM is applied to all nearly full-length HBV sequences available from GenBank in December 2009. The frequency of pure genotypes and recombinants among these tested sequences is determined and the recombinants are further evaluated to identify *circulating recombinant forms* that have arisen from *independent recombination events*.

5.7.1 Settings

As described in chapter 3, section 3.4 (p. 57), the application of jpHMM to viruses with circular genomes starts with an extension of each full-length query sequence and the given multiple sequence alignment (Fig. 3.12 and Fig. 3.13, p. 59). The number n of nucleotides by which a query sequence is extended at both ends, was set to $n = 500$. This value was determined empirically but seems to be enough for a precise prediction of recombination breakpoints near the original sequence ends. The length of the prefix by which each sequence in the alignment is extended after the duplication of the alignment is also set to 500 nt. This extension is necessary to allow a complete alignment of an extended query sequence to the extended alignment if the chosen origin for the query sequence ends is identical to that of the sequences in the alignment.

In the next step, the extended query sequences are aligned to the sequences in the extended alignment with BLAT to define active alignment column intervals for each query sequence position (section 3.2.2, p. 36). Due to the duplication of the alignment (Fig. 3.13, p. 60) usually two alignments of a query sequence to a certain sequence in the alignment are possible. These two alignments are also contained in the BLAT output. Usually, the alignment with the highest score is chosen for further evaluation. But it may happen that for a certain sequence $S1$ in the alignment, the alignment of the query sequence to the first part of $S1$ has the highest score whereas for another sequence $S2$ in the alignment, the highest score is given by the alignment to the second part of $S2$. In this case, a certain query sequence position is, for example, aligned to column x in $S1$ and to column $x + l$ in $S2$, if l is the length of the alignment. Thus, the length of the interval of aligned columns corresponds to the length of the alignment, and consequently, no restriction of the Viterbi search space is achieved. To avoid such problems for each pair of aligned sequences, the left-most pairwise alignment with respect to the given multiple sequence alignment is chosen if the score of this pairwise alignment is not much lower than the highest alignment score achieved for this pair of sequences. In case that this condition is not fulfilled and the assigned active alignment column intervals are very large, still the beam-search algorithm is applied to reduce the Viterbi search space (section 3.2.2, p. 42).

As input alignment the extended HBV alignment described in section 5.1.2 (p. 77) is used. The jpHMM parameters are the parameters estimated in section 5.2.2 (p. 85, Eq. (5.20)) on the basis of this alignment.

5.7.2 Accuracy

The accuracy of the recombination prediction of the circular jpHMM version for HBV was measured by the accuracy of the predicted breakpoint intervals and the accuracy of the predicted genotypes at each sequence position. The accuracy of the predicted breakpoint intervals was assessed by the sensitivity (Def. 5.2, p. 82), i.e. the number of true breakpoints that could be detected with the predicted breakpoint intervals, and the specificity of the predicted breakpoint intervals (Def. 5.1, p. 82), i.e. the number of breakpoint intervals that contain a true breakpoint. The accuracy of the predicted genotypes at each sequence position was measured by the ratio of the sequence positions that were assigned to the correct genotype to the total number of sequence positions. Only sequence positions located outside of breakpoint intervals and uncertainty regions were taken into account. Breakpoint intervals and uncertainty regions were defined on the basis of the posterior probabilities for different thresholds $t_{\text{BPI}} = t_{\text{UR}} \in \{0.9, 0.95, 0.99, 0.9999\}$.

For the evaluation of the accuracy of the recombination prediction, different datasets were examined. First, the accuracy for dataset D0_HBV (section 5.1.1, p. 76) that comprises semi-artificial recombinants with breakpoints at randomly chosen positions was determined. This is the dataset the jpHMM parameters for HBV have been estimated on as described in section 5.2.2 (p. 81). The results are given in Table 5.8. Second, the accuracy was evaluated for the datasets D1_HBV, D2_HBV and D3_HBV (section 5.1.1, p. 76), comprising semi-artificial recombinants with fixed breakpoint positions (D1: at every 1000th position, D2: alternately at every 500th and 1500th position, D3: alternately at every 300th and 1500th position). The results for these datasets are given in Table 5.9.

As for HIV-1 (section 5.3, p. 87), as default threshold $t_{\text{BPI}} = t_{\text{UR}} = 0.99$ is chosen since it provides the best trade-off between the average length and the accuracy of the predicted breakpoint intervals. For this default threshold, the specificity of the predicted breakpoint intervals is at least $98\,\%$ for all tested datasets. For the datasets with fixed breakpoint positions (D1, D2 and D3), the sensitivity is about as high as the specificity for this threshold. But for dataset D0_HBV with breakpoints at random positions the sensitivity of the predicted breakpoint intervals differs from the specificity considerably (Table 5.8). Taking into account only breakpoints that are located outside of uncertainty regions the sensitivity can be increased to $83.04\,\%$ for the default threshold. But this value is still much lower than the specificity ($99.12\,\%$). The reason is that in some sequences of this dataset recombination segments of a very short length (the shortest segment has a length of 2 nt) exist. Using the currently chosen jump probability $jp = 10^{-07}$, jpHMM is not able to detect segments of such short length. But a higher jump probability may lead to a higher sensitivity, but also to a frequent jumping within the model which could result in a decreased specificity. The

Table 5.8: For different thresholds t_{BPI} (column 1), the accuracy of the predicted breakpoint intervals for dataset D0_HBV is given in terms of the sensitivity (column 4) and specificity (column 5) of the breakpoint intervals. Additionally, the percentage of sequence positions outside of uncertainty regions and breakpoint intervals that are classified correctly, i.e. assigned to the correct subtype, is determined (column 6). For each threshold, the average (column 2) as well as the minimum and maximum length (column 3) of the breakpoint intervals is shown.

Dataset D0_HBV (breakpoints at random positions in the genome)

Threshold	BPI length		Accuracy		% Positions
t_{BPI}	Average	Min. / Max.	Sens.	Spec	Correctly classified
0.90	22.50	1 / 159	71.98	90.20	99.81
0.95	26.58	1 / 172	77.52	95.61	99.84
0.99	33.26	1 / 200	80.09	99.12	99.86
0.9999	51.76	1 / 292	78.95	99.82	99.87

results for the datasets with fixed breakpoint positions show that for sequences that do not contain very short recombination segments the sensitivity and the specificity of the predicted breakpoint intervals is very high. Additionally, the percentage of sequence positions located outside of uncertainty regions and breakpoint intervals is very high. For the default threshold, $\leq 0.15\,\%$ of the positions are classified incorrectly, which corresponds to only 5 nt in a sequence of length $3,200$ nt.

5.7.3 Application to real-world HBV genomic sequences

The high accuracy of the recombination prediction for semi-artificial recombinants demonstrates that the circular version of jpHMM is a suitable and powerful tool for recombination prediction in HBV sequences. Therefore, it was applied to all nearly full-length ($\geq 3,000$ nt) HBV genomic sequences that were available in GenBank in December 2009. In total, 2918 sequences were evaluated of which 339 were used to build the input alignment (section 5.1.2, p. 78).

5.7.3.1 Number of recombinants and pure genotypes in GenBank

588 of the 2918 sequences were predicted as recombinant sequences, the rest as pure genotypes. Thus, about $20\,\%$ of the full-length HBV sequences stored at GenBank are recombinant sequences. For 54 of the 588 predicted recombinant sequences, at least one uncertainty

Table 5.9: For different thresholds t_{BPI} (column 1), the accuracy of the predicted breakpoint intervals for dataset D1_HBV, D2_HBV and D3_HBV, is given in terms of the sensitivity (column 4) and specificity (column 5) of the breakpoint intervals. Additionally, the percentage of sequence positions outside of uncertainty regions and breakpoint intervals that are classified correctly, i.e. assigned to the correct subtype, is determined (column 6). For each threshold, the average (column 2) as well as the minimum and maximum length (column 3) of the breakpoint intervals is shown.

Dataset D1_HBV (1000/1000 nt)

Threshold	BPI length		Accuracy		% Positions
t_{BPI}	Average	Min. / Max.	Sens.	Spec	Correctly classified
0.90	22.64	1 / 408	82.50	84.46	99.89
0.95	27.34	1 / 422	90.18	90.18	99.95
0.99	34.86	2 / 432	98.84	98.84	100
0.9999	52.43	5 / 476	100	100	100

Dataset D2_HBV (500/1500 nt)

Threshold	BPI length		Accuracy		% Positions
t_{BPI}	Average	Min. / Max.	Sens.	Spec	Correctly classified
0.90	25.30	1 / 297	80.63	82.32	99.85
0.95	30.73	1 / 358	89.20	89.20	99.91
0.99	39.98	1 / 429	98.13	98.13	99.99
0.9999	58.49	1 / 482	100	100	100

Dataset D3_HBV (300/1500 nt)

Threshold	BPI length		Accuracy		% Positions
t_{BPI}	Average	Min. / Max.	Sens.	Spec	Correctly classified
0.90	23.16	1 / 196	90.36	94.05	99.67
0.95	27.44	1 / 243	95.09	96.82	99.80
0.99	33.80	1 / 338	97.05	98.82	99.85
0.9999	53.03	4 / 427	98.21	100	99.94

region was defined in the predicted recombination. These recombinants were excluded from further analysis of the recombinant sequences.

The most strongly represented pure genotype is genotype C with 1182 sequences which corresponds to $40.51\,\%$ of all tested real-world HBV sequences (Table 5.10). The genotype with the lowest number of sequences is genotype G with 21 sequences which corresponds to only $0.72\,\%$ of the tested sequences. The frequency of all genotypes is presented in Table 5.10.

All genotypes but genotype H are involved in the recombination events detected with jpHMM. The predicted recombinant sequences will be analyzed and discussed in the following subsection 5.7.4.

Table 5.10: The number of sequences for each genotype and their percentage among all tested HBV sequences.

Genotype	# Sequences	in %
A	400	13.71
B	70	2.40
C	1182	40.51
D	399	13.67
E	176	6.03
F	58	1.99
G	21	0.72
H	24	0.82
$\sum$	2330	79.85

5.7.3.2 Runtime of the circular jpHMM

The average runtime of the circular jpHMM for the tested, nearly full-length HBV sequences was 47.8 seconds.

5.7.4 Proposal for a classification system for recombinant forms of HBV

Many of the predicted recombinant HBV sequences share the same recombination pattern and almost identical positions of recombination breakpoints. The aim of this section is to group together recombinant sequences that may stem from the same recombination event in order to identify *recombinant forms* (Def. 5.14, p. 116) that are very likely to have arisen from *independent* recombination events. Such recombinant forms that have been *established* in the population will be called *circulating recombinant forms* (CRF) of HBV. Here, the main focus of interest is on the definition of *really independent* recombinant forms rather than the identification of *all* existing recombinant forms.

In contrast to HIV, for HBV, a classification system including the definition of circulating recombinant forms does not exist. Usually, recombinant forms are defined as new subgenotypes based on the sequence similarity (section 2.1.2, p. 13). This can confound recombination detection tools like jpHMM to such a degree that they are unable to distinguish between two genotypes, if one of the genotypes is included as a recombinant form in a subgenotype of the other genotype. Also, such a mixing of genotypes and recombinants of them will make it more difficult to understand the viral evolution and epidemiology. Therefore, a classification system that clearly distinguishes pure genotypes from recombinant forms is necessary. In the following section a classification system for recombinant forms is proposed: Initially, the predicted recombinants are scanned for identical recombination patterns and similar locations of recombination breakpoints to cluster recombinants that may stem from the same recombination event. Subsequently, for each remaining recombinant form, the corresponding sequences are compared in order to identify sequences from epidemiologically unlinked samples among them. Recombinant forms that are represented by at least two unrelated samples define a *circulating recombinant form* of HBV (Def. 5.20). The condition of at least two unrelated samples accounts for the fact that only recombinant forms that are able to survive are of interest. Recombinant forms that occur only once or only in related samples, can, for example, be a result of incorrect sequencing or a lacking survival fitness compared to other recombinant forms. These recombinant forms are called *unique recombinant forms* (URF).

The problem of identifying independent recombinant forms has also been addressed by a few other studies [90, 23, 106] but apparently none of these studies took into account the circularity of the genome. These studies are presented below in subsection 5.7.6.

In the following, all sequence information such as genomic positions and recombination patterns is given based on reference genome numbering (section 5.1.2, p. 77).

5.7.4.1 Clustering of recombinant forms

The 534 predicted recombinant HBV sequences were clustered by the recombinant forms (Def. 5.14) they represent in order to identify recombinant forms among them that stem from independent recombination events. For this purpose, first, the set of recombinant sequences was searched for *equivalent recombinant forms* using the following definitions:

Definition 5.14 (Recombinant form)
A recombinant form RF *is a recombination pattern (Def. 5.5) with recombination breakpoints at certain positions (or in certain intervals) in the (reference) genome that has been observed in at least one genomic sequence.*

Definition 5.15 (Equivalent recombinant forms)
Two recombinant forms RF_1 *and* RF_2 *are called* equivalent, *if*

1. *their recombination patterns are identical and*
2. *for each breakpoint interval* B *in* RF_1 *in between two genotypes* $\mathcal{S}_1$ *and* $\mathcal{S}_2$*, a breakpoint interval in* RF_2 *exists that has the same preceding (*$\mathcal{S}_1$*) and successive genotype (*$\mathcal{S}_2$*) and borders on or overlaps with* B *by at least one base, and vice versa.*

Two equivalent recombinant forms RF_1 *and* RF_2 *are notated*

$$RF_1 \simeq RF_2. \tag{5.23}$$

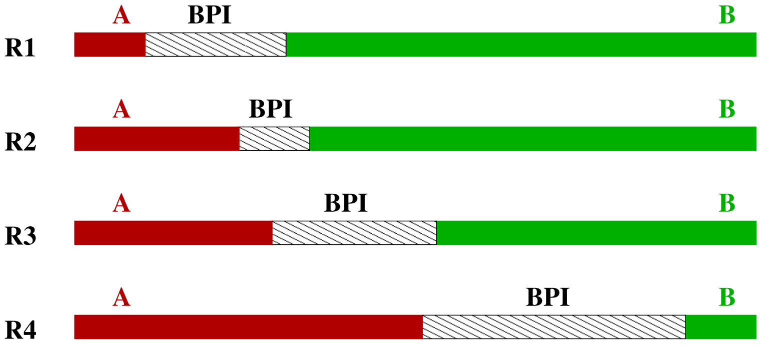


Figure 5.1: Four example recombinants $R1, \ldots, R4$ of genotypes A (red) and B (green) with identical recombination patterns and partly overlapping breakpoint intervals (BPI).

Second, recombinants that represent equivalent recombinant forms were filtered out iteratively from the set of recombinants and clustered such that each remaining recombinant sequence represents a set of recombinants with equivalent recombinant forms. This iteration is described in Algorithm 5.16.

Algorithm 5.16 (Algorithm for filtering out equivalent recombinant forms of HBV) Let $\mathcal{R} = \{R_1, \ldots, R_k\}$ be the set of predicted recombinants, $k = 534$, and $RF_i := RF(R_i)$ the recombinant form represented by R_i.

```
FOR $i = 1, \ldots, k$
  IF $R_i \in \mathcal{R}$
    define cluster $\mathcal{C}_{RF_i} := \{R_i\}$
    FOR $j = i + 1, \ldots, k$
      IF $R_j \in \mathcal{R}$ and $RF_j \simeq RF_i$
        1. remove $R_j$ from $\mathcal{R}$ :     $\mathcal{R} = \mathcal{R} \setminus R_j$
        2. add $R_j$ to cluster $\mathcal{C}_{RF_i}$ defined by the recombinant form $RF_i$ :
           $\mathcal{C}_{RF_i} = \mathcal{C}_{RF_i} \cup \{R_j\}$
```

Obviously, the result of this iterative algorithm, i.e. the clustering of the recombinants, depends on the order of comparing the predicted recombinant forms. For example, comparing iteratively the four recombinant sequences shown in Figure 5.1, can result in the following clusters:

1. $\mathcal{C}_{RF_3} = \{R1, R2, R3, R4\}$, comparing the sequences in the order $R3, R4, R2, R1$, or

2. $\mathcal{C}_{RF_2} = \{R1, R2\}, \mathcal{C}_{RF_4} = \{R3, R4\}$, comparing them in the order $R4, R3, R2, R1$.

In the second case, the recombinants $R1$ and $R2$ and the recombinants $R3$ and $R4$ are clustered respectively although $R1$ and $R2$ are both equivalent to $R3$. The reason is that $R4$ is not equivalent to $R1$ and $R2$ respectively. Hence, albeit being sensitive to the order of comparing the recombinant forms, algorithm 5.16 ensures that in every two clusters $\mathcal{C}_i \neq \mathcal{C}_j$, there are two recombinant forms $R_1 \in \mathcal{C}_i$ and $R_2 \in \mathcal{C}_j$ such that R_1 and R_2 are not equivalent. This is also shown in Figure 5.2. In this example, at least two clusters of recombinants, e.g. $\mathcal{C}_{RF_2} = \{R1, R2, R3\}$ and $\mathcal{C}_{RF_4} = \{R4, R5\}$, are built even if all recombinants seem to be very similar and possibly stem from the same recombination event.

A further comparison of the identified clusters is necessary to analyze if, for example, $R1$ and $R5$ in Figure 5.2 or $R1$ and $R4$ in Figure 5.1 can be clustered or if the distance of their breakpoint intervals is significant.

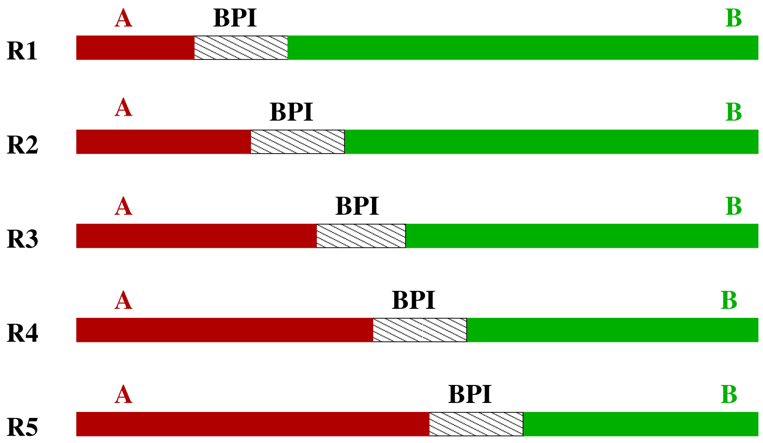


Figure 5.2: Five example recombinants $R1, \ldots, R5$ of genotypes A (red) and B (green) with identical recombination patterns and partly overlapping breakpoint intervals (BPI).

5.7.4.2 Evaluation of clusters of recombinant forms

In this section, recombinants representing different clusters of equivalent recombinant forms that share the same recombination pattern (Def. 5.5, p. 83) but do not fulfill the criterion of overlapping breakpoint intervals (Def. 5.15, p. 116) for at least one breakpoint interval are reevaluated. For each pair of such recombinants and each pair of non-overlapping, but possibly corresponding breakpoint intervals, the segments in the genomic region between the two shifted breakpoint intervals are examined in both recombinants to decide whether it is possible that the two breakpoint intervals have arisen from the same recombination event. This scenario is shown in Figure 5.3. Two recombinant sequences have the same recombination pattern (AB) but the breakpoint intervals in both sequences do not overlap. The two segments in-between the two breakpoint intervals are labeled s1 in the first sequence and s2 in the second sequence. s1 is assigned to genotype B and s2 to genotype A.

For both segments, the distance to all sequences included in both genotypes (Def. 5.17) involved in the two compared recombination events, genotype A and B in this example, as well as the distance to each of the two genotype as a whole (Def. 5.18) are calculated in the respective genomic region.

Definition 5.17 (Distance sequence-sequence) *Let $S1$ and $S2$ be two nucleotide sequences and $A(S1, S2) = (a_{i,j})_{1 \leq i \leq 2, 1 \leq j \leq l}$ a pairwise alignment of $S1$ and $S2$ with length l.*
The distance $d(S1, S2)$ *of $S1$ to $S2$ is defined as the number of mismatches in $A(S1, S2)$:*

$$d(S1, S2) = \#\{\textit{column } a_j \mid a_{1,j} \neq a_{2,j}, 1 \leq j \leq l\} \tag{5.24}$$

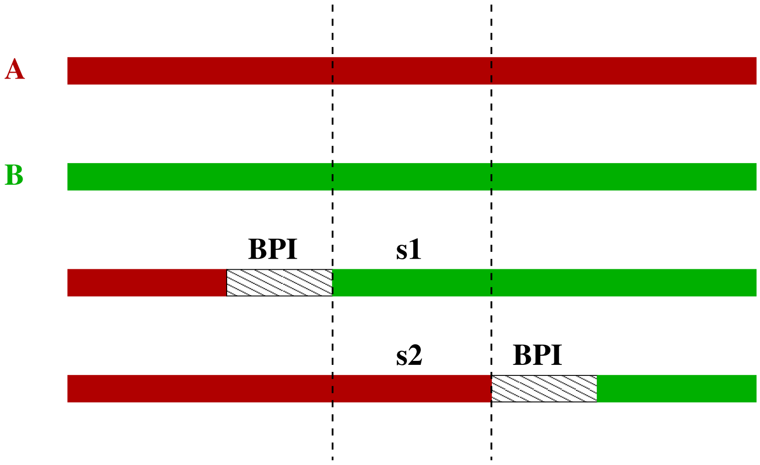


Figure 5.3: Comparison of segments s1 and s2 in between two breakpoint intervals in two different recombinants. Genotype A is represented by red and genotype B by green color.

Definition 5.18 (Distance sequence-genotype) *Let* S *be a nucleotide sequence and* $\mathcal{S} = \{S_1, \ldots, S_k\}$ *be a genotype consisting of sequences* $S_1, \ldots, S_k$*.*
The distance $d(S, \mathcal{S})$ *of* S *to* $\mathcal{S}$ *is defined as the average distance of* S *to all sequences in* $\mathcal{S}$ *(Def. 5.17):*

$$d(S, \mathcal{S}) = \frac{\sum_{j=1}^{k} d(S, S_k)}{k} \tag{5.25}$$

Then, the two breakpoint intervals are defined as having arisen from the same recombination event if, for one of the two segments,

1. the distance to the genotype that was assigned to this segment is higher than the distance to the second genotype, or
2. in the genotype that was not assigned to this segment, a sequence exists of which the distance to this genotype is as high as the distance of the respective segment to this genotype.

For the given example this means that the two breakpoint intervals are considered as having arisen from the same recombination event if the distance of s1 to genotype B is higher than the distance to genotype A (or vice versa for segment s2), or if in genotype A a sequence exists of which the distance to genotype A is higher than the distance of s1 to A (or the same for genotype B and s2).

The given criteria are chosen for the following reasons. In case that condition 1 holds, one can assume that the genotype assignment of jpHMM may be incorrect for this segment. If condition 2 is fulfilled the distance of a sequence to the genotype it belongs to can be a hint that also the examined segment can be assigned to this genotype even if it was assigned to the other genotype by jpHMM. If one of these conditions holds for all breakpoint intervals in the two compared sequences, these sequences are defined as equivalent recombinant forms and are clustered. For each remaining cluster, a representative sequence is chosen. These representative recombinants are considered to define recombinant forms that arose from independent recombination events.

5.7.4.3 Definition of circulating recombinant forms of HBV

Each of the representative recombinants for the clusters of equivalent recombinant forms defined in the two previous clustering steps may represent a *circulating recombinant form* (CRF), i.e. a recombinant form that has been established in the population. Only recombinant forms that are represented by at least two unrelated samples are accepted as CRFs (Def. 5.20). In lack of information whether two samples are related or not, we assume that two sequences are not directly epidemiologically related if they differ by at least 0.5% of their positions. Such two sequences will be termed *unrelated* sequences in this thesis (Def. 5.19).

Definition 5.19 (Unrelated HBV genomic sequences)
Two HBV genomic sequence $S1$ *and* $S2$ *are termed* unrelated *if the number of mismatches and gaps introduced in* $S1$ *in the pairwise alignment of* $S1$ *and* $S2$ *comprise at least* 0.5% *of the length of* $S1$*. (The lengths of gaps are not taken into account.)*

Definition 5.20 (Circulating recombinant form of HBV)
A circulating recombinant form *of HBV is a recombinant form that is represented by at least two unrelated (Def. 5.19) HBV genomic sequences.*

Recombinant forms that do not define a CRF are called *unique recombinant forms* to accentuate that they have been observed, up to now, in only one or in several related samples. This condition is similar to the condition for the definition of a CRF in HIV where a recombinant form must be observed in at least three full-length sequences or two full-length sequences and a fragmental sequence obtained from epidemiological unlinked patients [80]. Since the genotypes presented in a fragment of a sequence do not need to reflect the whole variety of genotypes involved in the full-length sequence the presence of a third fragmental sequence is neglected and only two full-length sequences representing the same recombinant form are required for the definition of a CRF.

5.7.4.4 Results

Among the 534 recombinant sequences 17 CRFs and 28 URFs are defined. The number of sequences assigned to each CRF varies greatly (Table 5.11). Six of the CRFs are only represented by two sequences. The CRF with the largest number of assigned sequences, a recombination of genotypes B and C, contains 385 sequences. The predicted recombination pattern for the representative sequence for this CRF (D00330) is plotted in Figure 5.4. In Table 5.11, the predicted recombination patterns of all CRFs are presented. The sequence of genotypes is given based on reference genome numbering, which means that, for example, AGA could be AG using a different start position in the genome. For each recombination pattern listed in the table, the number of CRFs that share this recombination pattern is given, and for each of these CRFs, the GenBank accession number of the chosen representative sequence and the number of sequences assigned to this CRF are indicated.

Table 5.11: Predicted recombination patterns for CRFs of HBV based on reference genome numbering. For each predicted recombination pattern (column 1), the number of CRFs sharing this pattern (column 2), the reference sequence(s) of the respective CRFs (column 3) and the number of sequences included in the CRF (column 4), are given.

Pattern	Frequency	Reference sequence(s)	# Sequences
AC [1]	1	AB231908	3
ADA	1	AF418674	4
ADADADA	1	AF418685	8
AG [1]	1	EF464099	2
BCBCB	1	FJ386674	5
BCB	2	D00330, EU939627	385, 2
CA	1	AY057947	2
CBC	2	EU796069, EU939628	26, 4
DAD	3	AF418681, AY236161, X80925	6, 2, 2
DC [1]	2	AY057948, AF461043	6, 40
DCD	1	FJ562223	3
GFG	1	EF464097	2
$\sum$	17		502

In the identified CRFs, all HBV genotypes but genotypes E and H have been involved. All CRFs are recombinants of two genotypes. Most of them contain two recombination breakpoints but in the CRF represented by sequence FJ386674, four breakpoints and in the

---

[1] breakpoint interval around origin of reference genome

CRF represented by sequence AF418685, six breakpoints have been observed.

Most URFs were observed only in one sequence but two of the 28 URFs are represented by two and four sequences respectively that do not satisfy our definition for being considered as epidemiologically unrelated sequences. As an example for a URF, the predicted recombination for sequence DQ078791, a recombination of genotypes A, C and G, is given in Figure 5.5.

The 17 CRFs and the 28 URFs identified with jpHMM in the set of all nearly full-length HBV genomic sequences downloaded from GenBank in December 2009 are published at

`http://jphmm.gobics.de/hbv_recombinant_forms.html`

and

`http://jphmm.gobics.de/hbv_unique_recombinant_forms.html`.

For each CRF and URF, the recombination of the representative sequence predicted with jpHMM is plotted using the software package Circos [44] and a list of sequences clustered with this sequence is provided.

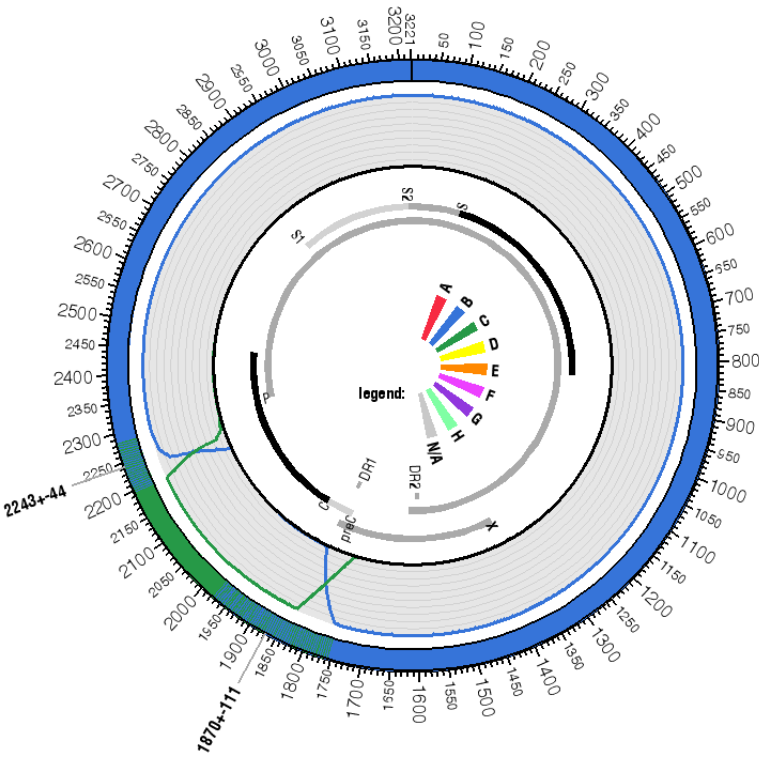


Figure 5.4: Plot of the predicted recombination for the representative sequence D00330 for the CRF with the largest number (N=385) of assigned sequences.

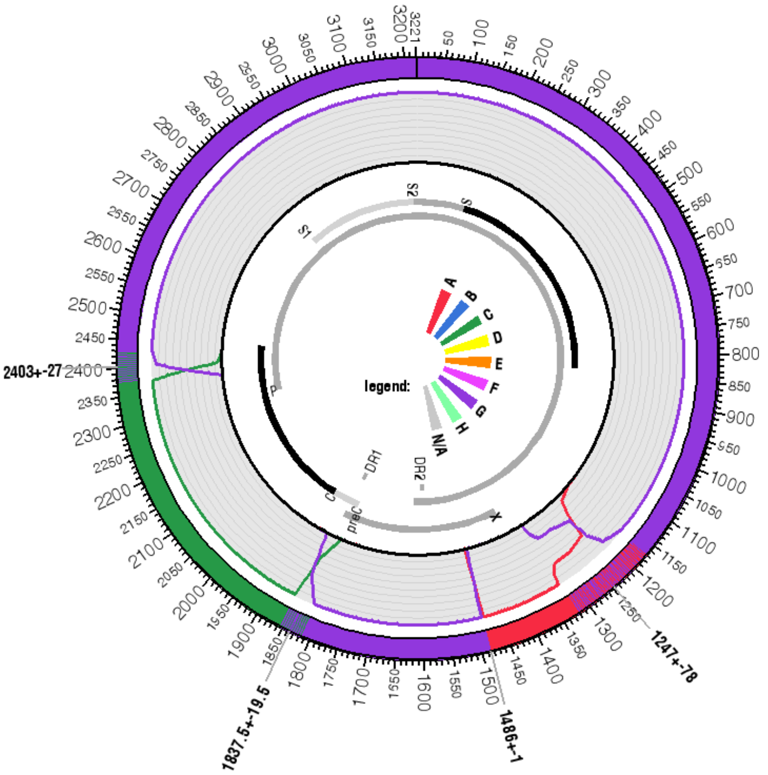


Figure 5.5: Plot of the predicted recombination for the URF represented by sequence DQ078791.

5.7.5 Comparison to other studies

Independent or circulating recombinant forms of HBV have also been identified in a few other studies. In 2005, Simmonds *et al.* [90] defined 24 phylogenetically independent recombinant forms on the basis of different genotype compositions or distinct breakpoints. But these recombinant forms also include recombinant forms that are only represented by one sequence because circulating recombinant forms were not distinguished from unique recombinant forms. In the PhD thesis of Fang Fang [23], which apparently was never published in a scientific journal, about 380 sequences were tested for recombination and nine CRFs were identified. Both studies are based on linearized genomes and the circularity of the genome has not been taken into account. Very recently, Lin *et al.* [106] evaluated 791 full-length HBV sequences. 61 recombinants from nine putative recombination events were identified but only two of the recombination events are represented by more than one sequence. In this study, RDP3 [56] and Simplot [53] were used as recombination detection tools. In the publication it is mentioned that for RDP3, the default parameters are used except when sequences were circular. This description is very unclear and no evidence could be found that all sequences have been treated as circular genomes. Additionally, in the RDP3 manual, no description of the method for circular genomes could be found. Thus, it is not clear whether the method used for circular genomes is different from that for linear genomes or if only the sequence ends are predicted as breakpoints if the two genotypes predicted at both ends of a sequence are different.

For most of the recombinants that were included in our as well as in one of the datasets in the presented publications, the recombinant forms predicted with jpHMM correspond to the published recombinant forms. For two recombinants published in [23], the number of predicted breakpoints was different and for five sequences, an additional genotype was predicted in [23] compared to our prediction (e.g. CAB instead of CB). Two of these five sequences were predicted as pure genotypes in our analysis. Compared to the recombinant forms published in [90] and [106], for one sequence, two instead of four breakpoints were predicted, and for another sequence, the recombination pattern BACB was predicted instead of BCBCB. Both recombinant forms were defined as unique recombinant forms. Additionally, in both publications, the sequence AY161146 (and AY161145 in [106]) was clustered with the sequence AY161141. In our evaluation, AY161145 and AY161146 represent the same URF and AY161141 is part of a CRF. The recombination patterns of both recombinant forms are very similar but do not fulfill the criterion (introduced above) for being clustered as one CRF. These results will be discussed below in subsection 5.7.6.

5.7.6 Discussion

The results of the jpHMM recombination prediction for semi-artificial HBV recombinants show that circular jpHMM is a very accurate tool for recombination detection in HBV sequences. With a runtime of only about 45 seconds, it is also applicable in large-scale studies. These facts and an output of the predicted recombination in a circular form makes it a recombination detection tool that HBV researchers will highly benefit from.

The differences between the results of the jpHMM recombination prediction and those published in [90, 23, 106] may be explained by the fact that in our study a method was used that explicitly takes into account the circularity of the genome (it is not clear how or if the circularity was taken into account in [23]). Recombinant segments that were missing in our prediction compared to the published results may be a consequence of the fact that small recombinant segments at one end of the linearized genome cannot be detected with the circular model. Using a linear model, only one breakpoint is necessary to predict such a segment whereas in the circular model a second breakpoint at the sequence end has to be predicted. Depending on the jump probability and the number of mismatches between the two genotypes involved, such a segment may not be detected with the circular version of jpHMM. As a consequence, the circular jpHMM may tend to predict pure genotypes or recombinants with less breakpoints compared to the linear model. On the other hand, short recombination segments at sequence ends in the published predictions that were not predicted with jpHMM can be a result of the usage of a linear method for recombination prediction in circular genomes as described in section 3.4. Such a recombination prediction is biased towards (incorrect) recombination segments at the sequence ends. These discrepancies must be further analyzed. A first approach can be the application of the original, linear version of jpHMM to linearized sequences followed by a comparison of the results of both jpHMM versions.

The main goal of this section was to evaluate the genetic diversity of HBV and to identify circulating recombinant forms (CRFs) that have arisen from independent recombination events. About 20% of the sequences published in the GenBank database were predicted as recombinants. Among these recombinants, 17 CRFs and 28 URFs have been identified. The CRF with the largest number of sequences is represented by sequence D00330 (Fig. 5.4). It contains 385 sequences which corresponds to about 72% of the recombinants (that do no include an uncertainty region).

The sequences clustered into one CRF most probably have arisen from the same recombination event. But, as it can be seen at

`http://jphmm.gobics.de/hbv_recombinant_forms.html`,

some of the CRFs and URFs share the same recombination pattern and similar locations of breakpoints, but were nevertheless defined as independent recombinant forms. For these sequences, it must be reevaluated if the shifted, non-overlapping breakpoint intervals really define distinct recombination events. This evaluation will be carried out in close collaboration with Prof. Paul Dény from INSERM in Lyon, France, since more biological expertise is necessary to answer these questions.

Chapter 6

Conclusions

Accurate genotyping methods that are able to identify recombination breakpoints are needed for monitoring the molecular epidemiology of viruses and tracking the viral evolution. During the last years, a large variety of tools for recombination analysis in viruses has been developed [78]. One of the most accurate methods in detecting recombination breakpoints in HIV-1 sequences is jpHMM [87, 109] which can be accessed by an easy-to-use web interface at `http://jphmm.gobics.de`. On the basis of comparing single representative HIV-1 sequences, recombination breakpoints identified by jpHMM were found to be significantly more accurate than breakpoints defined by methods that are traditionally used.

The extension of jpHMM described in this thesis includes interval estimates of breakpoints and a tagging of regions in which the model is uncertain about the predicted subtype. This extension improved the reliability of the predicted recombination immensely [85]. Less than $1\,\%$ of all positions outside uncertainty regions and breakpoint intervals were classified incorrectly. The definition of uncertainty regions may also help researchers to avoid drawing wrong conclusions based on doubtful, uninformative regions, such as the postulation of a new CRF. Additionally, the graph of the posterior probabilities provides information about which subtypes are most closely related in these regions. The length of predicted breakpoint intervals shows how precisely jpHMM can locate the breakpoint. For example, in conserved genomic regions, a precise prediction of breakpoint positions is impossible. Breakpoints located within such regions can be predicted at any position within this region which is reflected by a large breakpoint interval comprising the whole region.

Due to its high accuracy, jpHMM became a widely used tool for recombination detection in HIV-1 sequences. It was used in several studies such as subtype prevalence studies [35, 82], the detection and analysis of intersubtype recombinants [11, 103, 19, 10, 18] and even the definition of a new CRF [81]. Leading HIV experts from the Los Alamos National Laboratory used, in close collaboration with us, jpHMM, among other tools, in a large-scale resubtyping analysis [108] of the Los Alamos HIV Sequence Database [49]. Also, some of the reference breakpoints in known CRFs presented at this database were determined with jpHMM.

During this thesis, two modifications of jpHMM were achieved to reduce the runtime of the program. First, each query sequence is aligned to the given alignment sequences with BLAT defining the respective genomic position of each query sequence position. Second, a modification of the model architecture reduces the number of jumps within the model to be linear instead of quadratic in terms of the number of subtypes in the alignment. The realization of both modifications in combination with a reimplementation of the jpHMM source code lead to a reduction of the average runtime for nearly full-length HIV-1 sequences from ≥ 10 minutes to only 2 minutes and 21 seconds. This reduction is considerable. But as mentioned in the introduction, currently more than $2,300$ full-length and more than $330,000$ fragmental HIV-1 sequences are available in the Los Alamos HIV Sequence Database and the number is increasing rapidly. With the newly implemented jpHMM version, the runtime alone for all $2,300$ full-length sequences is already 90 hours. Even when running jpHMM simultaneously on several computers the runtime for all sequences including fragmental sequences could be too long for large-scale analyses.

To further reduce the runtime of the program, additional modifications of the model are conceivable. For example, in the current model, each column in the alignment corresponds to up to k match states, if k is the number of subtypes. Each of these states represents the nucleotide composition of the corresponding column in the respective subtype. In the case of (nearly) uninformative sites, which are represented by (nearly) identical nucleotide compositions of the respective column(s) in all subtypes, such a distinction of the subtypes in the model is not necessary. Additionally, at uninformative sites like conserved genomic regions, a precise prediction of recombination breakpoints is not possible and not even desirable. Therefore, at an uninformative site, only one match state or, if, for example, one subtype differs significantly from the other subtypes, two match states are needed to model the alignment of the query sequence to the given alignment at this position. Such a reduction of the number of models to be distinguished at uninformative sites could considerably reduce the number of states in the model.

But, nevertheless, the reduction of the runtime as well as the memory of the program that is achieved by the reduction of the number of jumps within the model, is large enough to enable the application of jpHMM to species with a large number of subtypes as it is shown in section 5.6. A prospective project is the application of jpHMM to chimera detection in 16S rRNA sequences. Chimeric artifacts of 16S rRNA sequences can be generated during PCR amplification when mixed bacterial populations are studied, for example in metagenomic projects. To avoid distortions in bacteria population studies and false identification of novel taxa, an accurate detection of such chimeric sequences is very important. The accuracy of jpHMM in predicting recombinants of HIV-1 and HBV subtypes justifies the attempt of applying jpHMM to this important issue.

In this thesis, an extension of jpHMM to predict recombinations in viruses with circular genomes such as HBV was developed. To our knowledge, this is the first approach for recombination detection that explicitly takes into account the circularity of genomes. It allows a recombination prediction that takes into account dependencies between nucleotides at the 5' and 3' end of the sequence and is not biased against recombination breakpoints in these regions. Additionally, the output of the predicted recombination in a circular form makes it a recombination detection tool that HBV researchers will highly benefit from. With a runtime of only about 45 seconds, it is also applicable in large-scale studies.

The circular version of jpHMM was applied to recombination detection in HBV assuming that the recombination in HBV takes place during the replication cycle when the genome is present in a circular form. In the case that the recombination takes place while the genome is present in a linear form - the mechanism of recombination has not been completely understood yet - this might not be an appropriate approach. It might happen that, with our approach, small recombinant segments at one end of the linear genome are not detected due to the fact that in the circular version two breakpoints are required for the detection of such a segment instead of one. To get more insights into this problem, the original linear jpHMM should be applied to all available HBV sequences and the results of the recombination prediction should be compared to the ones of the new circular version achieved during this thesis. Nevertheless, the circular jpHMM offers an alternative to the currently used linear recombination detection tools.

The location of recombination breakpoints in the genome can possibly give some indication when the recombination during the replication cycle takes place. If the recombination takes place when the genome is present in a linear form an odd number of breakpoints always leads to different genotypes at both ends of the linear sequence. In this case, after the synthesis of the partially double-stranded circular genome, an "artificial" breakpoint can be observed at the position corresponding to the origin of the linear sequence. Naturally, in the case of an even number of breakpoints, such an artificial breakpoint cannot be observed. A recombination event that takes place during the circular state of the genome always leads to two recombination breakpoints. Thus, a hotspot of recombination breakpoints in the genomic region that corresponds to the origin of the linear sequence can be a hint to the fact that recombination takes place when the genome is present in a linear form. Since the linear version of jpHMM is biased against recombination breakpoints close to the sequence ends, which would lead to an overrepresentation of artificial breakpoints at the origin of the linear form of the genome, the circular jpHMM version is necessary for this evaluation.

To avoid an overrepresentation of breakpoints in similar recombinant forms that all stem from the same recombination event, recombinants should be grouped to identify independent recombination events. For each of these recombination events, only one representative sequence should be taken into account for the determination of recombination hotspots. The evaluation of recombination hotspots is subject of a future project. To start with, in this thesis, a classification system to identify circulating recombinant forms that are very likely to have arisen from independent recombination events is proposed. In contrast to HIV, such a classification system including the definition of CRFs does not exist for HBV. Usually, recombinant forms are defined as new subtypes of genotypes. Such a mixing of genotypes and recombinants can confound recombination detection tools such as jpHMM so that they are unable to distinguish, for example, two genotypes if a recombinant of both genotypes is included as a subgenotype in one of them. Since an accurate classification of HBV genotypes is indispensable to understand the viral evolution, a classification system for HBV sequences that clearly distinguishes pure genotypes from recombinant forms is necessary.

In our study, 17 CRFs and 28 unique recombinant forms (URFs), i.e. recombinant forms that were observed only once or only in related samples, are defined. Some of these CRFs or URFs share the same recombination pattern and similar locations of breakpoints, but are defined as independent recombinant forms. Further analysis of biological experts will be necessary to find out whether these breakpoints really define distinct recombination events. But, nevertheless, the criteria for the definition of CRFs proposed in this thesis will hopefully help to establish a classification system for recombinant forms of HBV.

List of Publications

- T. Unterthiner, **A.-K. Schultz**, J. Bulla, B. Morgenstern, M. Stanke, I. Bulla (2011). Detection of viral sequence fragments of HIV-1 subfamilies yet unknown. *BMC Bioinformatics*, accepted.
- I. Bulla, **A.-K. Schultz**, F. Schreiber, M. Zhang, T. Leitner, B. Korber, B. Morgenstern, M. Stanke (2010). HIV classification using the coalescent theory. *Bioinformatics* 26(11):1409-1415.
- M. Zhang, B. Foley, **A.-K. Schultz**, J. Macke, I. Bulla, M. Stanke, B. Morgenstern, B. Korber, T. Leitner (2010). The role of recombination in the emergence of a complex and dynamic HIV epidemic. *Retrovirology* **7**:25.
- **A.-K. Schultz**, M. Zhang, I. Bulla, T. Leitner, B. Korber, B. Morgenstern, M. Stanke (2009). jpHMM: Improving the reliability of recombination prediction in HIV-1. *Nucleic Acids Research* **37** (suppl_2), W647-W651.
- **A.-K. Schultz**, M. Zhang, T. Leitner, B. Korber, B. Morgenstern, M. Stanke (2008). jpHMM: Erkennung von Rekombinationen bei HIV. *BIOforum* 2/2008, 44-46.
- M. Zhang, **A.-K. Schultz**, C. Calef, C. Kuiken, T. Leitner, B. Korber, B. Morgenstern, M. Stanke (2006). jpHMM at GOBICS: a web server to detect genomic recombinations in HIV-1. *Nucleic Acids Research* **34** (suppl_2), W463 - W465.
- **A.-K. Schultz**, M. Zhang, T. Leitner, C. Kuiken, B. Korber, B. Morgenstern, M. Stanke (2006). A Jumping Profile Hidden Markov Model and Applications to Recombination Sites in HIV and HCV Genomes. *BMC Bioinformatics* **7:**265
- M. Zhang, **A.-K. Schultz**, B. Morgenstern, M. Stanke, B. Korber, T. Leitner (2005). Greater HIV Genome Diversities Inferred From Re-subtyping of HIV Database Sequences. *Proc. German Conference on Bioinformatics* (Discovery Notes, Poster Abstracts), pp 5 - 7.

Bibliography

[1] M. Abdou Chekaraou, S. Brichler, W. Mansour, F. Le Gal, A. Garba, P. Deny, and E. Gordien. A novel hepatitis B virus (HBV) subgenotype D (D8) strain, resulting from recombination between genotypes D and E, is circulating in Niger along with HBV/E strains. *J Gen Virol*, 91(6):1609–1620, 2010.

[2] A. B. Abecasis, P. Lemey, N. Vidal, T. de Oliveira, M. Peeters, R. Camacho, B. Shapiro, A. Rambaut, and A.-M. Vandamme. Recombination Confounds the Early Evolutionary History of Human Immunodeficiency Virus Type 1: Subtype G Is a Circulating Recombinant Form. *J. Virol.*, 81(16):8543–8551, 2007.

[3] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*, chapter 5 "DNA Replication, Repair, and Recombination". Garland Science, Taylor & Francis Group, 5th edition, 2007.

[4] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic Local Alignment Search Tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[5] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[6] P. Arauz-Ruiz, H. Norder, B. H. Robertson, and L. O. Magnius. Genotype H: a new Amerindian genotype of hepatitis B virus revealed in Central America. *J Gen Virol*, 83(8):2059–2073, 2002.

[7] J. Beck and M. Nassal. Hepatitis B virus replication. *World Journal of Gastroenterology*, 13(1):48 – 64, 2007.

[8] H. S. Bilofsy, C. Burks, J. W. Fickett, W. Goad, F. I. Lewitter, W. P. Rindone, C. D. Swindell, and C. S. Tung. The GenBank genetic sequence databank. *Nucleic Acids Res.*, 14(1):1–4, 1986.

[9] S. M. Bowyer and J. G. M. Sim. Relationships within and between genotypes of hepatitis B virus at points across the genome: footprints of recombination in certain isolates. *J Gen Virol*, 81(2):379–392, 2000.

[10] G. Brindicci, G. Punzi, A. Lagioia, S. Lo Caputo, N. Ladisa, G. Di Nicuolo, A. Saracino, G. Angarano, and L. Monno. Difficulties in Classifying A/G Recombinants: Methodological Problems or Genetic Variability? *AIDS Research and Human Retroviruses*, 23(6):840–846, 2007.

[11] M. Carobene, C. Rodrigues, C. De Candia, G. Turk, and H. Salomon. In vitro dynamics of HIV-1 BF intersubtype recombinants genomic regions involved in the regulation of gene expression. *Virology Journal*, 6(1):107, 2009.

[12] L. Cavinta, G. Cao, and S. Schaefer. Description of a New Hepatitis B Virus C6 Subgenotype Found in the Papua Province of Indonesia and Suggested Renaming of a Tentative C6 Subgenotype Found in the Philippines as Subgenotype C7. *J. Clin. Microbiol.*, 47(9):3068–3069, 2009.

[13] CDC. Pneumocystis pneumonia – Los Angeles. *MMWR Morb Mortal Wkly Rep*, 30(21):250–2, 1981.

[14] D. Charif and J. Lobry. SeqinR 1.0-2: a contributed package to the R project for statistical computing devoted to biological sequences retrieval and analysis. In H. R. U. Bastolla, M. Porto and M. Vendruscolo, editors, *Structural approaches to sequence evolution: Molecules, networks, populations*, Biological and Medical Physics, Biomedical Engineering, pages 207–232. Springer Verlag, New York, 2007. ISBN : 978-3-540-35305-8.

[15] B.-F. Chen, P.-J. Chen, G.-M. Jow, E. Sablon, C.-J. Liu, D.-S. Chen, and J.-H. Kao. High prevalence of mixed genotype infections in hepatitis B virus infected intravenous drug users. *Journal of Medical Virology*, pages 536 – 542, 2004.

[16] M. Cornberg, U. Protzer, M. M. Dollinger, J. Petersen, H. Wedemeyer, T. Berg, W. Jilg, A. Erhardt, S. Wirth, P. Schirmacher, W. E. Fleig, and M. P. Manns. The German guideline for the management of hepatitis B virus infection: short version. *Journal of Viral Hepatitis*, 15:1–21, 2008.

[17] C. Cui, J. Shi, L. Hui, H. Xi, Zhuoma, Quni, Tsedan, and G. Hu. The dominant hepatitis B virus genotype identified in Tibet is a C/D hybrid. *J Gen Virol*, 83(11):2773–2777, 2002.

[18] C. De Candia, C. Espada, G. Duette, Y. Ghiglione, G. Turk, H. Salomon, and M. Carobene. Viral replication is enhanced by an HIV-1 intersubtype recombination-derived Vpu protein. *Virology Journal*, 7(1):259, 2010.

[19] U. C. de Silva, J. Warachit, N. Sattagowit, C. Jirapongwattana, S. Panthong, P. Utachee, T. Yasunaga, K. Ikuta, M. Kameoka, and N. Boonsathorn. Genotypic Characterization of HIV Type 1 env gp160 Sequences from Three Regions in Thailand. *AIDS Research and Human Retroviruses*, 26(2):223–227, 2010.

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

[21] S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9):755–763, 1998.

[22] R. C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucl. Acids Res.*, 32(5):1792–1797, 2004.

[23] F. Fang. *Bayesian recombination detection modeling and application*. PhD thesis, Iowa State University, Iowa - United States, 2006.

[24] G. Fang, B. Weiser, C. Kuiken, S. M. Philpott, S. Rowland-Jones, F. Plummer, J. Kimani, B. Shi, R. Kaul, J. Bwayo, O. Anzala, and H. Burger. Recombination following superinfection by HIV-1. *AIDS*, 18:153–159, 2004.

[25] G. E. FOX, K. R. PECHMAN, and C. R. WOESE. Comparative Cataloging of 16S Ribosomal Ribonucleic Acid: Molecular Approach to Procaryotic Systematics. *Int J Syst Bacteriol*, 27(1):44–57, 1977.

[26] W. Gentzsch. Sun Grid Engine: towards creating a compute power grid. In *Proceedings of the first IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 35–36, 2001.

[27] S. T. Goldstein, F. Zhou, S. C. Hadler, B. P. Bell, E. E. Mast, and H. S. Margolis. A mathematical model to estimate global hepatitis B disease burden and vaccination impact. *International Journal of Epidemiology*, 34(6):1329–1339, December 2005.

[28] B. J. Haas, D. Gevers, A. M. Earl, M. Feldgarden, D. V. Ward, G. Giannoukos, D. Ciulla, D. Tabbaa, S. K. Highlander, E. Sodergren, B. MethÃ©, T. Z. DeSantis, T. H. M. Consortium, J. F. Petrosino, R. Knight, and B. W. Birren. Chimeric 16S

rRNA sequence formation and detection in Sanger and 454-pyrosequenced PCR amplicons. *Genome Research*, 21(3):494–504, 2011.

[29] C. Hannoun, H. Norder, and M. Lindh. An aberrant genotype revealed in recombinant hepatitis B virus strains from Vietnam. *J Gen Virol*, 81(9):2267–2272, 2000.

[30] J. Hemelaar, E. Gouws, P. D. Ghys, and S. Osmanov. Global and regional distribution of HIV-1 genetic subtypes and recombinants in 2004. *AIDS*, 20, 2006.

[31] HMMER web page. http://hmmer.org/.

[32] M. Hoelscher, W. E. Dowling, E. Sanders-Buell, J. K. Carr, M. E. Harris, A. Thomschke, M. L. Robb, D. L. Birx, and F. E. McCutchan. Detection of HIV-1 subtypes, recombinants, and dual infections in East Africa by a multi-region hybridization assay. *AIDS*, 16(15):2055–2064, 2002.

[33] Y. Huang, B. Niu, Y. Gao, L. Fu, and W. Li. CD-HIT Suite: a web server for clustering and comparing biological sequences. *Bioinformatics*, 26(5):680–682, 2010.

[34] T. T. T. Huy, T. T. Ngoc, and K. Abe. New Complex Recombinant Genotype of Hepatitis B Virus Identified in Vietnam. *J. Virol.*, 82(11):5657–5663, 2008.

[35] G. Jacobs, A. Loxton, A. Laten, B. Robson, E. J. van Rensburg, and S. Engelbrecht. Emergence and diversity of different HIV-1 subtypes in South Africa, 2000-2001. *Journal of Medical Virology*, 81:1852–1859, 2009.

[36] A. Kay and F. Zoulim. Hepatitis B virus genetic variability and evolution. *Virus Research*, 127(2):164 – 176, 2007. Replicative and Evolutionary Aspects of Hepatitis Viruses.

[37] W. J. Kent. BLAT–The BLAST-Like Alignment Tool. *Genome Research*, 12(4):656–664, 2002.

[38] G. Kijak and F. McCutchan. HIV diversity, molecular epidemiology, and the role of recombination. *Current Infectious Disease Reports*, 7:480–488, 2005. 10.1007/s11908-005-0051-8.

[39] E. D. Kopczynski, M. M. Bateson, and D. M. Ward. Recognition of chimeric small-subunit ribosomal DNAs composed of genes from uncultivated microorganisms. *Appl. Environ. Microbiol.*, 60(2):746–748, 1994.

[40] B. Korber, B. Foley, C. Kuiken, S. Pillai, and J. Sodroski. Numbering Positions in HIV Relative to HXB2CG. In *Human Retroviruses and AIDS 1998*, pages 102–111. Theoretical Biology and Biophysics Group, Los Alamos National Laboratory, Los Alamos, NM, 1998.

[41] A. Kramvis, K. Arakawa, M. C. Yu, R. Nogueira, D. O. Stram, and M. C. Kew. Relationship of serological subtype, basic core promoter and precore mutations to genotypes/subgenotypes of hepatitis B virus. *Journal of Medical Virology*, 80:27–46, 2008.

[42] A. Kramvis, M. Kew, and G. François. Hepatitis B virus genotypes. *Vaccine*, 23(19):2409 – 2423, 2005.

[43] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov Models in Computational Biology: Applications to Protein Modeling. *J. Mol. Biol.*, 235:1501–1531, 1994.

[44] M. Krzywinski, J. Schein, I. Birol, J. Connors, R. Gascoyne, D. Horsman, S. J. Jones, and M. A. Marra. Circos: An information aesthetic for comparative genomics. *Genome Research*, 19(9):1639–1645, 2009.

[45] C. Kuiken, B. Foley, P. Marx, S. Wolinsky, T. Leitner, B. Hahn, F. McCutchan, and B. Korber. In *HIV Sequence Compendium 2008*. Theoretical Biology and Biophysics Group, Los Alamos National Laboratory, Los Alamos, NM, 2008.

[46] C. Kuiken and P. Simmonds. Nomenclature and numbering of the hepatitis c virus. In H. Tang, editor, *Hepatitis C*, volume 510 of *Methods in Molecular Biology*, pages 33–53. Humana Press, 2009. 10.1007/978-1-59745-394-3_4.

[47] F. Kurbanov, Y. Tanaka, A. Kramvis, P. Simmonds, and M. Mizokami. When Should "I" Consider a New Hepatitis B Virus Genotype? *J. Virol.*, 82(16):8241–8242, 2008.

[48] F. Kurbanov, Y. Tanaka, and M. Mizokami. Geographical and genetic diversity of the human hepatitis B virus. *Hepatology Research*, pages 14–30, 2010.

[49] LANL web page. http://www.hiv.lanl.gov.

[50] W. Li and A. Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.

[51] W. Liesack, H. Weyland, and E. Stackebrandt. Potential risks of gene amplification by pcr as determined by 16s rdna analysis of a mixed-culture of strict barophilic bacteria. *Microbial Ecology*, 21:191–198, 1991. 10.1007/BF02539153.

[52] C.-J. Liu and J.-H. Kao. Hepatitis b virus genotype: What should the clinician know? *Current Hepatitis Reports*, 6:17–23, 2007. 10.1007/BF02942174.

[53] K. S. Lole, R. C. Bollinger, R. S. Paranjape, D. Gadkari, S. S. Kulkarni, N. G. Novak, R. Ingersoll, H. W. Sheppard, and S. C. Ray. Full-length human immunodeficiency virus type 1 genomes from subtype C-infected seroconverters in India, with evidence of intersubtype recombination. *J. Virology*, 73:152–160, 1999.

[54] B. Lowerre. *The Harpy Speech Recognition System*. PhD thesis, Carnegie-Mellon University, 1976.

[55] M. I. Lusida, V. E. Nugrahaputra, Soetjipto, R. Handajani, M. Nagano-Fujii, M. Sasayama, T. Utsumi, and H. Hotta. Novel Subgenotypes of Hepatitis B Virus Genotypes C and D in Papua, Indonesia. *J. Clin. Microbiol.*, 46(7):2160–2166, 2008.

[56] D. P. Martin, P. Lemey, M. Lott, V. Moulton, D. Posada, and P. Lefeuvre. RDP3: a flexible and fast computer program for analyzing recombination. *Bioinformatics*, 26(19):2462–2463, 2010.

[57] J. Maydt and T. Lengauer. Recco: recombination analysis using cost optimization. *Bioinformatics*, 22(9):1064–1071, February 2006.

[58] F. E. McCutchan. Global epidemiology of HIV. *Journal of Medical Virology*, 2006.

[59] B. McMahon. The influence of hepatitis b virus genotype and subgenotype on the natural history of chronic hepatitis b. *Hepatology International*, 3:334–342, 2009. 10.1007/s12072-008-9112-z.

[60] B. H. M. Meldal, N. M. Moula, I. H. A. Barnes, K. Boukef, and J.-P. Allain. A novel hepatitis B virus subgenotype, D7, in Tunisian blood donors. *J Gen Virol*, 90(7):1622–1628, 2009.

[61] V. Morozov, M. Pisareva, and M. Groudinin. Homologous recombination between different genotypes of hepatitis B virus. *Gene*, 260(1-2):55 – 65, 2000.

[62] Mulyanto, S. Depamede, K. Surayah, F. Tsuda, K. Ichiyama, M. Takahashi, and H. Okamoto. A nationwide molecular epidemiological study on hepatitis b virus in

indonesia: identification of two novel subgenotypes, b8 and c7. *Archives of Virology*, 154:1047–1059, 2009.

[63] H. Naumann, S. Schaefer, C. F. T. Yoshida, A. M. C. Gaspar, R. Repp, and W. H. Gerlich. Identification of a new hepatitis B virus (HBV) genotype from Brazil that expresses HBV surface antigen subtype adw4. *J Gen Virol*, 74(8):1627–1632, 1993.

[64] M. Negroni and H. Buc. MECHANISMS OF RETROVIRAL RECOMBINATION. *Annual Review of Genetics*, 35(1):275–302, 2001.

[65] H. Njai, Y. Gali, G. Vanham, C. Clybergh, W. Jennes, N. Vidal, C. Butel, E. Mpoudi-Ngolle, M. Peeters, and K. Arien. The predominance of Human Immunodeficiency Virus type 1 (HIV-1) circulating recombinant form 02 (CRF02_AG) in West Central Africa may be related to its replicative fitness. *Retrovirology*, 3(1):40, 2006.

[66] H. Norder, A.-M. Couroucé, and L. O. Magnius. Complete Genomes, Phylogenetic Relatedness, and Structural Proteins of Six Strains of the Hepatitis B Virus, Four of Which Represent Two New Genotypes. *Virology*, 198(2):489 – 503, 1994.

[67] H. Norder, B. Hammas, S. Löfdahl, A.-M. Couroucé, and L. O. Magnius. Comparison of the amino acid sequences of nine different serotypes of hepatitis B surface antigen and genomic classification of the corresponding hepatitis B virus strains. *J Gen Virol*, 73(5):1201–1208, 1992.

[68] H. Okamoto, F. Tsuda, H. Sakugawa, R. I. Sastrosoewignjo, M. Imai, Y. Miyakawa, and M. Mayumi. Typing Hepatitis B Virus by Homology in Nucleotide Sequence: Comparison of Surface Antigen Subtypes. *J Gen Virol*, 69(10):2575–2583, 1988.

[69] C. M. Olinger, P. Jutavijittum, J. M. HÃ¼bschen, A. Yousukh, B. Samountry, T. Thammavong, K. Toriyama, and C. P. Muller. Possible new hepatitis B virus genotype, Southeast Asia. *Emerging Infectious Diseases*, 14(11):1777–1780, 2008.

[70] F. J. Palella, K. M. Delaney, A. C. Moorman, M. O. Loveless, J. Fuhrer, G. A. Satten, D. J. Aschman, and S. D. Holmberg. Declining morbidity and mortality among patients with advanced human immunodeficiency virus infection. *New England Journal of Medicine*, 338(13):853–860, 1998.

[71] N. Panjaworayan, S. Roessner, A. Firth, and C. Brown. HBVRegDB: Annotation, comparison, detection and visualization of regulatory elements in hepatitis B virus sequences. *Virology Journal*, 4(1):136, 2007.

[72] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 85(8):2444–2448, 1988.

[73] J.-C. Plantier, M. Leoz, J. E. Dickerson, F. De Oliveira, F. Cordonnier, V. Lemee, F. Damond, D. L. Robertson, and F. Simon. A new human immunodeficiency virus derived from gorillas. *Nature Medicine*, 15:871–872, 2009.

[74] T. Plötz and G. A. Fink. Accelerating the Evaluation of Profile HMMs by Pruning Techniques. Technical report, Bielefeld University, Faculty of Technology, 2004. Report 2004-03.

[75] M. A. Purdy, A. C. Gonzales, Z. Dimitrova, and Y. Khudyakov. Supragenotypic groups of the hepatitis B virus genome. *J Gen Virol*, 89(5):1179–1183, 2008.

[76] M. E. Quiñones-Mateu and E. J. Arts. HIV-1 Fitness: Implications for Drug Resistance, Disease Progression, and Global Epidemic Evolution. In *HIV Sequence Compendium 2001*, pages 134–170. Theoretical Biology and Biophysics Group, Los Alamos National Laboratory, 2001.

[77] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2006. ISBN 3-900051-07-0.

[78] Recombination analysis software. http://www.bioinf.manchester.ac.uk/recombination/programs.shtml.

[79] J. D. Reeves and R. W. Doms. Human immunodeficiency virus type 2. *J Gen Virol*, 83(6):1253–1265, 2002.

[80] D. L. Robertson, J. P. Anderson, J. A. Bradac, J. K. Carr, B. Foley, R. K. Funkhouser, F. Gao, B. H. Hahn, M. L. Kalish, C. Kuiken, G. H. Learn, T. Leitner, F. McCutchan, S. Osmanov, M. Peeters, D. Pieniazek, M. Salminen, P. M. Sharp, S. Wolinsky, and B. Korber. HIV-1 Nomenclature Proposal. *Science*, 288(5463):55d–, 2000.

[81] S. Sanabani, E. de Souza Pastena, W. Neto, V. Martinez, and E. Sabino. Characterization and frequency of a newly identified HIV-1 BF1 intersubtype circulating recombinant form in Sao Paulo, Brazil. *Virology Journal*, 7(1):74, 2010.

[82] S. Sanabani, E. Pastena, W. Neto, C. Barreto, K. Ferrari, E. Kalmar, S. Ferreira, and E. Sabino. Near full-length genome analysis of low prevalent human immunodeficiency virus type 1 subclade F1 in Sao Paulo, Brazil. *Virology Journal*, 6(1):78, 2009.

[83] S. Schaefer, L. Magnius, and H. Norder. Under Construction: Classification of Hepatitis B Virus Genotypes and Subgenotypes. *Intervirology*, pages 323–325, 2009.

[84] A.-K. Schultz. Ein springendes Profil-Hidden-Markov-Modell und Anwendungen auf Rekombinationen in viralen Genomen. Diploma thesis, Georg-August-Universität Göttingen, Germany, 2006.

[85] A.-K. Schultz, M. Zhang, I. Bulla, T. Leitner, B. Korber, B. Morgenstern, and M. Stanke. jpHMM: Improving the reliability of recombination prediction in HIV-1. *Nucleic Acids Research*, 37(suppl 2):W647–W651, 2009.

[86] A.-K. Schultz, M. Zhang, I. Bulla, T. Leitner, B. Korber, B. Morgenstern, and M. Stanke. jpHMM: improving the reliability of recombination prediction in HIV-1. *Nucleic Acids Research*, 38(3):1059, 2010.

[87] A.-K. Schultz, M. Zhang, T. Leitner, C. Kuiken, B. Korber, B. Morgenstern, and M. Stanke. A jumping profile Hidden Markov Model and applications to recombination sites in HIV and HCV genomes. *BMC Bioinformatics*, 7(1):265, 2006.

[88] C. Seeger and W. S. Mason. Hepatitis B Virus Biology. *Microbiol. Mol. Biol. Rev.*, 64(1):51–68, 2000.

[89] P. M. Sharp, G. M. Shaw, and B. H. Hahn. Simian Immunodeficiency Virus Infection of Chimpanzees. *J. Virology*, 79(7):3891–3902, 2005.

[90] P. Simmonds and S. Midgley. Recombination in the Genesis and Evolution of Hepatitis B Virus Genotypes. *J. Virol.*, 79(24):15467–15476, 2005.

[91] K. Sjölander, K. Karplus, M. Brown, R. Hughey, A. Krogh, I. S. Mian, and D. Haussler. Dirichlet Mixtures: A Method for Improved Detection of Weak but Significant Protein Sequence Homology. *Comput. Applic. Biosci.*, 12:327–345, 1996.

[92] R. Spang, M. Rehmsmeier, and J. Stoye. Sequence Database Search Using Jumping Alignments. In *Proceedings of ISMB 2000*, pages 367–375, 2000.

[93] R. Spang, M. Rehmsmeier, and J. Stoye. A Novel Approach to Remote Homology Detection: Jumping Alignments. *J. Comp. Biol.*, 9(5):747–760, 2002.

[94] B. Stroustrup. *The C++ programming language*. Addison-Wesley Series in Computer Science, 1991.

[95] L. Stuyver, S. De Gendt, C. Van Geyt, F. Zoulim, M. Fried, R. F. Schinazi, and R. Rossau. A new genotype of hepatitis B virus: complete genome and phylogenetic relatedness. *J Gen Virol*, 81(1):67–74, 2000.

[96] F. Sugauchi, E. Orito, T. Ichida, H. Kato, H. Sakugawa, S. Kakumu, T. Ishida, A. Chutaputti, C.-L. Lai, R. Ueda, Y. Miyakawa, and M. Mizokami. Hepatitis B Virus of Genotype B with or without Recombination with Genotype C over the Precore Region plus the Core Gene. *J. Virol.*, 76(12):5985–5992, 2002.

[97] J. Summers and W. S. Mason. Replication of the genome of a hepatitis B-like virus by reverse transcription of an RNA intermediate. *Cell*, 29(2):403 – 415, 1982.

[98] K. Tatematsu, Y. Tanaka, F. Kurbanov, F. Sugauchi, S. Mano, T. Maeshiro, T. Nakayoshi, M. Wakuta, Y. Miyakawa, and M. Mizokami. A Genetic Variant of Hepatitis B Virus Divergent from Known Human and Ape Genotypes Isolated from a Japanese Patient and Provisionally Assigned to New Genotype J. *J. Virol.*, 83(20):10538–10547, 2009.

[99] D. M. Tebit, I. Nankya, E. J. Arts, and Y. Gao. HIV Diversity, Recombination and Disease Progression: How Does Fitness F̈itÏnto the Puzzle? *AIDS Reviews*, 9(2):75–87, 2007.

[100] UNAIDS/WHO. AIDS Epidemic Update, 2009.

[101] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory*, IT-13:260–269, 1967.

[102] WHO. Hepatitis B Vaccines. *Weekly Epidemiological Record*, 84(40):405–420, 2009.

[103] E. Wilkinson and S. Engelbrecht. Molecular characterization of non-subtype C and recombinant HIV-1 viruses from Cape Town, South Africa. *Infection, Genetics and Evolution*, 9(5):840 – 846, 2009.

[104] M. Wistrand and E. L. Sonnhammer. Transition Priors for Protein Hidden Markov Models: An Empirical Study towards Maximum Discrimination. *J. Comp. Biol.*, 11(1):181–193, 2004.

[105] P. C. Y. Woo, S. K. P. Lau, J. L. L. Teng, H. Tse, and K.-Y. Yuen. Then and now: use of 16S rDNA gene sequencing for bacterial identification and discovery of novel

bacteria in clinical microbiology laboratories. *Clinical Microbiology and Infection*, pages 908–934, 2008.

[106] L. Ye, Y. Zhang, Y. Mei, P. Nan, and Y. Zhong. Detecting putative recombination events of hepatitis B virus: An updated comparative genome analysis. *Chinese Science Bulletin*, 55:2373–2379, 2010. 10.1007/s11434-010-3109-4.

[107] H. Yu, Q. Yuan, S.-X. Ge, H.-Y. Wang, Y.-L. Zhang, Q.-R. Chen, J. Zhang, P.-J. Chen, and N.-S. Xia. Molecular and Phylogenetic Analyses Suggest an Additional Hepatitis B Virus Genotype "I". *PLoS ONE*, 5(2):e9297, 02 2010.

[108] M. Zhang, B. Foley, A.-K. Schultz, J. Macke, I. Bulla, M. Stanke, B. Morgenstern, B. Korber, and T. Leitner. The role of recombination in the emergence of a complex and dynamic HIV epidemic. *Retrovirology*, 7(1):25, 2010.

[109] M. Zhang, A.-K. Schultz, C. Calef, C. Kuiken, T. Leitner, B. Korber, B. Morgenstern, and M. Stanke. jpHMM at GOBICS: a web server to detect genomic recombinations in HIV-1. *Nucleic Acids Research*, 34(suppl 2):W463–W465, 2006.

Lebenslauf

Name	Anne-Kathrin Schultz
Geburtsdatum	18. Oktober 1979
Geburtsort	Göttingen
Staatsangehörigkeit	deutsch

Schulbildung

1986-1990	Grundschule Bubenreuth
1990-1995	Ohm-Gymnasium Erlangen
1995-1998	Romain-Rolland-Gymnasium Dresden Abschluss: Abitur

Studium

10/1998 - 02/2006	Studium der Mathematik (Nebenfach Genetik) an der Georg-August-Universität Göttingen Abschluss: Diplom
09/2001 - 07/2002	Auslandsstudium (ERASMUS) an der Université Victor Segalen Bordeaux 2, Bordeaux, Frankreich, Studiengang: Mathématiques Appliquées et Sciences Sociales
seit 04/2006	Promotionsstudium im Studienfach Mathematik an der Georg-August-Universität Göttingen
11/2006 - 12/2006, 11/2007	Besuch der David R. Cheriton School of Computer Science an der University of Waterloo, Ontario, Kanada ("PPP Kanada", DAAD)

Berufstätigkeit

03/1999	Studentische Hilfskraft am Leibniz-Institut für Festkörper- und Werkstoffforschung Dresden
08/2002 - 09/2002	Studentische Hilfskraft am Max-Planck-Institut für Mathematik in den Naturwissenschaften in Leipzig
SS03 - SS05	Studentische Hilfskraft am Institut für Mikrobiologie und Genetik der Universität Göttingen
03/2006 - 07/2006	Wissenschaftliche Hilfskraft am Institut für Mikrobiologie und Genetik der Universität Göttingen
seit 09/2006	Wissenschaftliche Mitarbeiterin der Abteilung Bioinformatik des Instituts für Mikrobiologie und Genetik der Universität Göttingen