A LIKELIHOOD BASED CLUSTERING METHOD FOR DETECTION OF RECOMBINATION FOR DNA AND AMINO ACID SEQUENCES

by

Li Li

Submitted in partial fulfillment of the requirements
for the degree of Master of Science

at

Dalhousie University
Halifax, Nova Scotia
November 2015

This thesis is dedicated to My Mother

Table of Contents

List of Tables

List of Figures

Abstract

Genetic Recombination is a process where parts of different genes are combined to form a new gene. Recombination detection is an important part of phylogenetic analysis for DNA sequences and thus the detection of recombination events has received great attention in the phylogenetics literature. However most methods are either computationally expensive and are not suitable for analyzing many genes or a genome, or are computationally fast but are not accurate enough. Furthermore, almost all existing packages are developed for DNA sequences, and cannot be easily used to analyze amino acid sequences. We propose a new algorithm which is fast and accurate for recombination detection and can be used on both DNA or amino acid sequences. Our method is a simple clustering algorithm based on the site log-likelihood. Performance of the method is evaluated on both simulated and real data examples.

List of Abbreviations and Symbols Used

Symbols and Abbr.	Description
DNA	Deoxyribonucleic Acid
T	Thymine
C	Cytosine
G	Guanine
A	Adenine
ϕ_w	Pairwise homoplasy index
HIV	Human Immunodeficiency Virus
DEN-1	Dengue type 1 virus
ML	Maximum Likelihood
PLATO	Partial Likelihoods Assessed Through Optimisation
Phylo-HMM	Phylogenetic Hidden Markov Model
MM	Mixture Model
GARD	Genetic Algorithm Recombination Detection
AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion
AICc	Corrected Akaike Information Criterion
NJ	Neighbor Joining
PBC	Phylogenetic Block Clustering
τ	Tree Topology
$\hat{\theta}$	MLE
MLE	Maximum Likelihood Estimator
ω	Weight
PAML	Phylogenetic Analysis by Maximum Likelihood
RF	Robinson-Foulds
SPR	Sub-tree Prune and Regraft operation
GTR	Generalised time reversible
PHYLIP	Phylogeny Inference Package
PBC-O	PBC Oracle

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Prof. Hong Gu and Prof. Toby Kenney, for their excellent support of my research. Without their detailed guidance and thoroughness in the revision of this thesis, this work wouldn't be as presented here.

Chapter 1

Introduction

Genetic recombination is the process under which regions of genetic sequence which come from different DNA molecules are combined into a new DNA sequence, so that different parts of the DNA sequence have different evolutionary histories. Recombination plays a very important role in shaping the genetic patterns and exchanging the genetic information in all living organisms and viruses. Recombination is a strong driver of genetic diversity, and has a large influence on molecular evolution inference. We will be focused on homologous recombination in which recombination occurred among homologous genes.

The phylogenetic tree topology estimation can be complicated by recombination events. Posada and Crandall [24] studied the effect of recombination on phylogeny estimation under a range of simulated scenarios and found that in some cases, the tree describing the majority of cases could be inferred, but that in cases where the recombination event was recent, between divergent taxa, and when the breakpoint is located around the middle of the alignment, the inferred tree is usually different from the phylogenies used to simulate either part of the alignment. If the phylogenetic inference is inaccurate, all the applications that rely on the phylogeny will be influenced. Thus detecting the corresponding recombination events and the corresponding recombination breakpoints are very important steps before a phylogenetic tree is estimated.

A large number of methods have been developed for recombination detection. A list of different recombination analysis software packages are maintained by Robertson Lab (http://bioinf.man.ac.uk/robertson/recombination/programs.shtml). Among these software packages, except the two methods developed in Boussau, Guéguen and Gouy, 2009) [2] which can be used on both DNA sequence data and protein-coding sequences, all other packages are limited to DNA sequence data.

Different methods are based on different rationales, with most of the methods

Table 1.1: An example of aligned DNA sequences for 4 taxa

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
$Taxon_1$	A	A	T	C	G	T	G	T
$Taxon_2$	C	A	T	G	C	A	G	G
$Taxon_3$	C	T	G	C	C	A	T	G
$Taxon_4$	G	C	C	A	T	G	A	C

based on a general principle that recombination only happened if a multiple-sequence alignment can not be described by a single phylogeny. The accuracy of many of these methods have been evaluated through both simulation and empirical data analysis [23, 22]. In general, these methods can be classified into different categories based on the methodologies they involve. We present below a brief review of some of these methods within each of these categories after introducing basic notations for DNA or amino acid sequence data.

1.1 Notations of DNA sequences

Suppose we have a data set of aligned DNA sequences of length n for m taxa which form an $m \times n$ matrix. A DNA sequence consists of 4 different nucleotide characters: thymine (T) and cytosine(C) (Pyrimidine), guanine (G) and adenine (A) (Purines). Columns of the alignment, $X_1, X_2, ..., X_n$, also called sites, serve as observations of the evolutionary process. Table 1.1 is an example of aligned DNA sequences for 4 taxa. Data can also be in the form of amino acid sequences, where each entry in the above matrix represents one of 20 amino acids.

1.2 Methods based on distance patterns

The Phylogenetic-profile method (Weiller, 1998) [32] uses distance vectors to detect recombinant sequences by checking data coherence at every test location along a sequence alignment. The method splits the whole length of sequence into two parts of sequence data, one part contains the sequence data right before the test location, and the other part contains the sequence data right after the test location. These two parts of sequence data are interpreted by two distance vectors which contain pairwise

distances between pairs of sequences. The method compares the distance patterns by examining the linear correlation coefficient between the test sequence and all other sequences in each distance vector to see whether it shows similar patterns. Since this method doesn't involve estimating a tree topology, it is typically a fast method. However,this method can have low accuracy for locating the recombination signals if a large number of recombination events occurred. Posada classified this method as a distance method and noted the method is computationally efficient and could possibly be used in larger data sets (Posada, 2002) [22].

Another method that belongs to this category is the split decomposition analysis (Huson, 1998) [15], which uses the pairwise distance matrix to build the splits graph for detecting the recombinant sequences which contain the inconsistent phylogeny. The splits graph indicates the phylogenetic recombination event by showing the distances between each pair of taxa. This method only infers the presence or absence of a recombination event, it does not predict the recombination breakpoints and the corresponding different phylogenies.

1.3 Methods based on parsimony and parsimoniously informative sites

Dividing the sequence data into binary partitions, the Partition matrix method (Jakobsen, Wilson, and Easteal. 1997) [16] checks the phylogenetic consistency between the informative sites along a set of nucleotide sequences. Rows and columns of the partition matrix are the parsimoniously informative sites and binary partitions respectively. An informative site is one in which at least two nucleotides each occur twice. For example, site AACCT is an informative site because A and C occur twice. A site is consistent with a partition if for each nucleotide, all occurrences of that nucleotide are in the same block of the partition. For example, AACCT is consistent with the split $(1, 2, 5)$, $(3, 4)$ but not with the split $(1, 3, 4)$, $(2, 5)$, since there is one A in each block of this partition. This consistency information is arranged into a matrix whose ij^{th} entry is 1 if site i is consistent with partition j, 0 if site i is inconsistent with partition j. The idea is that from the way information is consistently distributed in a given sequence alignment, we can determine whether recombination has taken place. The method is very fast since less computation is involved.

Another approach to detect recombination in a set of nucleotide sequences is the

homoplasy test method (Smith and Smith, 1998) [29]. One important measure for this method is to find the apparent homoplasies h, where h is the difference between the most parsimonious number of substitutions at all sites under a given tree and the count of polymorphic sites. Given a fixed number of polymorphic sites and a fixed number of sites which have the same rate of evolutionary changes, estimating the value of the probability of having h greater than zero is the key procedure to determine that recombination has occurred. If this value is relatively large, this indicates that recombination is unlikely to have happened. This method is especially useful for alignments of sequences which are very compatible, i.e. which agree on 95%-99% of the sites. Applying the method on three bacterial genes, results show that the method performed very well if the value of mutation rate is high.

Similar to the partition matrix method, the refined incompatibility matrix method (Bruen, Philippe and Bryant, 2006) [3] is based on calculating a new test statistic ϕ_w, which is called the pairwise homoplasy index, and finding the distribution of this statistic. The method permutes the alignment sequences for detecting compatibility to determine the recombination between the parsimoniously informative sites. Two sites are compatible if and only if their evolutionary history can be described by a single phylogeny. Let M be the number of parsimoniously informative sites in an alignment, the pairwise refined incompatibility scores of informative sites are represented by the entries of the off-diagonal elements of an M x M matrix. The method calculates ϕ_w by computing the mean refined incompatibility score, where the refined incompatibility score between two informative sites is the difference between the total number of required substitutions for representing the possible tree topologies and the total number of character states of both sites plus a constant 2. For example, the refined incompatibility score between the site AACC and site AGGA is 1, because there is one tree that can represent the site AACC with a single substitution, but this tree needs 2 substitutions to represent the site AGGA. Therefore, 3 substitutions are needed to represent both sites. Meanwhile the total number of character states of both sites is 4, so 3 - 4 + 2 is equal to 1. This method estimates the recombination rate more accurately than the previous methods. Furthermore, this method can detect recombination correctly regardless of correlation among multiple sequences.

1.4 Permutation-based methods

Since different sites experience different mutation rates in their phylogenetic history, a permutation-based method uses the population recombination rate to detect and estimate recombination in multiple alignment sequences (Hudson, 2001) [12]. Based on the assumption that there is the same mutation rate at each site when estimating the population recombination rate under an evolutionary model, and considering the fact that the order of the sites affects the likelihood of the observed data, the method first estimates the population recombination rate per site by inputting the original data set, then permutes the sites and generates the new population recombination rate per site for each permutation correspondingly. Let P denote the probability that the population recombination rate per site for the permuted data sets is greater than or equal to the rate for the original data set. The method would conclude that recombination has occurred if P is lower than a certain significance level. This method worked well for detecting recombination events in both HIV-1 and HIV-2 sequences. However, when the actual population mutation rate is much smaller than it is supposed to be for estimating the likelihoods, it is hard for the method to detect the recombination (McVean, Awadalla, and Fearnhead. 2002) [20].

1.5 Likelihood based methods

Several methods for detecting recombination have been developed based on the likelihood scores of different regions of molecular sequences. These methods often involve sliding windows of different sizes along the sequence.

Holmes, Worobey, and Rambaut (1999) [11] developed a likelihood-based test to locate the position of recombination events in the alignment sequences. The null hypothesis is that there is only a single phylogeny for the sequences. The alternative hypothesis is that at least two different tree topologies exist within the sequence. For each possible breakpoint, a likelihood ratio test is applied to test if the model fitting on two tree topologies on two sides of the breakpoint is significantly better than fitting one tree topology on the whole sequence. The null distribution for the likelihood ratio test is generated by Monte Carlo simulations. To optimize both the tree topologies and branch parameters is computationally prohibitive, especially for

large trees, thus the split decomposition method [15] is applied to identify the candidate parental sequences prior to the application of this algorithm. This maximum likelihood method had been tested on the DEN-1 viruses and recognized the recombination point by identifying strong contradictory signals from the nucleotide sequence around the breakpoints [11]. Posada classified these methods as phylogenetic-based methods and found that phylogenetic-based methods can detect recombination even when the recombination events were frequent [22]. This method is computationally very expensive.

Grassly and Holmes (1997) [10] described a method (PLATO) based on a statistic Q calculated for each window with different windows of varying sizes and positions along the sequence to identify regions of the sequence with either recombination or varying selective forces. More specifically, Q is defined as the ratio of average log-likelihood for the sites within a window and the average log-likelihood for the rest of the sites. The site log-likelihood is calculated on the global maximum likelihood phylogeny with the substitution model fixed as F84 (transition transversion ratio set as 2). For each window size, the maximum of the Q values indicates that the sequence within that window has the smallest average site log-likelihood under the global maximum phylogeny. Based on a Normal approximation to the null distribution, the test is performed for each window size and the Bonferroni correction is used to control the type-I error for the multiple tests with different window sizes. Through simulation, Grassly and Holmes (1997) [10] demonstrated that this method can detect recombination accurately except in the case where the recombination happened in a very short region. This is reasonable because most methods can't detect recombination correctly as the size of region of recombination becomes smaller. The computation in this method is not heavy, because the site log-likelihood values are all based on one ML tree with no model parameters estimated from the data. From Posada's comparison [22] over the 14 methods, PLATO is not shown to be one of the most powerful methods though.

1.6 Methods based on a mixture model and a hidden Markov model

Many methods reviewed so far are focused on the recombination rates without detecting the breakpoints and reconstructing the phylogenies in the same algorithm.

Boussau, Guéguen and Gouy (2009) [2] developed two methods, based on a mixture model and a hidden Markov model, to simultaneously detect recombination breakpoints and reconstruct phylogenies. In addition, these two methods can handle both nucleotide and protein sequences.

Both these methods in fact belong to the category of likelihood based methods. We review these methods separately because these methods have close connection to the method developed later in this thesis. The mixture model method maximizes the total log-likelihood of the data assuming that each site follows a mixture distribution over $|T|$ trees with the across-site rate heterogeneity incooperated in the substitution model by a gamma distribution. The mixing probability used in the likelihood for each site is $1/|T|$ for each tree, i.e. assuming the same probability for each site generated by each tree. The $|T|$ tree topologies and all the parameters are optimized in the algorithm. The mixture model algorithm in Boussau, Guéguen and Gouy (2009) [2] outputs a site likelihood matrix over the $|T|$ optimized trees by maximizing the mixture log-likelihood of the data and then apply a partitioning algorithm to partition the sites into consecutive k blocks. The application of this method in the simulation study and the real data analysis in Boussau, Guéguen and Gouy (2009) [2] used $|T| = 2$, this contributed greatly to the claimed fast speed of the algorithm. The starting two trees in Boussau, Guéguen and Gouy (2009) [2] were each estimated from half of the input alignments, this naturally gives the advantage to the alignments with true breakpoints in the middle and could be problematic to the cases that the true breakpoints are towards the ends of the alignments. Finally the method to decide the number of partitions is not convincing.

The basic idea of the mixture model approach is to estimate the different underlying topologies more accurately by assuming a mixture probability on each site. If two underlying topologies are very different and each site has clearly very different likelihood over the two different topologies, then each site will mainly contribute to the estimation of its underlying true tree. Since there are many uninformative sites that can come from each tree, these sites will contribute to the estimation of different topologies with different strength. These sites likely will lead to the wrong estimation for the Gamma parameters which is used to model the rate variation across sites, which in turn could lead to the wrong tree estimation. It is not clear how this will

influence the site likelihood matrix on which the partition method is based. From the simulation study, the performance of this method is inferior to that of the method based on a hidden Markov model in that the estimated breakpoints are less accurate. When we apply this method on our simulated data, the computation time is much longer than that indicated in Boussau, Guéguen and Gouy (2009) [2]. It is very time-consuming if we set $|T|$ larger than 2.

The method based on a hidden Markov model (Phylo-HMM) can be viewed as a mixture model with mixing elements as models of partitions. This model is better than the mixture model approach above in that the block partition structure is taken care of in the modelling. Again Boussau, Guéguen and Gouy (2009) [2] used $|T| = 2$, with the initial two trees each estimated from half of the input alignments. The results of this method seem to heavily rely on the accuracy of the input starting trees. For the simulations with true breakpoint in the middle of the sequence, the initial tree estimates are close to the two true underlying topologies, the Phylo-HMM can accurately detect the breakpoints. In cases where the recombinant section is short relative to the whole alignment, when Phylo-HMM inputs two estimated topologies by each half of the sequence which might be two similar initial topologies both representing the majority of the sequence, it creates difficulty in optimizing the two trees. Indeed, Boussau, Guéguen and Gouy (2009) [2] found that the Phylo-HMM method performs poorly when the breakpoint position is near either end of the alignment, and that the phylogenies of both trees are poorly reconstructed in this case. It should be possible to overcome these issues by considering a larger number of starting trees. However, when we apply Phylo-HMM method with a larger number of starting tress on the simulated data using the same simulation scenarios in [2] we found that not only the computational cost is dramatically increased but also the recombination detection accuracy is greatly reduced. The number of breakpoints are highly associated with the number of input trees $|T|$. This makes the method invalid when our purpose is to find if there is any breakpoint and how many there are.

1.7 GARD method

The Genetic Algorithm Recombination Detection (GARD) method is another recently developed recombination detection method by Pond et al. [21]. Two procedures

are described in [21]. The first procedure is for screening for a single breakpoint in an alignment by thoroughly searching each variable site, i.e. by calculating the corrected Akaike Information Criterion (AICc) [4] for all possible partitions of two continuous blocks where a Neighbor-Joining (NJ) tree is estimated on each block of the alignment and the branch parameters are estimated by maximizing the likelihood over the estimated NJ trees. The other model parameters are estimated on a NJ tree using the whole sequence and are fixed in the subsequent recombination detection procedures. If any partition generates a AICc score less than the whole sequence AICc score without breakingpoint, then they deduce that there is a recombination event.

The procedure for searching for multiple breakpoints uses in principle the same idea as the procedure for searching for a single breakpoint. Since it is infeasible to exhaustively calculate the AICc for all possible number of breakpoints on all possible locations, a genetic algorithm is implemented to stochastically search for this space. It was claimed that GARD has more power and smaller false positive rate than all other existing recombination detection methods. The idea of GARD is simple, it tries to find the model with smallest AICc by thoroughly or stochastically searching the whole model space. From the simulation results in [21], it seems that the false positive rate is still very high. This may partially be related to the model criterion AICc and partially related to the stochastic searching. Needless to say that GARD is a method that relies heavily on computing resources.

In the next chapter, we develop a much simpler likelihood based clustering procedure to separate the sequence alignments into consecutive blocks so that each block corresponds to a different evolutionary history. The method is based on site log-likelihoods on a number of trees and can be applied to amino acid sequences as well as nucleotide sequences without any additional work. The speed of the method makes it possible to detect all recombination events in a genome as well as in a gene. We call our method the phylogenetic block clustering (PBC) method.

Chapter 2

Phylogenetic block clustering (PBC) method

With larger number of genomes sequenced, the scale of inference on these genomic data continuously becomes larger. The detection of recombination, as one of the most important steps in the phylogenetic inference, is also facing the difficulties of this larger scale. The computationally intensive methods such as that by Holmes, Worobey, and Rambaut (1999) [11] and GARD [21] can potentially suffer from the scale of the problem. Currently GARD can only detect recombinations for DNA sequences. Although its principles should work the same for amino acid sequences, due to the computational burden or other issues, it's not available for analyzing amino acid sequences. The methods based on a mixture model and a hidden Markov model [2] could be performed on amino acid sequences, but both their accuracy and speed in detecting multiple breakpoints are problematic.

We propose in this thesis a much simpler method which is accurate, fast and is scalable. There are two important elements in our method. The first is that we start from the site log-likelihood over a number of tree topologies with all the branch parameters and model parameters estimated from the whole gene alignment. The second element is a block clustering algorithm along the sequence. Given any DNA or protein sequence data, we first calculate a site log-likelihood matrix based on a set of plausible trees. We then cluster the sites of the sequences into blocks of consecutive sites, so that the maximum likelihood scores on each block of sites are achieved on different topologies. Finally, we select the number of breakpoints according to commonly used statistical criteria such as AIC (Akaike information criterion) and BIC (Bayesian information criterion).

2.1 Site log-likelihood matrix

As we review the related methods that are likelihood based, we find that the basic building elements for this class of methods are the site likelihood or equivalently the

site log-likelihood. However different methods calculate the site likelihood differently. Intuitively, the site likelihoods that are calculated on the true topologies that are underlying the corresponding sites should be the most informative on recombination detection. Thus all the phylogenetic based methods with sliding window approach, methods using likelihood ratio tests [11] and the GARD [21] try to calculate the site-likelihood using estimated trees for different blocks of sequences, which creates a computational burden and inaccuracy of the estimated tree topologies due to using a smaller part of the alignment. PLATO [10] is fast and simple because it is based on the site log-likelihood of the maximum likelihood phylogeny estimated over the full alignment. However it is not powerful and accurate enough due to the weak signals in the site log-likelihood calculated on the ML tree only.

The PBC method is based on a site log-likelihood matrix which is calculated over a number of plausible tree topologies with tree branch and model parameters all estimated using the whole alignment. This idea works because for any tree topology that may not be exactly the true tree topology, the sites that are from different trees have support to the topology in consideration with different strength. The change of support to a tree along the sequence contains some information to the underlying true process. With multiple trees included in our site log-likelihood matrix, this signal can be reinforced and thus can help to identify the block structures along the sequence with different underlying evolutionary processes.

We can also view this as the joint influence of multiple sites to the log-likelihood on any given tree topology. The idea of local influence was first developed by Dennis Cook [5]. Local influence analysis is well established in statistical literature and is an important method for both model and data sensitivity analysis. The basic idea is to find the perturbation to which the inference of the model is most sensitive. This kind of sensitivity analysis can usually be achieved by first designing a perturbation scheme under which the concerned perturbation to either data or model can be mathematically expressed by a numerical vector. Then the inference results based on either the total model likelihood or particular parameter estimate of interest under the perturbation can be compared with the inference results without the perturbation. Assuming the perturbation scale is small (local), the most sensitive direction

of the perturbation then can be calculated along the first order derivative of the perturbed parameter estimate or the second order derivative of the log-likelihood. The sensitivity analysis has been made possible to phylogenetic analysis now since the first and second order derivatives of phylogenetic log-likelihood to all parameters can be calculated [17].

For the recombination detection problem, suppose we have a tree topology τ which is a good approximation to one of the true underlying tree topologies, then most of the sites in one block of the alignment will have relatively higher log-likelihood support to τ than the other block of the alignment. Take the maximized log-likelihood of tree τ over the whole alignment as a target statistic $T = l(\tau, \hat{\theta}) = \sum_{i=1}^{n} l_i(\tau, \hat{\theta})$, where $\hat{\theta}$ is the MLE of all the parameters including tree branch lengths and all the other model parameters, and $l_i(\tau, \hat{\theta})$ is the ith site log-likelihood evaluated under τ and $\hat{\theta}$. Now consider a perturbation scheme of the data as discussed in [17] by assigning a weight ω_i to the ith site of the data. The weight can be thought of as the frequency with which that site pattern was observed, which does not necessarily need to be an integer. The log-likelihood under the new weights for tree topology τ can be now re-written as $l(\tau, \theta_\omega) = \sum_{i=1}^{n} \omega_i l_i(\tau, \theta_\omega)$, where $\omega = (\omega_1, \cdots, \omega_n)$. The MLE of the parameters under this weighted log-likelihood is denoted as $\hat{\theta}_\omega$. The local weighting that most dramatically change the target statistic T lies in the direction $\frac{\partial T_\omega}{\partial \omega}$. It turns out that the most influential direction is proportional to the site log-likelihood, through the following steps:

$$\frac{\partial T_\omega}{\partial \omega_j} = \frac{\partial [\sum_{i=1}^{n} \omega_i l_i(\tau, \hat{\theta}_\omega)]}{\partial \omega_j} \tag{2.1}$$

$$= l_j(\tau, \hat{\theta}_\omega) + \sum_{i=1}^{n} \omega_i \frac{\partial l_i(\tau, \hat{\theta}_\omega)}{\partial \hat{\theta}_\omega} \frac{\partial \hat{\theta}_\omega}{\partial \omega_j} \tag{2.2}$$

Evaluating this derivative at $\omega = (1, \cdots, 1)$ and $\hat{\theta}_\omega = \hat{\theta}$ which corresponds to the point of T_ω without perturbation, we have $l_j(\tau, \hat{\theta}_\omega) = l_j(\tau, \hat{\theta})$ and $\sum_{i=1}^{n} \omega_i \frac{\partial l_i(\tau, \hat{\theta}_\omega)}{\partial \hat{\theta}_\omega} = \frac{\sum_{i=1}^{n} l_i(\tau, \theta)}{\partial \theta}|_{\hat{\theta}} = 0$.

A block of sites that have large joint local influence to the total log-likelihood for the tree topology τ means that there is some systematic difference on the evolutionary process for this block of sites. One possibility of this large local influence

is a recombination event, or alternatively it may mean some changes in the positive selection pressure. Changes in selection pressure will often have similar influence to likelihood under all trees, while recombination events will lead to sites with positive influence under some trees and negative influence under others. It is therefore often possible to identify the block corresponding to recombination, as opposed to positive selection.

As an example, we show some simple simulation results on the site log-likelihood. We simulate a DNA sequence data set D with sequence length 600 for 4 taxa. The sites in the following intervals: $([1, 150], [201, 300], [451, 600])$ are generated from tree τ_1; whereas the sites in the intervals $([151, 200], [301, 450])$ are generated from tree τ_2. Site log-likelihoods of τ_1 and τ_2 and the differences of site log-likelihoods of τ_1 and τ_2 are shown in Figure 2.1 and Figure 2.2 respectively. From these figures, we can clearly observe the patterns of site supports to two different tree topologies. In this example, the two true tree topologies are used to illustrate the direct difference of the site support. The site log-likelihood matrix is obtained using standard software; for the examples in this thesis, we used PAML (Yang, 1997) [35].

In Chapter 3, we use the same simulation design as in [2] to simulate 1000 data sets. As another example of site log-likelihood, we show in Figure 2.3 the site log-likelihood for 1000 sites under 6 different tree topologies from one of the simulated data with the first 500 sites generated under one tree and the remaining 500 sites generated under a different tree. In order to show the pattern more clearly, we have centralized the site log-likelihood for each site by the mean site log-likelihood. The total number of taxa is 40. The details of the simulation will be described later. The site log-likelihood are calculated on 6 trees of which none is exactly equal to any of the two true topologies. The RF distances [26] of the tree on which the site log-likelihood is calculated from both true trees are labelled below the horizontal axis. RF distance is a commonly used distance measure for two unrooted phylogenetic trees on the same set of taxa, it calculates the number of the binary patitions of the taxa which are compatible with one tree but not the other tree. The top two trees have smaller RF distance to one of the two true trees, the rest of the trees have larger RF distances with both true trees. It can be seen that even though the tree is far away from either of the true topologies, the different blocks of sites show clear block

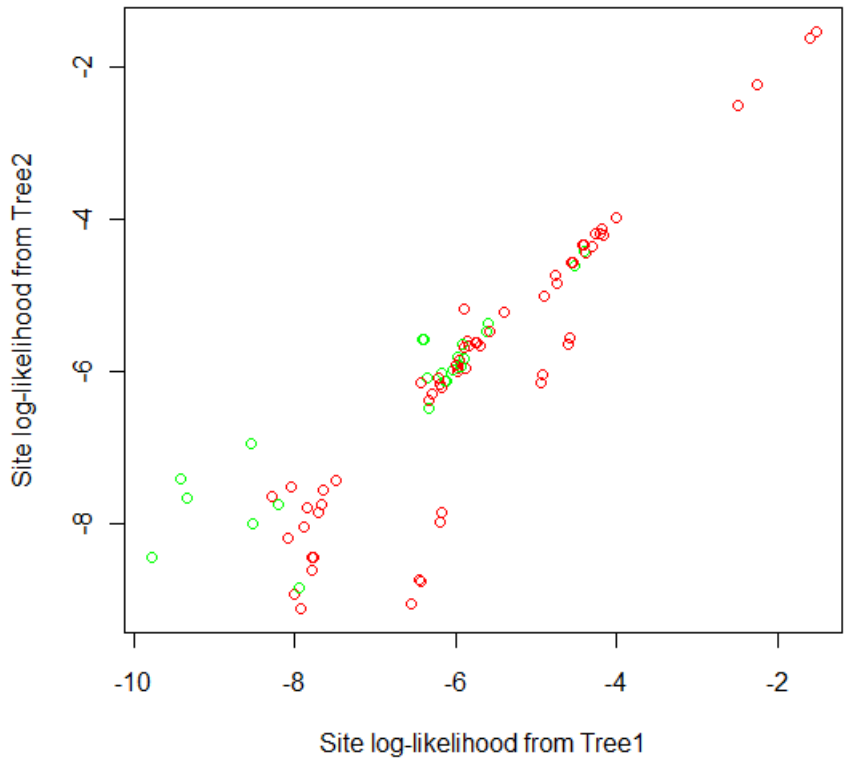

Figure 2.1: Site log-likelihood under Tree 1 and Tree 2: green points are sites simulated under Tree 2, red points are sites simulated under Tree 1.

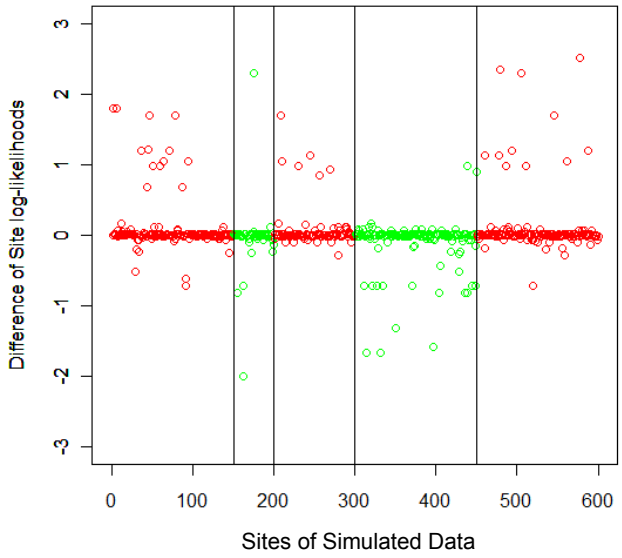

Figure 2.2: Site log-likelihood difference of Tree 1 and Tree 2: green points are sites simulated under Tree 2, red points are sites simulated under Tree 1.

patterns of support to the tree. With all these signals collected together in our site log-likelihood matrix, we have gained great power to find the block cluster patterns which provides a solid basis for recombination detection.

Since our method is based on the site log-likelihoods calculated using the whole data set on several tree topologies, which involves maximizing the likelihood only once for each tree topology, this is generally much faster than the methods that fit a tree on each block of sites, such as window sliding methods [11] or GARD [21]. It is also much faster than the methods based on a mixture model and a hidden Markov model [2] which need to maximize the likelihood over either a mixture model over several topologies or estimate a hidden Markov model over several tree topologies. Our method has some similarities to the mixture model in that the basic idea of both methods are to perform a block clustering on a site log-likelihood or site likelihood matrix, however these site likelihoods are fundamentally different.

2.2 Phylogenetic block clustering method

Our PBC method for detecting recombination starts by constructing the site log-likelihood matrix, which is defined as follows. Suppose we have identified a short-listed set of tree topologies, denoted as $\tau_1, \tau_2, ..., \tau_d$; Given data X, over the whole

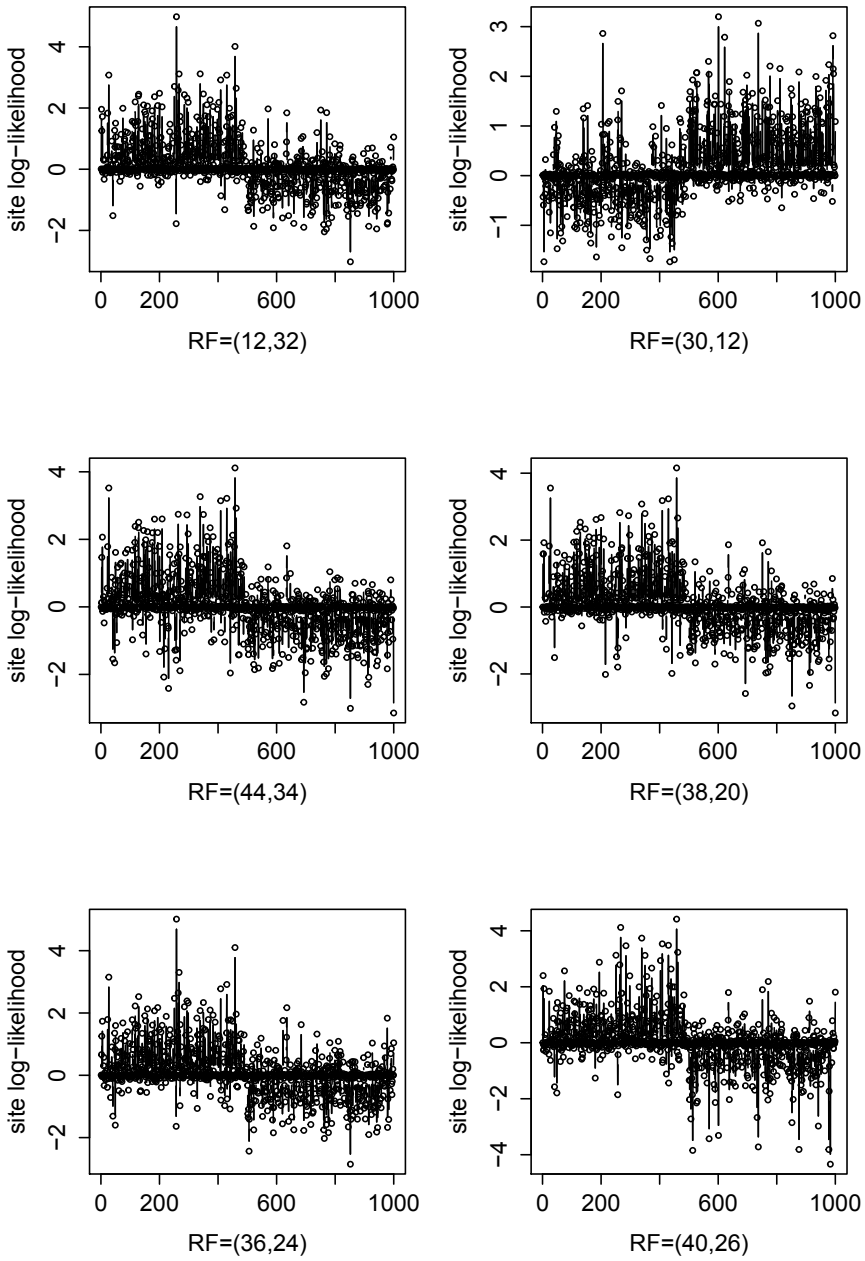

Figure 2.3: Site log-likelihood for 1000 sites under 6 different tree topologies, the true breakpoint is at site 500. The RF distances of the tree on which the site log-likelihood is calculated from both true trees are labelled below the horizontal axis.

sequence, for each tree topology τ_i, $i = 1, ..., d$, we can get the MLE for all the model parameters for the specified model. Denote the MLE of all model parameters for tree τ_i as $\hat{\theta}_i$, and MLE for branch lengths as $\hat{t}_i$. Then we have the site log-likelihood matrix $L = (l_{ij})$, where l_{ij} is the log-likelihood value of the j^{th} site given $(\tau_i, \hat{\theta}_i, \hat{t}_i)$, $i = 1, ..., d$; $j = 1, ..., n$.

We expect columns corresponding to sites for different trees to have different expected values. If the columns of this matrix are clustered as we expect, we need a tool to identify these clusters. Since the clusters consist of consecutive sites in the sequences, we can identify them by giving the breakpoints over the sequence. We introduce a new block clustering method to find these breakpoints.

Given a $d \times n$ matrix X, we want to divide this into a number, k, of blocks $B_1, ..., B_k$, where B_i is an interval $x_{[a_i,b_i]} = \{x_{a_i}, x_{a_i+1}, ..., x_{b_i-1}, x_{b_i}\}$. Our block-clustering method aims to maximize an objective function with the additional constraint that the clusters be blocks of consecutive columns. Our objective function is the maximum likelihood for a fixed set of parameters $(\hat{\theta}_i, \hat{t}_i)$ for each tree which are estimated from the whole data, and where each block is allowed to have a different tree topology. That is we maximize the sum of blocks of site log-likelihoods with different rows for different blocks:

$$W(k) = \underset{B}{\operatorname{argmax}} \left\{ \sum_{i=1}^{k} \max_{l} \left(\sum_{j \in [a_i, b_i]} X_{lj} \right) \right\} \tag{2.3}$$

Where a_i and b_i are the starting point and ending point in the block B_i.

Our algorithm first divides the whole sequence into many blocks which gives an initial set of breakpoints, then we refine the positions of these breakpoints by locally optimizing the position of each point. Finally we use the model selection criterion such as AIC, AICc or BIC to decide which breakpoints are significant.

To get a good final result, we need a good initial set of breakpoints. We find this by starting with the whole block and optimally subdividing one block at a time by exhaustive search, until we have all blocks optimal. At each stage, we subdivide the block which increases the objective function the most. For the analyses in this thesis, we restrict minimum block size to 30 to avoid finding very small regions which are not statistically significant. We stop when no subdivision can increase the objective

function any more. The exhaustive search is not only possible but also very fast, because it only involves calculating the sum of the site log-likelihood for each tree for each possible breakpoint.

To test each b_i is optimally positioned between a_i and b_{i+1} , we only need to consider the data points $x_{[a_i,b_{i+1}]}$. We can quickly find this optimal b_i by an exhaustive search. This provides a greedy searching procedure, where, starting from an initial block clustering, we reposition each b_i optimally between a_i and b_{i+1}, until each b_i is locally optimal. A block-clustering is locally optimal if each b_i is optimally positioned between a_i and b_{i+1}. Since each step increases the objective function, this procedure is guaranteed to converge.

2.2.1 Algorithm for the Block-clustering method

In this section, we give a detailed algorithm for our block-clustering method, as summarized below:

1. Initializing: Start with a given site log-likelihood matrix X, compute the cumulative site log-likelihood matrix. The cumulative site log-likelihood matrix is defined as $C_{ij} = \sum_{l=1}^{j} X_{il}$, so the objective function for fixed k is: $W(b_1, ..., b_{k-1}) = \sum_{j=1}^{k} \max_i (C_{ib_j} - C_{ib_{j-1}})$, where $b_0 = 0$, and $b_k = N$.

2. Adding breakpoints: Find the best new breakpoint sequentially by adding one breakpoint each time over the sequence, such that the objective function is maximized at each step. It takes k-1 steps to generate a list of k-1 breakpoints $\{b_1, \ldots, b_{k-1}\}$, where $0 = b_0 < b_1 < b_2 < \ldots < b_{k-1} < b_k = N$;

3. Refine the breakpoints sequentially: For each breakpoint b_j, find the new optimum cutting point, b_j^{new} by repositioning b_j between b_{j-1} and b_{j+1}, such that the objective function is maximized. Remove b_j if $W(\ldots, b_{j-1}, b_j^{new}, b_{j+1}, \ldots) - W(\ldots, b_{j-1}, b_{j+1}, \ldots)$ is less than a pre-specified tolerance δ, then the total number of blocks will be reduced by one. Repeat this step until convergence.

4. Testing the significance of the breakpoints: Step 3 generates a list of plausible breakpoints, to reduce them to a final list, we use the AIC method [1] to test the significance for each breakpoint. (Note that we refit the model parameters and

tree branch length parameters for the corresponding trees for each block when calculating AIC, this is according to the definition of AIC). We first calculate an AIC value for all plausible breakpoints as a baseline value, then we calculate an AIC value for the circumstance without the specific breakpoint B_i sequentially, if the AIC value is less than the baseline value, then the corresponding breakpoint is not significant according to AIC, we remove it and go back to step 3.

5. Return the list of breakpoints.

The initial set of tree topologies on which the site log-likelihood matrix X is based on can be estimated from different sections of the sequence. In our analysis, we use sliding window approach to estimate a set of tree topologies by the BioNJ [9] method. Alternatively any method that can provide a confidence region for the tree topologies should work too.

Chapter 3

Simulation Results

We follow the simulation design from Boussau, Guéguen and Gouy (2009) [2], which makes it easier to directly compare to the mixture model (MM) method and the hidden Markov model (Phylo-HMM) method by Boussau, et al. (2009) [2]. We also run GARD [21] on these simulated data to compare with the performance of PBC, MM and Phylo-HMM methods.

3.1 Simulation design

We choose the first 100 40-taxon trees from the 5000 "true 40-taxon trees", together with their branch lengths from the PhyML website. For each of these 100 trees, we perform a random sub-tree Prune and Regraft operation (SPR). The SPR procedure randomly selects a subtree from the original tree and then attaches it to a new branch. This way we get 100 pairs of trees. The RF distances of these 100 pairs of trees range from 2 to 22 as given in Figure 3.1. Each pair of these trees will serve as the true underlying two phylogenies in a DNA sequence alignment with one breakpoint.

We simulate nucleotide data based on a GTR model with the Gamma rate variation parameter fixed at 0.8. The rate matrix parameters are set to equal to the estimates from the globin pseudogenes data in [34] with $a = 0.987, b = 0.11, c = 0.218, d = 0.243, e = 0.395, f = 1$ and the equilibrium frequency parameters $\pi_T = 0.308, \pi_C = 0.185, \pi_A = 0.308, \pi_G = 0.199$. Indelible [8] is used to simulate the DNA sequences. From each pair of trees, we simulate 10 DNA sequence data sets, with sequence length for each data set fixed as 1000-nucleotides. Among these 10 data sets, 9 of them have one breakpoint positioned after site 100, 200, ... 900, and the last data set has no breakpoint. This way, for each true breakpoint at position L ($L = 100, 200, ..., 900$), we get 100 simulated data sets, each of which generated from

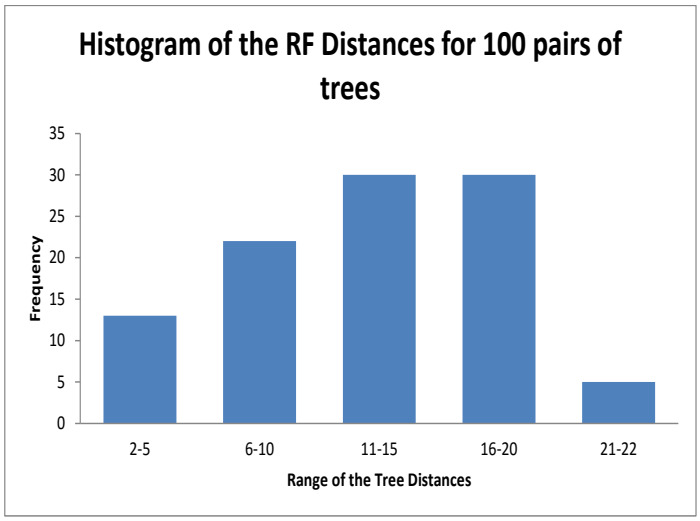

Figure 3.1: The RF distances of 100 pairs of trees, each pair of the trees serve as the true underlying two phylogenies in a DNA sequence alignment with one breakpoint.

one pair of the 100 pairs of trees, and furthermore, we have 100 trees with no breakpoints generated from the first of each pair of trees (these are labelled as the $L = 1000$ case). The different RF distances among these 100 pairs of trees determine the difficulty of the breakpoint detection problem. Generally, if the two true trees have larger RF distances, the problem is a relatively easy one and the hardest problem is when the RF distance between two underlying trees is very small.

3.2 Analysis of the simulated data

In our algorithm, we need an initial set of topologies to get the site log-likelihood matrix. Naturally the algorithm will work better if we can include tree topologies which are close to both true underlying topologies for each data set. In order to get this initial set of tree topologies, we use the following procedure.

First, for each data set, a set of topologies are estimated using blocks of the DNA alignment of length 100 sites with the block moving along the DNA sequence in steps of 50 sites. Thus the first block includes sites 1-100, the second block includes sites 51-150, the third block includes sites 101-200, etc. This way we have 19 blocks for each data set. We then compute the pairwise distance matrices for each block under the F84 model with gamma rate variation across sites parameter equal to 1 using the

DNADIST program in the PHYLIP package [6]. Note that this is a misspecified model from the true model, which mimics the reality that we never know the true model and almost always use a more simplified model than the truth in phylogenetic analysis. Then 19 trees are estimated using BioNJ [9] based on these pairwise distances for each data set. The resulting 19 estimated tree topologies for each data set could include duplicates. The duplicates can be removed at this step; however, it does not cause any problems for our method to include them all. Thus we include all these 19 topologies. The site log-likelihood matrices for the corresponding 19 estimated tree topologies for each data set are calculated using PAML, based on the HKY85 model with gamma rate variation across sites. The model parameters and tree branch lengths are estimated using the whole sequence, and again we include some model misspecification here to reflect the reality. We then apply the first three steps of our PBC algorithm on each data set, we obtain an initial list of breakpoints for each data set. With this initial set of breakpoints, we re-estimate the tree topologies for each block of the alignment using exactly the same settings as we did above, i.e. estimate trees by BioNJ based on the pairwise distances calculated by DNADIST. We then add these newly estimated trees to the set of 19 trees for each data set. This added 2 to 6 trees for different data sets. This expanded set of trees for each data set are now the initial set of tree topologies. A final list of breakpoints can be then obtained by applying the PBC method on the site log-likelihood matrix based on this expanded set of trees.

The above procedure doesn't guarantee that the set of estimated trees include the true trees. This is especially true when the number of taxa is large. This raises the question of whether our method could be improved by using a more sophisticated tree-search method. In order to know how much the PBC method could perform better with a better tree-search method, for each data set, we add the two true tree topologies to the set of 19 tree topologies instead of adding the estimated tree topologies based on the initial set of breakpoints, and re-apply the PBC method on the new site log-likelihood matrices for each data set. Since the true trees are never known in reality, we call this method PBC Oracle (PBC-O).

For comparison, we also apply the mixture model (MM) method and the hidden

Markov model (Phylo-HMM) method [2] using the default setting of two tree topologies and also setting the number of tree topologies equal to 4, 10 and 20 for each method. As for the PBC methods, BioNJ [9] is used by both the MM and Phylo-HMM to build the starting trees and the model used in the analysis is HKY85. Finally the GARD [21] multiple-breakpoint searching procedure is applied to our simulated data sets.

3.3 Results of the simulation analysis

There should be one breakpoint detected in the first nine scenarios, and no breakpoint detected in the last scenario. Figure 3.2 shows the number of times among 100 simulations for each scenario that the correct number of breakpoints were detected by different methods, in the order PBC, PBC-O, MM2, MM4, Phylo-HMM2, Phylo-HMM4, GARD, where Phylo-HMM2 and Phylo-HMM4 refer to the Phylo-HMM methods with the number of tree topologies set as 2 and 4 respectively; MM2 and MM4 are similarly defined. The blue or green bars below the horizontal axis in Figure 3.2 show the number of times the methods detect the correct number of breakpoints, the bars above the horizontal axis show the number of times that the corresponding methods detect no breakpoint (false negative) or more breakpoints than the truth (false positive). It is clear that the PBC method with the true trees included is the most accurate, but since true trees are never known in reality, this is not really achievable. The PBC method with estimated trees performs only slightly worse than the PBC method with true trees included in each scenario. The overall power of the PBC method is the highest and the false negative and false positive rates of the PBC methods are the lowest among the several methods we have included in this analysis. For some of the datasets with breakpoints near the middle, Phylo-HMM2 does outperform PBC, however, the performance of Phylo-HMM2 decays badly as the breakpoint moves nearer to the end of the alignment. This is not surprising because Phylo-HMM estimates its initial starting trees based on dividing the sequence into two equal pieces, so if this is indeed the case, then it's initially estimated trees will be close to the truth. However, when the breakpoint is actually towards one end of the alignment, then the initial trees will be poorly estimated and the performance will suffer. The performance of MM2 and Phylo-HMM2 are similar to that shown in [2]

because we followed their simulation design. Note that even when the number of trees is set as 2, which has given these methods a great advantage for these simulations with only a single breakpoint, both MM2 and Phylo-HMM2 don't perform as well as PBC. When we set the number of trees to 4, MM4 has a lower rate of correctly detecting one breakpoint than both MM2 and Phylo-HMM2. The performance of Phylo-HMM4 decayed even more dramatically. For the last column, when the truth is that there is no breakpoint, the PBC methods correctly indicate no breakpoint 100 times, but both the MM and the Phylo-HMM detected many cases with one breakpoint. The GARD has comparable results to PBC when the true breakpoint is in the middle of the sequence alignment, but it quickly loses power when the true breakpoint moves towards the ends of the sequence alignments. It didn't detect any case with more than one breakpoint, also didn't detect any breakpoint when the truth is no breakpoint. The false positive rate is controlled very well by GARD, but the power is much worse than the PBC method. Like the other methods, the PBC performance for scenarios close to the either end is slightly worse than scenarios close to the middle. This is expected because shorter sequence lengths do not always give enough signal to indicate the recombination event. However, the performance of PBC decays less than the performance of other methods in these cases.

Figure 3.3 shows the boxplots of the estimated breakpoint positions for cases where exactly one breakpoint is detected for the first nine scenarios and the estimated positions when the truth is no breakpoint. From the left to right for each scenario in Figure 3.3 the methods are in the order of PBC, PBC-O, MM2 , Phylo-HMM2 and GARD. The best prediction for the breakpoint positions is given by PBC-O. The PBC performance in terms of accuracy of estimated breakpoint positions is very similar to PBC-O. With the true trees included in the tree set, PBC performs only slightly better according to both detection of the correct number of breakpoints and estimation of their locations. Therefore, there is limited scope for improving the method by implementing more sophisticated tree searching methods, at least for the situations tested in this simulation. Both MM2 and Phylo-HMM2 are much worse in that there are many outlier predictions which are far away from the true breakpoint positions. When the true breakpoint positions are at 100, 800 and 900, the majority of the predictions by MM2 and Phylo-HMM2 are far away from the truth. The GARD

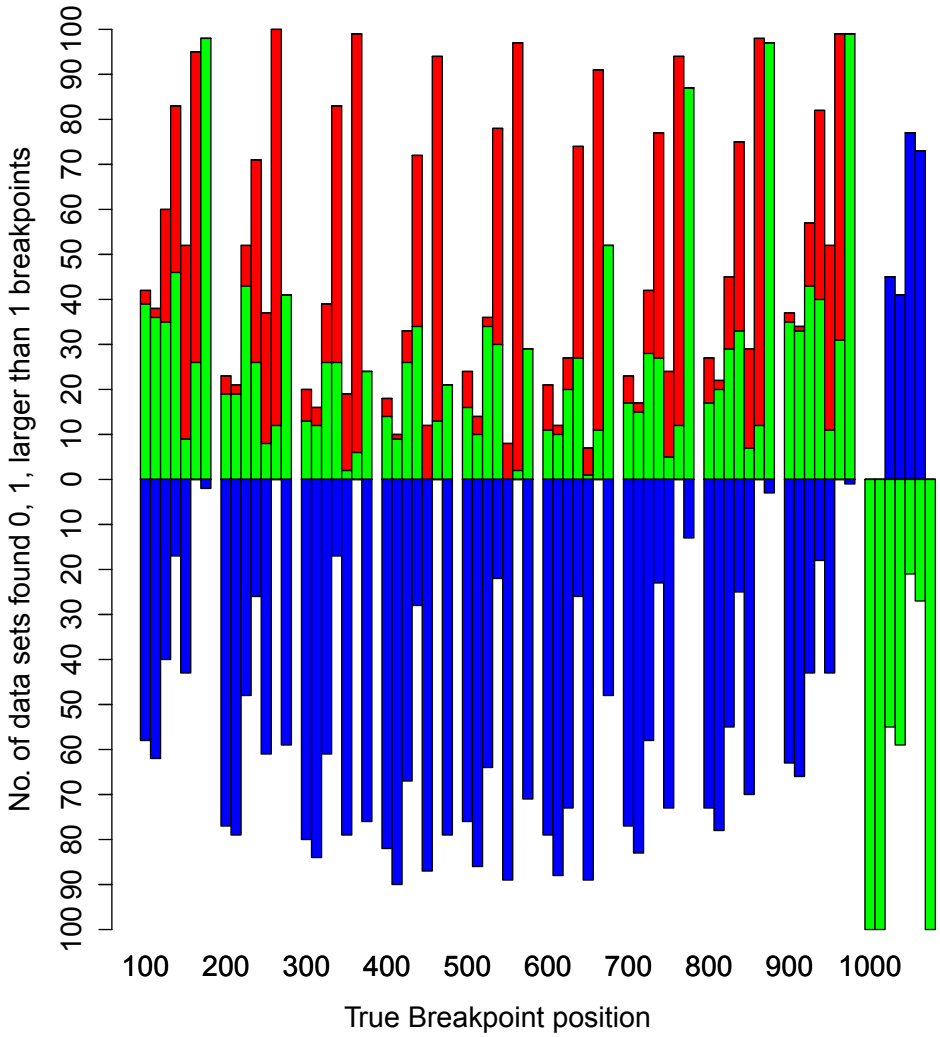

Figure 3.2: For each true breakpoint position, the number of data sets found with 0 (green), 1 (blue) and more than 1 (red) breakpoints. The methods from left to right for each position is in the order of PBC, PBC-O, MM2, MM4, Phylo-HMM2, Phylo-HMM4, GARD. The bars below indicate the number of data sets with the correct number of breakpoint found, the bars above indicate either false positive or false negative rates.

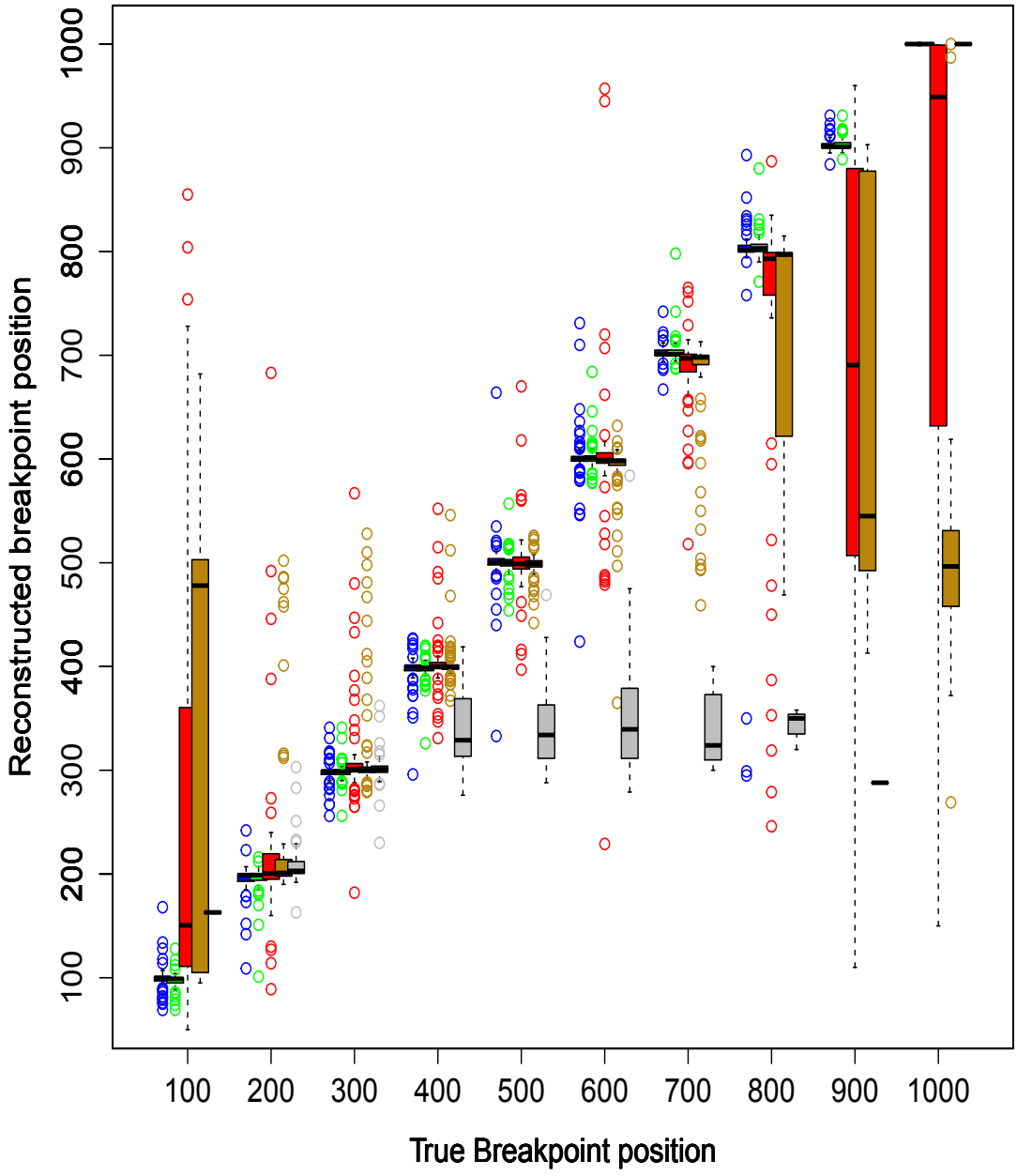

Figure 3.3: Among the data sets for which one breakpoint was found for the first 9 scenarios, the boxplots of the estimated breakpoint positions are plotted for methods PBC (blue), PBC-O (green), MM2 (red), Phylo-HMM2 (brown) and GARD (grey). For the last scenario, the MM2 and Phylo-HMM2 estimated many times with one breakpoint, the boxplot of the estimated positions are also shown.

behaves strangely in that most of the predictions fall away from the truth. The reason for this is not clear.

In addition to the above comparisons on accuracy, we also found that the PBC method requires the least computation time. We recorded the machine time in running different methods on these 1000 simulated data sets; the times are listed in Table 3.1. Notice that Phylo-HMM10 and Phylo-HMM20 seem to have great difficulty with the convergence of the program, thus only output the analysis results for a small fraction of the data sets. Phylo-HMM2 and Phylo-HMM4 can output results for majority of the data sets but are quite time consuming. The GARD is the most computationally intensive method. For PBC-O, we only analyzed 960 data sets, since the two true trees are already in the originally estimated 19 trees for 40 of the data sets, thus we didn't re-analyze these data sets. The recorded times for PBC and PBC-O are only for the breakpoint detection program after inputting the site log-likelihood matrix for each data set. That is, we have not included the time applying BIONJ to estimate the tree topologies or the time used to calculate the site log-likelihood matrix with PAML. However, this pre-processing work cost no more than 10 hours for 1000 data sets, so even including this time, PBC will outperform the other methods on computation time.

Table 3.1: Running time of different methods on the 1000 simulated data sets

Methods	No. of data sets run out by the Method	Time
PBC	1000	102 Mins 46.234 seconds
PBC-O	960	57 Mins 1.401 seconds
MM2	1000	3691 Mins 74 seconds
MM4	1000	2296 Mins 21.792 seconds
MM10	1000	2550 Mins 35.208 seconds
MM20	1000	5557 Mins 47.006 seconds
Phylo-HMM2	972	1465 Mins 10 seconds
Phylo-HMM4	967	1909 Mins 1 seconds
Phylo-HMM10	277	3112 Mins 46 seconds
Phylo-HMM20	185	28303 Mins 43 seconds
GARD	1000	23424 Mins 47 seconds

Through this round of the simulation, we have shown that the PBC method outperforms the mixture model based and hidden markov model based methods, and the GARD method, in all aspects, including: higher power with lower false positive

rate; accuracy of the predicted breakpoint positions; and in terms of computation time, even including the time for topology estimation and the site log-likelihood calculation.

Chapter 4

Real data analysis

We applied our method, in comparison with several other methods, on three real data sets which include two DNA multiple sequence alignments and one protein multiple sequence alignment.

4.1 The *Zea Mays* data

Moniz and Drouin [27] detected some recombination events among forty-four *actin genes* from five angiosperm species in 1996. They also mentioned that recombination rate varied between 2% – 12% within the *Zea maize* species. These *Zea Mays* data, which have been analyzed by Husmeier and McGuire (2003) [13], consist of four *Zea Mays actin* gene sequences of 1007 nucleotides. The original four sequences are downloaded from the GenBank: the *maize* sequence names and accession numbers are: Maz56 (U60514), Maz63 (U60513), Maz89 (U60508) and Maz95 (U60507). Since Moniz and Drouin [27] only analyzed underlined exons of Maz56 and Maz81, for comparison, we use clustalw2 to delete the alignment gaps and get the same exons of the four *Zea mays actin* gene sequences. There are only 4 taxa in the data set, we use all three unrooted tree topologies to compute the site log-likelihood matrix. We first apply our PBC method and the GENECONV method (Sawyer, 1989) [28] which was implemented by the RDP program (Martin et al, 2010) [19], then we compare the results of the PBC method and the GENECONV method (Sawyer, 1989) [28] with the other published results by Moniz and Drouin [27] and the two methods by Husmeier and McGuire (2003) [13].

Table 4.1 shows the comparison results of five different methods. The PBC method finds that the phylogeny changes from tree 1 (Figure 4.1) to tree 3 (Figure 4.2) at site 636 of this multiple alignment sequence. The recombination event was placed at site 620 by the maximum likelihood approach on a Hidden Markov Model, while sites 410 and 880 are the two recombination points detected by the GENECONV method

(Sawyer, 1989) [28]. Without considering other sequences, Moniz and Drouin [27] found a gene conversion event around site 876 between Maz56 and Maz63. Husmeier and McGuire [13] placed the gene conversion event around site 886 using the Bayesian approach on a Hidden Markov Model. We found that the phylogenetic signal (maximum site log likelihoods) is almost the same for tree 1 and tree 3 inside the region from site 636 to site 876. It seems hard to tell exactly where in this region to place the recombination event. All methods agree that there is a recombination event in this region.

Detecting Recombination	First Breakpoint	Second Breakpoint
PBC	636	None
Moniz&Drouin(1996)	876	None
HMM-ML(Husmeier and Wright, 2001)	620	None
Bayesian-HMM(Husmeier and McGuire, 2003)	886	None
GENECONV(Sawyer, 1989)	410	880

Table 4.1: Recombination points for *Zea Mays actin gene* by different methods

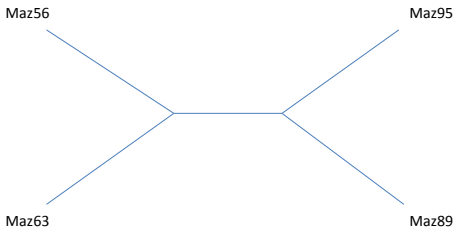

Figure 4.1: Tree 1

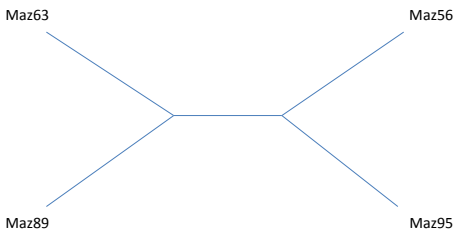

Figure 4.2: Tree 3

4.2 The *Neisseria argF* genes data

The *argF* gene is a kind of gene product which is the component of *ornithine transcarbamylase*. Different strains of *Neisseria argF* gene have shown different sequence variation (Zhou and Spratt, 1992) [36]. Recombination rate among the strains

Detecting Recombination	First Breakpoint	Second Breakpoint	Third Breakpoint	Fourth Breakpoint
PBC	189	None	531	None
Zhou&Spratt(1992)	202	507	538	None
HMM-ML(Husmeier and Wright, 2001)	200	None	None	749
Bayesian-HMM(Husmeier and McGuire, 2003)	195	510	545	750
GENECONV(Sawyer, 1989)	208	498	574	None

Table 4.2: Recombination points for *Neisseria argF* genes by different methods

of *Neisseria argF* gene for *ornithine transcarbamoylase* is around 0–1.3%, and 1.0% interspecies recombination was also found between the strain *Neisseria meningitidis* (HF116) and the strain *Neisseria gonorrhoeae* (FA19) (Zhou and Spratt, 1992) [36]. We pick 9 strains of *Neisseria argF* Gene, each with 787 nucleotides for this example. These DNA sequences are downloaded from GenBank. The GenBank access numbers and species names are: *Neisseria gohorrhoeae* - X64860, *Neisseria meningitidis* - X64863, X64864, X64866; *Neisseria cinera* - X64869, *Neisseria polysaccharea* - X64870, *Neisseria lactamica* - X64871, *Neisseria flavescens* - X64872, and *Neisseria mucosa* - X64873. Husmeier and McGuire [13] chose 4 out of these 9 strains for their analysis and detected some recombination events.

To get an initial set of trees, we use a sliding window technique so that a set of topologies are estimated using blocks of the DNA alignment of length 100 sites with the block moving along the DNA sequence in steps of 50 sites. Thus the first block includes sites 1-100, the second block includes sites 51-150, the third block includes sites 101-200, etc. This way we have 15 blocks for this data set. Since the sequence length is 787, the last two blocks use sequences of lengths 87 and 137 respectively. We then compute the pairwise distance matrices for each block under the F84 model with gamma rate variation across sites parameter equal to 1 using the DNADIST program in the PHYLIP package [6]. Tree topologies are estimated by BioNJ [9] and used

to construct the site log-likelihood matrix. As in the simulations, we apply PBC to find an initial set of breakpoints, then we estimate additional trees from the regions identified by these breakpoints, and add these trees to the site-likelihood matrix. We apply the PBC method on these 9 strains and compare the result from our method with the formerly published results from other methods.

Table 4.2 presents all the recombination points detected by the former research, and the results by GENECONV method [28] using RDP program (Martin et al, 2010) [19] and the PBC method. Most methods agree that there is a recombination event around site 200. Zhou and Spratt [36] used a different labeling scheme, they labeled the nucleotides from location 296 to location 1082 corresponding to the positions 1 to 787 here. They also mentioned in their paper that there is a very short region, corresponding to positions between 507 and 538, with genetic variation at 32.3%. Our method detects site 531 as the second recombination point, but no other recombination points on either side of this breakpoint. This is expected because our method was configured to only detect the recombinant sequences at least 30 nucleotides long. The first and forth breakpoints detected by HMM-ML are very similar to the ones detected by Bayesian-HMM. The first and second breakpoints detected by GENECONV are very similar to our method; Three breakpoints detected by Bayesian-HMM are all close to the ones detected by Zhou and Spratt (1992) [36]. However, Bayesian-HMM finds an extra recombination point which was not detected by PBC, GENECONV or Zhou and Spratt (1992) [36].

4.3 *Mycobacterial Genomes*

Genetic variation plays a very important role in *Mycobacterial* evolution, and has been detected in a data set consisting of 18 distinct *Mycobacterial* strains with 13 species, by various authors recently (Smith et al, 2012) [30]. Martin (2010) [19] applied RDP on a single concatenated DNA sequence from detectable homologous segments of 18 *Mycobacterial* strains and found 74 recombination events. This suggests that there are strong incongruence phylogenetic signals for these protein sequences. Our data set includes 1052 amino acid alignments with different numbers of strains for different proteins. The sequences are all from strains of *Mycobacterium tuberculosis* CDC1551. As before, we use sliding window technique to get an initial set of trees

so that a set of topologies are estimated using blocks of the amino acid sequences of length 100 sites with the block moving along the amino acid sequence in steps of 50 sites. We then compute the pairwise distance matrices for each block under the Dayhoff PAM matrix model with gamma rate variation across sites parameter equal to 1 using the PROTDIST program in the PHYLIP package [6]. Tree topologies are estimated by BioNJ [9] and used to construct the site log-likelihood matrix. The PBC method, Mixture Model methods and Hidden Markov Model Method (Boussau et al, 2009) [2] with the default choice of two trees are applied on each gene of the *Mycobacterial Genomes*.

Mixture Model / PBC Method	None	1	2	3	>3
None	14	209	24	16	68
1	13	224	15	15	40
2	0	134	7	14	41
3	4	63	4	6	18
>3	4	74	6	13	26

Table 4.3: Number of *Mycobacterial* protein data sets identified by different type of breakpoints of PBC method and MM2 method

The PBC method and the Mixture Model method detect the same number of breakpoints on 277 of the amino acid alignments (see the diagonal cells of Table 4.3). The numbers of the alignments in the lower triangular cells of the table show that PBC method finds more breakpoints than the Mixture Model method for total of 315 amino acid alignments. However, the Mixture Model method detects more breakpoints than PBC in the other 460 amino acid alignments (see the upper triangular cells of the table). Additionally, according to the first column of the lower triangle of the table, PBC detects breakpoints in 21 amino acid alignments where the Mixture model method doesn't find any breakpoint; Meanwhile, the Mixture model method detects breakpoints on 317 amino acid alignments where PBC doesn't find any breakpoint. All these are consistent with the results of our simulation study (see Figure 3.2) where the mixture model showed a much higher false positive rate and lower power than

PBC method. There is some agreement among all the breakpoints detected by both PBC and MM2 methods as shown in Figure 4.3. Among all the breakpoints detected by PBC method, about 17% of them are within 10 sites, about 22% of them are within 15 sites and about 27% of them are within 20 sites of breakpoints detected by the Mixture Model method.

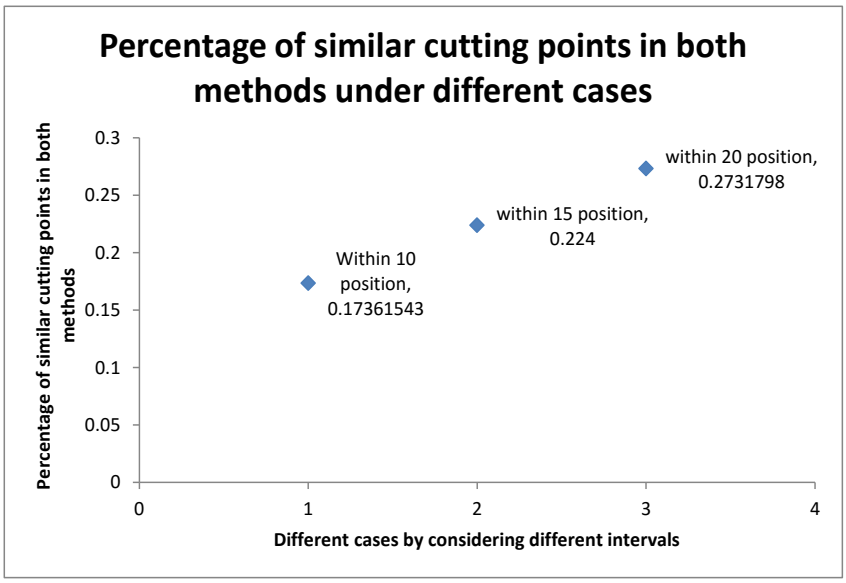

Figure 4.3: Among all the breakpoints detected by PBC method, agreement between MM2 and PBC methods on the estimated breakpoint positions for *Mycobacterial Genomes*

The PBC method and the Hidden Markov Model method detect the same number of breakpoints on 168 amino acid alignments (see the diagonal cells of Table 4.4). The numbers of the alignments in the upper triangular cells of the table show that Hidden Markov method finds more breakpoints than our PBC method for total of 839 amino acid alignments. However, the PBC method detects more breakpoints than Hidden Markov Model method from only 45 amino acid alignments (see the lower triangular cells of the table). Additionally, from the first row of the upper triangle the table, we see that the Hidden Markov Model Method detects breakpoints in 318 amino acid alignments where the PBC method doesn't find any breakpoint; Meanwhile, the

HMM Model / PBC Method	None	1	2	3	>3
None	13	71	17	36	194
1	20	37	6	14	230
2	5	6	0	2	183
3	6	2	1	0	86
>3	5	0	0	0	118

Table 4.4: Number of *Mycobacterial* protein data sets identified by different type of breakpoints of PBC method and HMM2 method

PBC method detects breakpoints on only 36 amino acid alignments where the Hidden Markov Model method doesn't find any breakpoint. Compared to the Mixture Model, there is much more agreement among all the breakpoint positions detected by both PBC and HMM2 methods as shown in Figure 4.4. About 71% of the breakpoints found by PBC are within 10 sites of breakpoints found by HMM2; about 80% are within 15 sites and about 86% are within 20 sites. This greater similarity may be partially explained by the fact that HMM2 estimated more breakpoints than MM2, so the chance of being close to one breakpoint estimate from HMM2 is larger. However, this does not seem sufficient to achieve this level of accuracy. It therefore seems that there is good agreement between breakpoints found by PBC and by HMM2.

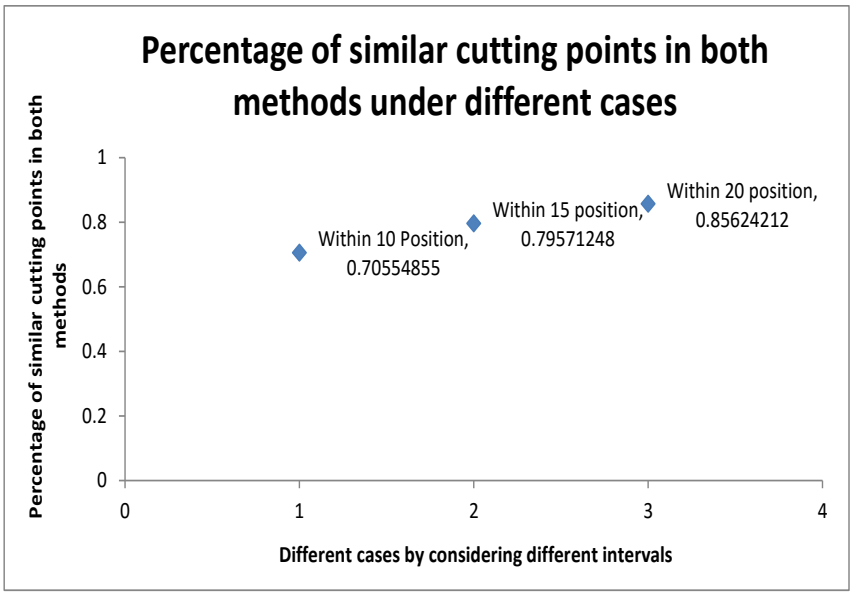

Figure 4.4: Among all the breakpoints detected by PBC method, agreement between HMM2 and PBC methods on the estimated breakpoint positions for *Mycobacterial Genomes*

Chapter 5

Conclusions and Future work

We have developed a very simple and yet very fast and effective method for detecting recombination, named the Phylogenetic Block Clustering (PBC) method. The Phylogenetic Block Clustering method is based on the site log-likelihood matrix on a number of tree topologies for detecting the recombination points in both DNA and Amino Acid sequence data. By using the site log-likelihood under different trees with all tree branch and model parameters estimated over the whole sequence, the PBC method is able to exploit the weak signals of each site's support to each tree and harvest the joint influence of blocks of sites to different trees. This effectively enhanced the accuracy of the clustering algorithm developed in this thesis in detecting recombination events. The method was tested on both simulated data sets and real data sets. The simulation results showed that it has the ability to detect accurately the positions of breakpoints most of the time. Meanwhile, it has very low false positive rate. The method is computationally very efficient because the method doesn't need to re-estimate the tree topologies and the model parameters many times and yet it is a likelihood-based method. The results from both simulated data sets and real data sets show that the site log-likelihood matrix that our PBC method is based on contains abundant information and is a useful tool for detecting recombination events.

With the limited simulation results shown in this thesis, we can conclude that PBC has high potential to rank as one of the best recombination detection methods. More simulations are needed to find in which situations it performs the best and in which situations it needs more tuning. The number of tree topologies in our simulation and real data analysis was fixed beforehand. It will be very useful to know how sensitive the results are to the initial number of topologies used, and to the accuracy of these trees. Our intuition is that including more trees could further enhance the signals and thus increase the power of the method but won't greatly increase the false positive

rate as it does for the mixture model or hidden markov model, because of the way the clustering algorithm is designed. Increasing number of the trees could also increase the running time, but only linearly, so the method should scale well. The best balance needs to be constructed through further simulation studies.

We also fixed the minimum of the block of the recombinant sequences as 30, since it is very unlikely that sequences shorter than this would be confirmed by AIC. Therefore, identifying such short sequences could greatly increase computation time, while having almost no effect on the overall results. This decision was somewhat heuristic, and more careful study of how choice of minimum block size affects the results and computation time of the method should be performed in future.

There are a number of ways that the current method can be improved. The most obvious is to improve the initial tree searching method to improve the chance of including the true trees among the set of trees used. From the simulation with PBC-Oracle, we see that improved tree search has the potential to improve results, but that the improvement that could be attained is limited. Additionally, the method currently uses AIC to test the significance of the breakpoints, but there are a number of alternatives available, and more work needs to be done on determining which alternative performs best for this purpose.

Bibliography

[1] Hirotogu Akaike. Information theory and an extension of the maximum likelihood prin. 1973.

[2] Bastien Boussau, Laurent Guéguen, and Manolo Gouy. A mixture model and a hidden markov model to simultaneously detect recombination breakpoints and reconstruct phylogenies. *Evolutionary Bioinformatics*, 5:67, 2009.

[3] Trevor C Bruen, Hervé Philippe, and David Bryant. A simple and robust statistical test for detecting the presence of recombination. *Genetics*, 172(4):2665–2681, 2006.

[4] Kenneth P Burnham and David Anderson. Model selection and multi-model inference. *A Pratical informatio-theoric approch. Sringer*, 2003.

[5] Ralph D Cook. Assessment of local influence. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 133–169, 1986.

[6] Joseph Felsenstein. Phylogeny programs. *Internet address: http://evolution. gs. washington. edu/phylip/software. html*, 1995.

[7] Joseph Felsenstein and Gary A Churchill. A hidden markov model approach to variation among sites in rate of evolution. *Molecular Biology and Evolution*, 13(1):93–104, 1996.

[8] William Fletcher and Ziheng Yang. Indelible: a flexible simulator of biological sequence evolution. *Molecular biology and evolution*, 26(8):1879–1888, 2009.

[9] Olivier Gascuel. Bionj: an improved version of the nj algorithm based on a simple model of sequence data. *Molecular biology and evolution*, 14(7):685–695, 1997.

[10] Nicholas C Grassly and Edward C Holmes. A likelihood method for the detection of selection and recombination using nucleotide sequences. *Molecular Biology and Evolution*, 14(3):239–247, 1997.

[11] Edward C Holmes, Michael Worobey, and Andrew Rambaut. Phylogenetic evidence for recombination in dengue virus. *Molecular biology and evolution*, 16(3):405–409, 1999.

[12] Richard R Hudson. Two-locus sampling distributions and their application. *Genetics*, 159(4):1805–1817, 2001.

[13] Dirk Husmeier and Gráinne McGuire. Detecting recombination in 4-taxa dna sequence alignments with bayesian hidden markov models and markov chain monte carlo. *Molecular Biology and Evolution*, 20(3):315–337, 2003.

[14] Dirk Husmeier and Frank Wright. Detection of recombination in dna multiple alignments with hidden markov models. *Journal of Computational Biology*, 8(4):401–427, 2001.

[15] Daniel H Huson. Splitstree: analyzing and visualizing evolutionary data. *Bioinformatics*, 14(1):68–73, 1998.

[16] Ingrid B Jakobsen, Susan R Wilson, and Simon Easteal. The partition matrix: exploring variable phylogenetic signals along nucleotide sequence alignments. *Molecular biology and evolution*, 14(5):474–484, 1997.

[17] Toby Kenney and Hong Gu. Hessian calculation for phylogenetic likelihood based on the pruning algorithm and its applications. *Statistical applications in genetics and molecular biology*, 11(4), 2012.

[18] Darren Martin and Edward Rybicki. RDP: detection of recombination amongst aligned sequences. *Bioinformatics*, 16(6):562–563, 2000.

[19] Darren P Martin, Philippe Lemey, Martin Lott, Vincent Moulton, David Posada, and Pierre Lefeuvre. Rdp3: a flexible and fast computer program for analyzing recombination. *Bioinformatics*, 26(19):2462–2463, 2010.

[20] Gil McVean, Philip Awadalla, and Paul Fearnhead. A coalescent-based method for detecting and estimating recombination from gene sequences. *Genetics*, 160(3):1231–1241, 2002.

[21] Sergei LK Pond, David Posada, Michael B Gravenor, Christopher H Woelk, and Simon DW Frost. Automated phylogenetic detection of recombination using a genetic algorithm. *Molecular biology and evolution*, 23(10):1891–1901, 2006.

[22] David Posada. Evaluation of methods for detecting recombination from dna sequences: empirical data. *Molecular biology and evolution*, 19(5):708–717, 2002.

[23] David Posada and Keith A Crandall. Evaluation of methods for detecting recombination from dna sequences: computer simulations. *Proceedings of the National Academy of Sciences*, 98(24):13757–13762, 2001.

[24] David Posada and Keith A Crandall. The effect of recombination on the accuracy of phylogeny estimation. *Journal of molecular evolution*, 54(3):396–402, 2002.

[25] Paul D Rawson, Karen L Joyner, Keith Meetze, and Thomas J Hilbish. Evidence for intragenic recombination within a novel genetic marker that distinguishes mussels in the mytilus edulis species complex. *Heredity*, 77(6), 1996.

[26] David F Robinson and Leslie R Foulds. Comparison of phylogenetic trees. *Mathematical biosciences*, 53(1):131–147, 1981.

[27] Mario Md Sa and Guy Drouin. Phylogeny and substitution rates of angiosperm actin genes. *Molecular Biology and Evolution*, 13(9):1198–1212, 1996.

[28] Stanley Sawyer. Statistical tests for detecting gene conversion. *Molecular biology and evolution*, 6(5):526–538, 1989.

[29] John M Smith and Noel H Smith. Detecting recombination from gene trees. *Molecular biology and evolution*, 15(5):590–599, 1998.

[30] Silvia E Smith, Patrice Showers-Corneli, Caitlin N Dardenne, Henry H Harpending, Darren P Martin, and Robert G Beiko. Comparative genomic and phylogenetic approaches to characterize the role of genetic recombination in mycobacterial evolution. *PloS one*, 7(11):e50070, 2012.

[31] Peter HA Sneath, Michael J Sackin, and Richard P Ambler. Detecting evolutionary incompatibilities from protein sequences. *Systematic Biology*, 24(3):311–332, 1975.

[32] Georg F Weiller. Phylogenetic profiles: a graphical method for detecting genetic recombinations in homologous sequences. *Molecular Biology and Evolution*, 15(3):326–335, 1998.

[33] Michael Worobey and Edward C Holmes. Evolutionary aspects of recombination in rna viruses. *Journal of General Virology*, 80(10):2535–2543, 1999.

[34] Ziheng Yang. Estimating the pattern of nucleotide substitution. *Journal of Molecular Evolution*, 39(1):105–111, 1994.

[35] Ziheng Yang. Paml: a program package for phylogenetic analysis by maximum likelihood. cabios 13, 555–556. *Supplementary material The following supplementary material is available for this article online: Fig. S*, 1, 1997.

[36] Jia J Zhou and Brian G Spratt. Sequence diversity within the argf, fbp and reca genes of natural isolates of neisseria meningitidis: interspecies recombination within the argf gene. *Molecular microbiology*, 6(15):2135–2146, 1992.

[37] Jian L Zhuang, Amanda E Jetzt, Guo L Sun, Hong Yu, George Klarmann, Yacov Ron, Bradley D Preston, and Joseph P Dougherty. Human immunodeficiency virus type 1 recombination: rate, fidelity, and putative hot spots. *Journal of virology*, 76(22):11273–11282, 2002.