Maximum Likelihood Estimation and Phylogenetic Tree based Backward Elimination for reconstructing Viral Haplotypes in a Population

Raunaq Malhotra 1,*, Steven Wu 2, Allen Rodrigo 2, Mary Poss 3, and Raj Acharya 1*

1 Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802
2 Department of Biology, Duke University, Box 90338, Durham, NC 27708
3 Department of Biology, The Pennsylvania State University, University Park, PA 16802

Received on XXXXX; revised on XXXXX; accepted on XXXXX

ABSTRACT

A viral population can contain a large and diverse collection of viral haplotypes which play important roles in maintaining the viral population. We present an algorithm for reconstructing viral haplotypes in a population from paired-end Next Generation Sequencing (NGS) data. We propose a novel polynomial time dynamic programming based approximation algorithm for generating top paths through each node in a De Bruijn graph constructed from the paired-end NGS data. We also propose two novel formulations for obtaining an optimal set of viral haplotypes for the population using the paths generated by the approximation algorithm. The first formulation obtains a maximum likelihood estimate of the viral population given the observed paired-end reads. The second formulation obtains a minimal set of viral haplotypes retaining the phylogenetic information in the population. We evaluate our algorithm on simulated datasets varying on mutation rates and genome length of the viral haplotypes. The results of our method are compared to other methods for viral haplotype estimation. While all the methods overestimate the number of viral haplotypes in a population, the two proposed optimality formulations correctly estimate the exact sequence of all the haplotypes in most datasets, and recover the overall diversity of the population in all datasets. The haplotypes recovered from popular methods are biased toward the reference sequence used for mapping of reads, while the proposed formulations are reference-free and retain the overall diversity in the population.

1 INTRODUCTION

At any given time, the viral population present within a host consists of a collection of distinct, albeit closely related genetic variants, known as viral haplotypes. The high genetic diversity of a virus population has important consequences in disease progression as it allows the virus to evade host defenses and confounds preventative and therapeutic interventions. An important task while studying viral populations is to identify the number of viral haplotypes and their individual sequences present in a viral population.

NGS technologies have revolutionized the field of genomics and opened up an array of possibilities for characterizing genetic diversity in viral populations using a large number of short DNA sequences (called reads) sampled from the population. The challenge of reconstructing viral genomes stems from the high genetic diversity of the population. The genetic variability of these haplotypes is due to the high rate of mutations, resulting in insertions, deletions and substitutions within a genome and recombination between viral haplotypes [5][6].

Several algorithms have been developed to reconstruct viral haplotypes by aligning reads to a reference genome [4]. A reference genome can be helpful when its sequence is highly similar to the haplotypes. However, due to the presence of recombination and high mutation rates in some viral populations (e.g. RNA viruses), a large percentage of the reads are unaligned to the reference genome, and are ignored while estimating viral diversity. Moreover, the existing algorithms have been known for predicting a large number of false-positive haplotypes in a population [23].

De novo approaches for assembling viral haplotypes provide an alternative to reference-based haplotype estimation. These approaches estimate the viral haplotypes based on the reads itself and assemble partial or full length genomes of the viral haplotypes. Also, with the availability of paired-end sequencing data, there is need for algorithms that can incorporate such data.

We propose a de novo assembly algorithm for viral haplotypes using paired-end sequencing data as an alternative to reference-based viral haplotype estimation. The main contributions of the paper are (i) a novel polynomial time dynamic programming based approximation algorithm for recovering top $L$--paths through every node in a De Bruijn graph generated from the paired-end sequencing reads, (ii) a maximum likelihood estimate of the viral population based on a generative model for sequencing paired reads from the viral population, and (iii) a phylogenetic tree based backward elimination algorithm that retains a minimal set of haplotypes that retains the phylogenetic information present in the viral population.

*Correspondence to be addressed to raunaq@psu.edu
The paper is organized as follows: We discuss the methods available in the literature in Section 2. Section 3 describes the proposed algorithm. Briefly, the reads obtained from viral population are processed as follows: a De Bruijn graph is constructed from the sampled reads. Paired-end reads are used to generate a set of paired relations amongst the nodes in the De Bruijn graph (Subsection 3.2). Paths in the graph from a source generate a set of paired relations amongst the nodes in the De Bruijn graph constructed from the sampled reads. Paired-end reads are used to population are processed as follows: a De Bruijn graph is the proposed algorithm. Briefly, the reads obtained from viral available in the literature in Section 2. Section 3 describes Malhotra et al. The second formulation defines a phylogenetic tree-based metric to obtain a minimal set of paths that retains the phylogenetic diversity in the viral population (Subsection 3.6). The top \(L\)-paths through the source nodes in the graph are utilized for estimating a minimal set of haplotypes constituting the viral population. The first formulation for obtaining viral haplotypes in the population constitutes defining a generative model for the sequenced reads from the population. We estimate a maximum likelihood set of paths representing the viral population using backward elimination algorithm (Subsection 3.5).

The second formulation defines a phylogenetic tree-based metric to obtain a minimal set of paths that retains the phylogenetic diversity in the viral population (Subsection 3.6). Section 4 details the results on simulated data of varying complexity and sequence lengths. We conclude the paper with discussion and future directions in section 5.

2 RELATED WORK

A number of methods have been reported for haplotype reconstruction that are able to infer genomes of individual haplotypes as well as their abundance \([2, 13, 30, 15]\). A survey of viral haplotype estimation methods can be found in \([4]\). The frequency of individual haplotypes is computed using an expectation-maximization (EM) algorithm \([22, 7, 35, 31, 36]\). Haplotype estimation can be performed locally along segments of the viral genome or globally across the whole genome. The local haplotype estimation is based on first aligning the reads to a reference genome and then estimating the number of haplotypes \([13, 22, 35]\). Probabilistic methods for estimating the haplotypes have been explored in \([30, 23, 35, 36]\).

The global estimation of the haplotypes is based on a graph theoretic solution, wherein a set of haplotypes are obtained by calculating a minimal set of paths that covers all the nodes in a graph of aligned reads \([7, 35]\). The problem is formulated as a network flow problem or as a read graph from which the set of haplotypes is estimated \([31, 27]\). However, it is difficult to infer haplotypes over lengths greater than the read length \([34]\) due to the increasing number of possible paths in the read graph. HaploClique is another based on enumeration of maximal cliques of reads present in the population \([28]\). It has been shown to recover full length haplotypes from the viral population. Global estimation of haplotypes can also be based on probabilistic mixture models.

PredictHaplo is a probabilistic approach wherein an infinite mixture model is used as a generative model for the haplotypes \([21]\). There are methods that also incorporate recombination amongst viral haplotypes for viral population reconstruction. QuasiRecomb assumes a jumping hidden Markov model (HMM) for estimating the haplotypes under recombination \([29]\). However, it generates a large number of haplotypes present at low frequencies constituting the viral population \([29]\).

The above methods rely on the existence of a representative reference genome. Due to high mutation rates and recombinations in viral populations, the sequence of an assembled reference genome can be different than the viral haplotypes present in the given population. This leads to a large number of unaligned yet relevant reads being discarded from subsequent analysis in the above methods. In \([2]\), the reads are iteratively aligned to an updated consensus sequence to maximize the number of reads aligned. Such iterative realignment increases the number of reads aligned to the assembled reference sequence.

De novo methods for estimating the number of viral haplotypes and their relative frequencies in a given population are also known \([20, 17]\). The counts of small substrings of reads are used for identifying insertions and deletions amongst viral haplotypes. The viral haplotypes can be reconstructed using any of single genome assemblers for assembling the viral haplotype sequence.

An alternative to alignment based haplotype reconstruction is to perform de novo assembly of the reads \([32]\). Cortex uses colored De Bruijn graphs to assemble different variants from a population \([12]\) and can detect and genotype, simple and complex genetic variants in populations with relatively low diversity, such as human. However, its performance for high genetic diversity populations, such as viruses, is unknown \([32]\). With the availability of paired-end sequencing data, there is a need for de novo algorithms that can utilize such data, as they help in generating long length segments originating from a single haplotype. The paired-end version of minimal set of paths that cover a read graph has been shown to be a NP-hard problem \([23, 3]\).

3 METHODS

3.1 Definitions

A string of length \(L\) over the alphabet \(\Sigma = \{A, G, C, T\}\) is called a read \(R \in \Sigma^L\). A substring of a read \(R\) from position \(i\) to position \(j\) is denoted as \(R[i, j]\). The reverse complement of a read \(R\), denoted as \(\bar{R}\), is the read reversed and then its bases complemented. A string of length \(k\) is known as a \(k\)-mer. The reverse complement of a \(k\)-mer \(u\) is denoted as \(\bar{u}\).

A viral population \(H\) is a collection of viral haplotypes \(\{H_1, H_2, \ldots, H_N\}\), where each haplotype \(H_i\) is a string of length \(L_i\) defined over the alphabet \(\Sigma^{L_i}\). We assume that each of the viral haplotypes \(H_i\) do not have large repeats. In other words, for \(k \neq k'\), if \(H_i[k, k + r] = H_i[k', k' + r]\) then \(r < D_i\), where \(D_i\) is small.

A paired read \(\{R_f, R_e\}\), of lengths \(|R_f| = L_f, |R_e| = L_e\), are two reads sampled from the same viral haplotype \(H\) where one of the reads is the reverse complement of the sampled substring of \(H\), and the two reads are separated by some distance \(d\). An example of a paired read sampled from haplotype \(H\) can be \(R_f = H[i, i + L_f]\) and \(R_e = H[i + L_f + d, i + L_f + d + L_e]\).

Problem Definition: Viral population reconstruction using paired reads. Formally, our aim is to reconstruct a viral population \(H\) that represents a collection of paired reads \((\{R_{1f}, R_{1e}\}, \{R_{2f}, R_{2e}\}, \ldots, \{R_{nf}, R_{ne}\})\) sampled from the viral population. For the purpose of this paper, we assume that the reads have been error-corrected.

3.2 Pre-processing of reads: De Bruijn Graph Construction and Paired-Nodes constraints

In order to reconstruct the viral haplotypes in the population \(H\), the sampled reads from the population are first error corrected. The reads can be error-corrected using any of the error-correction algorithms \([18, 11, 14, 33]\). The collection of error-corrected reads, \(\{R_{1f}, R_{1e}\}, \{R_{2f}, R_{2e}\}, \ldots, \{R_{nf}, R_{ne}\}\), are then represented in a De Bruijn graph. Each of the reads \((R_{ij}, R_{kij})\) is broken into \(k\)-mers. The \(k\)-mers constitute nodes in the De
Braun, graph, while a directed edge is defined between two consecutive k-mers in a read. As one of the reads in the pair \((R_{ij}, R_{uv})\) is reverse-complemented, we add k-mers from both \(R_{ij}\) and \(R_{uv}\). The De Bruijn graph thus contains nodes and edges \((V, E)\), where \(V = \{(u, v) = R_{ij}[a, a+k-1] | i \in \{1, \ldots, n\}, j \in \{1, \ldots, |R_{ij}|\}, a \in \{(j, 0, k) | R_{ij}[a, a+k-1], v = R_{ij}[a, a+k-1] \}\}. The same holds for \(R_{ij}\).

An edge \((u, v)\) is known as an incoming edge for node \(v\) and as an outgoing edge for node \(u\). The value of \(k\) used for constructing the De Bruijn graph is greater than the size of repeats observed in the viral population \((k > D)\). As there are no repeats in the viral haplotypes greater than \(D\) bases, choosing \(k > D\) would ensure that the graph \(G_v\) obtained is a directed acyclic graph.

The De Bruijn graph \(G(V, E)\) defined above can be reduced to an equivalent graph \(G_e\), wherein a chain of nodes in \(G\) with only one incoming and outgoing edge is reduced to single compressed node in \(G_e\). The edge set of \(G_e\) maintains the edge relationships observed in \(G\) of the first and last nodes in such a chain. We use to the compressed graph, \(G_e(V_e, E_e)\), in the rest of the text.

The nodes in the graph \(G_e\) with no incoming edges are known as source nodes, while nodes with no outgoing edges are known as sink nodes in the graph. If the graph \(G_e\) contains multiple source and sink nodes, two new nodes are added in the graph. One node known as a universal source node has directed edges from itself to each of the source nodes in the graph. The second node, known as a universal sink node has directed edges from all sink nodes to itself.

\[ \text{Paired Nodes constraints:} \]

The pairing information of the reads is also stored in the form of a paired-nodes set, which consists of all pairs of nodes that comprise a paired read. For example, if a paired read \((R_{ij}, R_{uv})\) is represented in the graph \(G_e\) as a set of nodes \((u_1, u_2, \ldots, u_f)\) and \((u_1, u_2, \ldots, u_g)\), then the paired-nodes set for the paired read is \(PS = \{(u_1, u_2, \ldots, u_f), (u_1, u_2, \ldots, u_g)\}\). As the paired read \((R_{ij}, R_{uv})\) is at most separated by the insert size length \(IS = L_f + L_v + d\), the read only contributes paired-nodes to the set \(PS\) that are separated by at most \(IS\). The set \(PS\) is computed separately and takes \(O(n \cdot |R|^2)\) time.

### 3.3 Estimating the viral haplotypes comprising the population

A set of paths from universal source node to universal sink node that collectively visits every node in the graph is known as a cover of the graph. The set of paths is denoted as \(P = \{P_1, P_2, \ldots, P_k\}\), where \(P_i = (u_1, u_2, \ldots, u_k, u_{sink})\) and \(\forall u \in V_e, \exists P_i \in P\) s.t. \(u \in P_i\). A path \(P\) is a minimal cover of the graph. Assuming that every genomic location of the viral haplotypes is sampled in the paired reads, the paths through the graph \(G_e\) from universal source node to universal sink node correspond to possible haplotypes in the graph, if they satisfy certain constraints. A sampled paired read is obtained from a single haplotype in the viral population and thus a path that contains the nodes constituting the paired read represents a possible haplotype. Such paths should constitute the cover of the graph. A cover of the graph that satisfies all such paired read constraints represents a set of haplotypes from the viral population.

The problem of finding a minimum cover of a graph with paired node constraints has been shown to be NP hard \([3, 4]\). We next present a polynomial time dynamic programming approximation algorithm for recovering the top set of paths per node satisfying the paired-node constraints in the set \(PS\). These paths are used as input for estimating a maximum likelihood estimate of the viral population (Subsection 3.5) and a minimal set of paths retaining the phylogenetic information in the viral population (Subsection 3.6).

### 3.4 Approximation algorithm for estimating top \(L\) paths per node

Given the graph \(G_e(V_e, E_e)\) and the paired-node set \(PS\), we compute top \(L\) paths \(\{P_{a1}, P_{a2}, \ldots, P_{aL}\}\) through each of the nodes \(u\) in \(V_e\) to the universal sink node. In particular, the paths from the source nodes in the graph correspond to possible haplotypes in the viral population.

A paired read sampled from a single viral haplotype imposes paired-node constraints on paths in the graph. In particular, the path corresponding to the viral haplotype representing the paired-read will satisfy paired-node constraints in the set \(PS\). Ideally, the path in the graph corresponding to the viral haplotype should satisfy all the paired-node constraints imposed by the reads from the viral haplotype.

We compute a score for each path based on the number of paired-node constraints that it satisfies. Consider a path \(P_u = (u, s_1, s_2, \ldots, u_{sink}, u_{sink})\) from node \(u \in V_e\) to universal sink node \(u_{sink}\) in the graph \(G_e\). The score for the path \(P_u\) is based on the number of node-pairs \((r, s)\) \(\in P_u\) that are within the insert size distance \(IS\) and are present in the paired-node set \(PS\). Also, we penalize the score of the path \(P_u\) for every node pair within the insert size distance that is absent in the paired-node set \(PS\).

\[
S(P_u) = \sum_{(r,s) \in P_u \cap distance(d(r,s) < IS)} 1 \left[ (r,s) \in PS \right] - p \cdot 1 \left[ (r,s) \not\in PS \right]
\]

Here, \(IS\) is the insert size of the paired reads, \(p\) is the penalty for absence of paired-node constraints within the insert size \(IS\), and \(d(u)\) denotes the depth of the first \(k\)-mer in the node \(u\) with respect to universal source node in the De Bruijn graph \(G\). As the graph \(G_e\) is a directed acyclic graph (DAG), implying the graph \(G\) is also a DAG, a relative ordering \(d(.)\) amongst the nodes can be computed in a time linear in the number of nodes and edges \(O(|V| + |E|)\) using a depth first search algorithm. The score can be further normalized based on the expected number of node-pairs for a path \(P_u\).

It should be noted that the score for a path \(P_u = (u, P_{su})\) can be expressed in terms of the score of its sub-path \(P_{su}\) as follows:

\[
S(P_u) = S(P_{su}) + \sum_{s \in P_{su} \cap distance(d(s) - d(u) < IS)} 1 \left[ (u,s) \in PS \right] - p \cdot 1 \left[ (u,s) \not\in PS \right]
\]

Thus, the score for a path \(P_u\) can be constructed using the scores of its sub-paths in a recursive fashion, similar to dynamic programming using memoization.

Algorithm describes the pseudo-code for computing the top \(L\) paths for every node in the graph. The algorithm starts by initializing the paths from all nodes \(u\) to the universal sink node \(u_{sink}\) to empty sets. The scores for all paths from each node are also initialized to empty sets (Lines 1-4). Next it iterates over all nodes \(u \in V_e\) in decreasing order of their depth \(d(u)\) and computes the top \(L\) paths for each node Paths\((u)\) using the function TOP-L-PATHS-FOR-NODE\((u, V_e, PS)\) (Lines 5-10).

Algorithm describes the memoized algorithm that computes the top \(L\) paths from a node \(u\) to the universal sink node \(u_{sink}\). It first recovers the top \(L\) paths from all the neighbors of \(u\) having an incoming edge from \(u\) into the array \(TP\) and their scores in \(SP\) (Lines 1-6). In other words, \(TP = \bigcup_u Paths(u); \forall s.t. (u, s) \in E_e\) and \(SP = \bigcup_u Score(s); \forall s.t. (u, s) \in E_e\). As the depth of the node \(u\) is smaller than the depth of nodes \(s\), the array \(Paths(s)\) and \(Score(s)\) have already been updated (Algorithm in line 9). It then updates the score for each of the path in \(TP\) when adding the node \(u\) to the path and ranks them based on their score using equation. (Lines 7-20). The penalty term is proportional to the length of the path \(T\) (Line 16). The paths stored in the array \(TP\) are sorted based on their updated scores (Line 21). The first \(L\) paths in the array \(TP\) and their scores \(SP\) are stored as the paths through node \(u\) to the universal sink node \(u_{sink}\) (Lines 22-26).

As at each step only the top \(L\) paths are returned to Algorithm the algorithm computes an approximate search in the space of all paths. The scoring mechanism ensures that the true haplotypes are propagated along
Algorithm 1 COMPUTE-TOP-L-PATHS() : Top L paths per node based on paired-node constraints for each node in the graph $G_c$
Input: Condensed Directed De Bruijn graph $G_c(V_c, E_c)$, Paired-nodes constraint set $PS, D(.)$ the depth of every node in the graph.
Output: Paths($u$) = $\{P_1, P_2, \ldots , P_{uL}\} \forall u \in V_c$
Score($u$) = $\{S_1, S_2, \ldots , S_{uL}\} \forall u \in V_c$
\begin{enumerate}
1. for each node $u \in V_c$ do
2. \hspace{0.5cm} Paths($u$) = $[\emptyset]$ //Initialize the paths for each node
3. \hspace{0.5cm} Score($u$) = $[\emptyset]$
4. \hspace{0.5cm} end for
5. Score($usink$) = $[\emptyset]$
6. Paths($usink$) = $[\emptyset]$ // Place holder between nodes
7. $N =$ SORT-DECREASING($V_c, D(.)$) // Sort nodes based on their depth in decreasing order
8. for $i = 1, \ldots , |N|$ do
9. \hspace{0.5cm} Paths($N[i], Score(N[i])) =$TOP-L-PATHS-FOR-NODE($N[i], E_c, PS$)
10. end for
end for

Algorithm 2 TOP-L-PATHS-FOR-NODE ($u, E_c, PS$) : Top L paths for a node u in the graph $G_c$
Input: Node $u$, Edge set $E_c$, Paired-edge constraints $PS$. Insert size IS
Output: Paths($u$), Score($u$), Set of top L-paths for the node u, and their respective scores
1. $TP = [\emptyset]$
2. $SP = [\emptyset]$
3. for each $(v, \{u, v\} \in E)$ do
4. \hspace{0.5cm} $TP =$ JOIN ($TP, Paths(v)$) // Obtain top L-paths of the neighbor
5. \hspace{0.5cm} $SP =$ JOIN ($SP, Score(v)$)
6. end for
7. for each path $T \in TP$ do
8. \hspace{0.5cm} $l =$ length($T$)
9. \hspace{0.5cm} $SP(T) =$ $SP(T)$ · $\frac{l(l-1)}{2}$
10. \hspace{0.5cm} for each node $w \in T$ do
11. \hspace{0.5cm} \hspace{0.5cm} if $(u, w) \in PS$ then
12. \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} $SP(T) =$ $SP(T)$ + 1
13. \hspace{0.5cm} \hspace{0.5cm} else if $D(w) - D(u) > IS$ then
14. \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} $SP(T) =$ $SP(T)$ + 1
15. \hspace{0.5cm} \hspace{0.5cm} else
16. \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} $SP(T) =$ $SP(T) - l$
17. \hspace{0.5cm} \hspace{0.5cm} end if
18. \hspace{0.5cm} end for
19. \hspace{0.5cm} $SP(T) =$ $SP(T)$ · $\frac{2}{|T| + 1}$
20. end for
21. ($TP, SP$) = SORT-DECREASING($TP, SP(.)$) // Sort TP paths based on the corresponding scores SP of the paths
22. for $i = 1 \ldots L$ do
23. \hspace{0.5cm} Paths($u$) = JOIN ($\{u\}''TP[i]'', Paths($u$)$) // Add u to the path
24. \hspace{0.5cm} Score($u$) = JOIN ($\{SP[i]$$, Score(u)$
25. end for
26. Return Paths($u$), Score($u$)

\[ P[R|H] = \frac{q}{P \cdot (G - L + 1)} = z_R \]

In this equation, $q$ is the number of haplotypes in $H$ that have $R$ as their sub-string, $P$ is the number of haplotypes in the viral population, and $G$ is the average length of the viral haplotypes, and $L$ is the average length of the viral haplotypes. For a paired-read ($R_f, R_c$), equation \[ is the same. Here, $q$ denotes the number of haplotypes in $H$ that have the paired read ($R_f, R_c$) as their sub-string and $L$ is the distance between the two pairs, $R_f$ and $R_c$, in the haplotypes. If a paired read is shared amongst two or more haplotypes, then it is possible that the distance between two read pairs is not same across the haplotypes. In such cases, $L$ denotes the shortest distance between all the haplotypes in $H$. It should be noted that Equation \[ also holds if we consider $R_i$ a k-mer or a paired-node. Also, the distance between two paired nodes can be obtained using the depth $D(.)$ function used in Algorithm 2.

Thus, for a collection of paired reads $\{(R_{f1}, R_{c1}), (R_{f2}, R_{c2}), \ldots , (R_{fn}, R_{cn})\}$, assuming independent sampling, the joint probability of observing the paired reads given the viral population $H$ can be expressed as:

\[ P((R_{f1}, R_{c1}), (R_{f2}, R_{c2}), \ldots , (R_{fn}, R_{cn}))|H) = \prod_{i=1}^{n} \frac{q_i}{P \cdot (G - L_i + 1)} = \prod_{i=1}^{n} z_{R_i} \]

where we denote the probability term as $z_{R_i}$, and other terms are as described above for paired reads. If we denote the number of times a read ($R_{f_i}, R_{c_i}$) is sampled as $c_i$, assuming independent sampling of each genomic location, we can express the probability of the paired reads as a multinomial expression.
5 can be simplified by assuming that reads sampled from each genomic location are independent of each other, thus, breaking the multinomial expression into a position by position binomial terms:

\[
P(\{c(R_1, R_\ell) = c_1, \ldots, c(R_n, R_\mu) = c_n\} | \mathbf{H}) = \frac{M!}{c_1! \cdot c_2! \cdot \cdots \cdot c_n!} \prod_{i=1}^{n} z_{R_i}^{c_i} \cdot (1 - z_{R_i})^{M-c_i}
\]

where, \( M = \sum_{i=1}^{n} c_i = n \). The multinomial expression in equation 5 can be simplified by assuming that reads sampled from each genomic location are independent of each other, thus, breaking the multinomial expression into a position by position binomial terms:

\[
P(\{c(R_1, R_\ell) = c_1, \ldots, c(R_n, R_\mu) = c_n\} | \mathbf{H}) = \frac{M!}{c_1! \cdot c_2! \cdot \cdots \cdot c_n!} \prod_{i=1}^{n} z_{R_i}^{c_i} \cdot (1 - z_{R_i})^{M-c_i}
\]

Now, we can estimate a maximum likelihood set of haplotypes \( \mathbf{H}_{\text{ml}} \) using equation 6 as follows:

\[
\mathbf{H}_{\text{ml}} = \max_{\mathbf{H}} P(\{c(R_1, R_\ell) = c_1, \ldots, c(R_n, R_\mu) = c_n\} | \mathbf{H})
\]

Here the set of all possible haplotypes can be arbitrarily large, making such a computation tractable. We use the top \( L \)–paths through all the source nodes as candidates for possible haplotypes for computing the maximum likelihood set of haplotypes. The maximum likelihood set of haplotypes for a given dataset is computed using backward elimination. We start the likelihood computation with all the top \( L \)–paths through the source nodes and iteratively remove one path from the set of all paths until the likelihood of the remaining paths in the set starts to decrease. The remaining set of haplotypes constitute a maximum likelihood estimate of the viral population.

### 3.6 Minimal set of haplotypes based on phylogenetic information

The diversity of haplotypes present in a viral population \( \mathbf{H} \) can be measured in terms of the phylogenetic information contained in its haplotypes. There are a number of measures for capturing phylogenetic information [16, 9] (See [10] for survey). We use phylogenetic diversity (PD) as the measure of phylogenetic information present in the population [9]. We construct a phylogenetic tree \( T \) from the haplotypes in the population \( \mathbf{H} \) using the relaxed neighbor-joining algorithm [25]. We use a grammar based distance metric for computing the pairwise distances between the haplotypes [24].

Given a phylogenetic tree \( T \) the total tree length is defined as:

\[
\text{TREE-LENGTH}(T) = \sum_{b \in T} l(b)
\]

where \( l(b) \) denotes the length of branch \( b \) in the tree. The presence of two or more haplotypes with very small branch length provides very little phylogenetic diversity and thus only a single haplotype from such group can capture the phylogenetic diversity for the population.

Keeping this in mind, we estimate a minimal set of haplotypes representing the viral population using a phylogenetic tree length based backward elimination algorithm (Algorithm 3). The algorithm uses the top \( L \)–paths recovered from all source nodes in the graph to generate a bootstrap phylogenetic tree. The phylogenetic diversity is measured as the total tree length. \( T_{\text{baseline}} \) of the above tree. The observed standard deviation, \( T_{\text{stddev}} \), amongst the bootstrap samples is used to design a stopping criteria.

Algorithm 3 initializes \( C \) to all the paths (Lines 1–4). Next, the algorithm iteratively removes a path \( c \) from the current set \( C \) and computes the total tree length \( T_{c} \) of the remaining paths (Line 8). The path \( \text{Rem} \) which has the minimum divergence in total tree length from the baseline tree total length \( T_{\text{baseline}} \) is removed. This ensures that the path removed contains the minimum phylogenetic information with respect to the remaining set of haplotypes \( T_{\text{Rem}} \). In other words, the current set of paths \( C = \{C \setminus \text{Rem}\} \) still maintains the phylogenetic information from the viral population. This process is iteratively continued till the total tree length of the remaining paths diverges more than 2 standard deviations from \( T_{\text{stddev}} \) (Lines 5–17). The remaining set of haplotypes is the minimal set of haplotypes that captures the phylogenetic signal in the population.

#### Algorithm 3 Backward elimination based on phylogenetic trees

**Input:** TopS : Set of top paths from all the source nodes in the graph from Algorithm 1

**Output:** \( \mathbf{H}_{\text{phy}} \) : Minimal set of haplotypes based on Phylogenetic Tree’s total tree length.

1. \((T_{\text{baseline}}, T_{\text{stddev}}) = \text{TREE-LENGTH}(\text{TopS}, 100) \)
2. Compute tree-length using 100 bootstrap runs over the TopS set of haplotypes and their stddev
3. \( C = \text{TopS} \setminus \emptyset \)
4. \( \text{Rem} = \emptyset \)
5. while \( T_{\text{change}} < 2 \cdot T_{\text{stddev}} \) do
6. \( M_{\text{cur}} = 0; \ T_{\text{min}} = -1e100 \)
7. for \( c \in C \) do
8. \( T_{\text{c}} = \text{TREE-LENGTH}(C \setminus c, 1) \)
9. if \( M_{\text{cur}} = |T_{\text{c}} - T_{\text{baseline}}| \)
10. \( T_{\text{min}} = M_{\text{cur}} \)
11. \( \text{Rem} = c \)
12. end if
13. end for
14. \( C = C \setminus \text{Rem} \)
15. \( T_{\text{change}} = T_{\text{min}} \)
16. end while
17. Return: \( C \)

#### 3.7 Simulated Data

The simulated datasets are generated using Bayesian Serial SimCoal Simulator (BayeSSC) [8, 1]. The simulator takes population parameters such as population size, mutation rates, and genome length as input and generates a hypothetical tree and corresponding set of haplotypes. We then use the set of haplotypes as the population from which we simulate error-free Illumina sequencing paired-reads using dwgsim (https://github.com/nh13/dwgsim). These reads are used as input to evaluate our proposed algorithm.

We generated 10 random populations containing seven haplotypes each (D1-D10, Table 1). We then simulated 7000 paired-reads with average read length of 150 bps. We also simulated populations containing viral haplotypes with longer length (4500 bp), at a higher mutation rate and varying number of haplotypes in the population (Datasets D11-D14, Table 1). We also simulate 100000 Illumina sequencing reads with average read length of 150 bps for datasets D11-D14. The mutation rates and population sizes chosen for simulation are those that are typically observed in viral populations.

#### 4 RESULTS AND DISCUSSION

We first evaluate Algorithm 1 on the simulated datasets (D1-D10). Each of the datasets contains 7 haplotypes of length 1200 base pairs. We used Algorithm 1 with \( L = 10 \) to generate top 10-paths per node from the De Bruijn graph. The penalty term \( p \) is set to 10 for all nodes in the graph. The number of paths generated by Algorithm 1 is a fraction of the total number of paths present in the De Bruijn
Table 1. Simulated datasets with varying # of haplotypes in a population

| Dataset | Mutation Rate, Population Size | Genome Length | # of haplotypes in population |
|---------|---------------------------------|---------------|-------------------------------|
| D1-D10  | $10^{-3}$, 5000                | 1200          | 7                            |
| D11     | $2 \cdot 10^{-6}$, 5000        | 4500          | 10                           |
| D12     | $2 \cdot 10^{-6}$, 5000        | 4500          | 7                            |
| D13     | $2 \cdot 10^{-6}$, 5000        | 4500          | 8                            |
| D14     | $2 \cdot 10^{-6}$, 5000        | 4500          | 13                           |

We next compare the results on datasets D1-D10 for the two formulations of backward elimination (BE), namely the maximum likelihood estimate, and the phylogenetic tree based BE. The phylogenetic tree based BE generates more haplotypes at convergence as compared to the maximum likelihood based method (Table 3). The maximum likelihood estimate completely recovers the true haplotypes with an exact sequence match for datasets D1, D5 and D9, while it recovers more than 5 haplotypes with exact sequence match in all ten datasets (numbers in brackets, Column 2 of Table 3). The phylogenetic tree based BE, however, recovers 0 – 4 out of seven haplotypes with exact sequence match in the ten datasets.

We investigate whether the haplotypes predicted by phylogenetic tree based BE capture the phylogenetic information present in the true set of haplotypes. We generate bootstrap neighbor-joining trees using the predicted haplotypes and the true set of haplotypes. Overall, the phylogenetic tree based BE recovers the phylogenetic information present in the viral population (Figure 1). Although the phylogenetic tree based BE recovers only 0-2 true haplotypes with an exact sequence match (numbers in bracket in Table 3) for datasets D1-D3, D5, D8, and D9, we can see the phylogenetic information of at least six haplotypes present in these populations are still recovered in the tree (Figure 1). In dataset D2, there are no predicted haplotypes for haplotype 5, while in dataset D3, there is no predicted haplotype corresponding to haplotype 6. Also for datasets D5 and D9, one of the true haplotypes is not recovered by the phylogenetic tree based method (haplotypes 4 and 1 respectively).

We observe similar trends for other datasets (D4, D6-D7, D10).

Datasets D11-D14 denote populations containing haplotypes of length 4500 bps and different number of unique haplotypes in the population (7-13). Algorithm 1 with $L = 35$ retains at most 0.5% of the total paths in graph (Table 2), significantly reducing our search space for the haplotypes in the viral population. The exact sequences of the true haplotypes are retained for all datasets by Algorithm 1 except for dataset D14, where the exact sequences of two haplotypes were not recovered.

We also compare our results to those obtained from the software ShoRAH [33], QuasiRecomb [29], and PredictHaplo [21]. Another tool for generating full length viral haplotypes HaploClique [28] was not evaluated as it is not supported. The number of paths predicted by the maximum likelihood formulation is closest to

Table 2. Comparison of total number of paths to paths generated in top $L$-paths

| Dataset | Total paths in the graph | Top $L$- paths algorithm $^a$ | # of True haplotypes present (# haplotypes in population) $^b$ |
|---------|--------------------------|-------------------------------|-------------------------------------------------------------|
| D1      | 4824                     | 69                            | 7(7)                                                        |
| D2      | 44299                    | 46                            | 7(7)                                                        |
| D3      | 325585                   | 51                            | 7(7)                                                        |
| D4      | 164387                   | 69                            | 7(7)                                                        |
| D5      | 6768                     | 86                            | 7(7)                                                        |
| D6      | 1665626                  | 42                            | 7(7)                                                        |
| D7      | 2423                     | 78                            | 7(7)                                                        |
| D8      | 8712                     | 96                            | 7(7)                                                        |
| D9      | 4895                     | 91                            | 7(7)                                                        |
| D10     | 1357                     | 100                           | 7(7)                                                        |
| D11     | 35389440                 | 661                           | 10(10)                                                      |
| D12     | 30720                    | 345                           | 7(7)                                                        |
| D13     | 995328                   | 416                           | 8(8)                                                        |
| D14     | 35389444                 | 517                           | 11(13)                                                      |

$^a$ Total # of paths obtained from the graph using Top $L$- paths algorithm (Algorithm 1), $L=10$ for D1-D10 and $L=35$ for D11-D14.

$^b$ The # of true haplotypes in the population that are retained by Algorithm 1. Number in bracket indicates total number of true haplotypes in the population.

Fig. 1. Neighbor-joining trees condensed at $>75\%$ generated from haplotypes predicted Phylogenetic tree based BE and the true haplotypes. True haplotypes: red colored circles, Predicted haplotypes: light blue colored squares. For most of the red circles, there are light blue colored squares in the same cluster.

We also compare our results to those obtained from the software ShoRAH [33], QuasiRecomb [29], and PredictHaplo [21]. Another tool for generating full length viral haplotypes HaploClique [28] was not evaluated as it is not supported. The number of paths predicted by the maximum likelihood formulation is closest to
the true number of haplotypes in all the datasets. The number of predicted haplotypes by ShoRAH range from 1 to 128 (Table 3). ShoRAH over-estimates the number of haplotypes present in the datasets in most cases and is not able to recover the correct number of haplotypes for any dataset. It is able to retain four of the true haplotypes in dataset D8, and retains less than two of the true haplotypes for eight out of ten simulated datasets. QuasiRecomb also over-estimates the number of haplotypes present in a population, although a large number of haplotypes are reported as low frequency variants (relative frequency in the population is less than $5 \cdot E^{-4}$, Table 3). PredictHaplo under estimates the number of haplotypes in the population. It recovers at least one correct in all datasets (except for D1 and D10).

We also investigate the phylogenetic relationships of the predicted haplotypes from all algorithms to the true set of haplotypes used for simulation. For dataset D2, we generate a bootstrap neighbor-joining phylogenetic tree using the true haplotypes and the predicted haplotypes from ShoRAH, QuasiRecomb, PredictHaplo, our maximum likelihood estimate, and the phylogenetic tree based BE method (Figure 2). The maximum likelihood formulation predicts haplotypes that cluster with each of the true haplotypes except haplotype 3 (Green colored squares in Figure 2). The phylogenetic tree based BE predicts a cluster of haplotypes representative of all seven haplotypes (light blue squares in Figure 2), while all other algorithms predict haplotypes related to haplotype 3. It is interesting to note that all three algorithms (ShoRAH, QuasiRecomb, and PredictHaplo) used haplotype 3 as the reference for mapping all the reads. As can be seen, all the predicted haplotypes from these methods are similar to haplotype 3. This shows that it is difficult to capture the phylogenetic information of the population from the reference-based methods used above, while the phylogenetic tree based BE method effectively captures it. Similar trends are observed on other datasets.

We next examine the likelihood landscape for the generative model based on the paired reads. In general, the maximum likelihood of observed paired-read counts given a set of haplotypes increases as the cardinality of the set of haplotypes decreases (Figure 3), suggesting that the datasets can be explained by a small set of haplotypes. For datasets D1 (Figure 3(a)) and D10 (Figure 3(d)), maximum likelihood is obtained for a set larger than 7 haplotypes (total number of true haplotypes in the dataset), while the rest of the datasets follow trends similar to datasets (Figures 3(b) and 3(c)). As no sets of less than six haplotypes can explain all the observed paired-node counts, we terminate the backward elimination algorithm below it. In five out of the ten datasets, the true haplotypes that were used for simulation are retained in the final maximum likelihood set of haplotypes, while for the remaining datasets, six out of seven true haplotypes are retained at convergence.

For datasets D11-D14, Algorithm I generates both full length and partial haplotypes. Even though the regions with low coverage breaks the assembly of full length haplotypes, Algorithm I retains the partial paths. As the paired relations set is violated for partial paths, it is not possible to recover a maximum likelihood estimate for datasets D11-D14.

The phylogenetic tree based BE method, though, can operate on the partial haplotypes, and generates a set of viral haplotypes representing the viral population. The phylogenetic tree based BE method produces the best set of results when compared to QuasiRecomb on for D11-D14 (Table 4). ShoRAH crashed repeatedly while running on these datasets. The number of haplotypes predicted by these algorithms is larger than the actual number of haplotypes in the dataset. PredictHaplo predicts one or two haplotypes for these datasets. PredictHaplo only predicts a single correct haplotype for dataset D12. Our algorithm generates the smallest set of haplotypes and also recovers the maximum number of true haplotypes from the population.

**Table 4. D11-D14: Comparison of number of haplotypes generated by Algorithm I Tree based Backward Elimination, and QuasiRecomb**

| Data set  | # of True Haplotypes | Phylogenetic tree based BE a | Quasi Recomb a | Quasi Recomb $\alpha > 0.0005$ a | Predict Haplo a |
|-----------|----------------------|-----------------------------|---------------|-----------------------------|----------------|
| D11       | 10                   | 233(5)                      | 8339(2)       | 81 (0)                     | 2(0)           |
| D12       | 7                    | 116(5)                      | 3324(3)       | 638(2)                     | 1(1)           |
| D13       | 8                    | 157(5)                      | 6328(3)       | 252 (2)                    | 2(0)           |
| D14       | 13                   | 176(10)                     | 1471(6)       | 249 (5)                    | 1(0)           |

a The # in brackets indicate the number of true haplotypes that are present amongst these haplotypes.
Table 3. Comparison of number of predicted paths by Algorithm 1, ShoRAH, and QuasiRecomb

| Dataset | Maximum likelihood BE | Phylogenetic tree based BE | ShoRAH | QuasiRecomb | QuasiRecomb $\alpha > 0.0005$ | PredictHaplo |
|---------|-----------------------|---------------------------|--------|-------------|-------------------------------|-------------|
| D1      | 7(7)                  | 10(1)                     | 61(0)  | 6063(3)     | 237(2)                       | 2(0)        |
| D2      | 7(5)                  | 9(2)                      | 8(0)   | 118(1)      | 76(1)                        | 1(1)        |
| D3      | 6(6)                  | 5(0)                      | 47(1)  | 225(1)      | 218(1)                       | 1(1)        |
| D4      | 6(6)                  | 19(3)                     | 18(2)  | 6176(2)     | 127(2)                       | 4(2)        |
| D5      | 7(7)                  | 11(0)                     | 128(1) | 4078(2)     | 420(1)                       | 1(1)        |
| D6      | 10(6)                 | 6(3)                      | 1(1)   | 4(1)        | 4(1)                         | 1(1)        |
| D7      | 6(6)                  | 23(4)                     | 67(0)  | 9999(0)     | 0(0)                         | 1(1)        |
| D8      | 7(5)                  | 10(1)                     | 26(4)  | 7093(2)     | 106(0)                       | 2(1)        |
| D9      | 7(7)                  | 19(0)                     | 49(0)  | 4846(3)     | 340(3)                       | 3(1)        |
| D10     | 11(7)                 | 31(2)                     | 8(1)   | 9996(0)     | 0(0)                         | 2(0)        |

*a The # in the bracket indicates the # of true haplotypes that are present in the predicted set with an exact match (See section 3.5)

*b The column lists the # of paths from QuasiRecomb that have an abundance in the population greater than 0.0005.

Fig. 3. Maximum Likelihood estimation using backward elimination. X-axis is the number of haplotypes present in the predicted set, Y-axis shows the maximum likelihood for any combination of predicted haplotypes of the size mentioned on x-axis.

We investigate whether the phylogenetic tree based BE recovers the phylogenetic signal in dataset D14. For this dataset, there were two haplotypes which were not recovered by Algorithm 1. We generate a bootstrap neighbor-joining tree using the 176 haplotypes predicted by phylogenetic tree based BE, the top 146 haplotypes predicted from QuasiRecomb, 13 true haplotypes, and one haplotype from PredictHaplo (Figure 4). The phylogenetic tree based BE predicts haplotypes close to all of the true haplotypes including the two haplotypes that were not captured by Algorithm 1. There is very low bootstrap support for the predicted haplotypes that form separate groups on the phylogenetic tree. The haplotypes predicted by QuasiRecomb cluster with few of the true haplotypes, but fail to capture haplotypes close to two of the true haplotypes.

5 CONCLUSIONS

We propose a polynomial time dynamic programming based approximation algorithm that generates a possible set of haplotypes constituting a viral population using paired-end sequencing data. We also propose two novel formulations for estimating the haplotypes present in a population: (i) backward elimination based on a phylogenetic tree total tree length and (ii) a maximum likelihood estimate of the viral haplotypes based on paired reads. We show that the approximation algorithm (Algorithm 1) is robust and retains true set of haplotypes on simulated datasets. Algorithm 1 generates both full length and partial haplotypes. As regions with low coverage would break the assembly of haplotypes, the algorithm retains the partial paths. The phylogenetic tree based BE formulation can operate on the partial paths to generate a set of viral haplotypes representing the viral population. The two proposed formulations can recover haplotypes that similar to the set of true haplotypes and are not biased by a reference sequence. The phylogenetic based formulation predicts a larger number of haplotypes compared to the true set of haplotypes, nevertheless, the phylogenetic information is retained in the predicted set of haplotypes.

Funding: This work is partly funded by National Evolutionary Synthesis Center (NESCent) and NSF award #1421908.

REFERENCES

[1] Christian NK Anderson, Uma Ramakrishnan, Yvonne L. Chan, and Elizabeth A. Hadly. Serial simcoal: a population genetics model for data from multiple populations and points in time. *Bioinformatics*, 21(8):1733–1734, 2005.

[2] Irina Astrovskaya, Bassam Tork, Serghei Mangul, Kelly Westbrooks, Ion Mândou, Peter Balfe, and Alex Zelikovsky. Inferring viral quasispecies spectra from 454 pyrosequencing reads. *BMC Bioinformatics*, 12(6), 2011.

[3] Niko Beerenwinkel, Stefano Beretta, Paola Bonizzoni, Riccardo Dondi, and Yuri Pirola. Covering pairs in directed acyclic graphs. In Adrian-Horia Dediu, Carlos Martín-Vide, Jos-Luis Sierra-Rodríguez, and Bianca Truthe, editors, *Language and Automata Theory and Applications*, volume 8370 of Lecture Notes in Computer Science, pp. 126–137. Springer International Publishing, 2014.
Neighbor-joining phylogenetic tree of the predicted haplotypes from D14 along with true haplotypes. The true haplotypes are highlighted as red circles. The two black circles are the haplotypes that were not captured by Algorithm 1. Other predicted haplotypes are as follows: Phylogenetic tree based BE: light blue squares, QuasiRecomb: Green squares, PredictHaplo: Black Square. The haplotypes from Phylogenetic tree based BE have clusters around each of the true haplotypes. Other predicted haplotypes are as follows: Phylogenetic tree based BE: light blue squares, QuasiRecomb: Green squares, PredictHaplo: Black Square. The haplotypes from Phylogenetic tree based BE have clusters around each of the true haplotypes.

[1] Yun Heo, Xiao-Long Wu, Deming Chen, Jian Ma, and Wen-Mei Hwu. Bless: Bloom filter-based error correction solution for high-throughput sequencing reads. *Bioinformatics*, 20(6):1–4, 2004.

[2] Mattia Prosperi, Luciano Prosperi, Alessandro Bruselles, Gabriella Rizzi, Donatella Vincenti, Maria Solmona, Maria Capobianchi, and Giovanni Ulivi. Combinatorial analysis and algorithms for quasispecies reconstruction using next-generation sequencing. *BMC Bioinformatics*, 12(5), 2011.

[3] Romeo Rizzi, Alexandre I Tomsic, and Veli Mäkinen. On the complexity of minimum path cover with subpath constraints for multi-assembly.

[4] Pavel Skums, Nicholas Mancuso, Alexander Arytymenok, Bassam Tork, Ion Mandoiu, Yury Khudyakov, and Alex Zelikovsky. Reconstruction of viral quasispecies structure from next-generation sequencing data using multicommodity flows. *BMC Bioinformatics*, 14(10), 2013.

[5] Mattia Prosperi, Tobias Marschall, Rowena A Bull, Fabio Luciani, Alexander Schönhuth, and Niko Beerenwinkel. Viral quasispecies assembly via maximal clique enumeration. *PLoS computational biology*, 10(3):e1003515, 2014.

[6] Mattia Prosperi, Sandhya Prabhakaran, Volker Roth, Eran Halperin, and Niko Beerenwinkel. Probabilistic inference of viral quasispecies subject to recombination. *Journal of Computational Biology*, 20(2):113–123, 2013.

[7] Chunlin Wang, Yumi Mitsuya, Baback Ghazradasht, Mostafa Ronaghi, and Niko Beerenwinkel. Viral population estimation using pyrosequencing. *PLoS Comput Biol*, 4(5):e1000074, 2008.

[8] Laurent Excoffier, J. Neville, and Stefan Schneider. Computer note. simcoal: a general coalescent program for the simulation of molecular data in interconnected populations with arbitrary demography. *Journal of Heredity*, 91(6):506–509, 2000.

[9] Daniel P. Faith. Conservation evaluation and phylogenetic diversity. *Biological Conservation*, 61(1):1–10, 1992.

[10] Matthew R Helmus, Thomas J Bland, Christopher K Williams, and Anthony R Ives. Phylogenetic measures of biodiversity. *The American Naturalist*, 169(3):E68–E83, 2007.

[11] Yun Heo, Xiao-Long Wu, and Wen-Mei Hwu. Bless: Bloom filter-based error correction solution for high-throughput sequencing reads. *Bioinformatics*, 20(6):1–4, 2004.

[12] Z. Iqbal, M. Caccamo, I. Turner, P. Flice, and G. McVean. De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nature Genetics*, 2012.

[13] Vladimir Jojic, Tomer Hertz, and Nebojsa Jojic. Population sequencing using short reads: HIV as a case study. In *Proc. Pac Symp Biocomput*, pp. 114–125, 2008.

[14] David R Kelley, Michael C Schatz, Steven L Salzberg, et al. Quake: quality-aware detection and correction of sequencing errors. *Genome Biol.*, 11(11):R116, 2010.

[15] Alexander R. Macalalad, Michael C. Zody, Patrick Charlebois, Niall J. Lennon, Ruchi M. Newman, Christine M. Malboeuf, Elizabeth M. Ryan, Christian L. Bouitwell, Karen A. Power, Doug E. Brackney, Kendra N. Pesko, Joshua Z. Levin, Gregory D. Ebel, Todd M. Allen, Bruce W. Birren, and Matthew R. Henn. Highly sensitive and specific detection of rare variants in mixed viral populations from massively parallel sequence data. *PLoS Comput Biol*, 8(3):e1002417, 2012.

[16] Anne E Magurran and Anne F Magurran. *Ecological diversity and its measurement*, volume 168. Springer, 1988.

[17] Raunag Malhotra, Shruthi Prabhakara, Mary Poss, and Raj Acharya. Estimating viral haplotypes in a population using k-mer counting. In *Pattern Recognition in Bioinformatics*, pp. 265–276. Springer, 2013.

[18] Pavel Medvedev, Eric Scott, Boyko Kakaradov, and Pavel A. Pevzner. Error correction of high-throughput sequencing datasets with non-uniform coverage. *Bioinformatics (ISMB/ECCB)*, 27(13):137–141, 2011.

[19] J. A. Pevzner, H. Tang, and M. S. Waterman. An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences*, 98:9748–9753, 2001.

[20] Shruthi Prabhakara, Raunag Malhotra, Raj Acharya, and Mary Poss. Mutant-bin: Unsupervised haplotype estimation of viral population diversity without reference genome. *Journal of Computational Biology*, 20(6):453–463, 2013.

[21] Z. Iqbal, M. Caccamo, I. Turner, P. Flicke, and G. McVean. De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nature Genetics*, 2012.

[22] Mattia Prosperi, Tobias Marschall, Rowena A Bull, Fabio Luciani, Alexander Schönhuth, and Niko Beerenwinkel. Viral quasispecies assembly via maximal clique enumeration. *PLoS computational biology*, 10(3):e1003515, 2014.

[23] Armin Töpfer, Tobias Marschall, Rowena A Bull, Fabio Luciani, Alexander Schönhuth, and Niko Beerenwinkel. Viral quasispecies assembly via maximal clique enumeration. *PLoS computational biology*, 10(3):e1003515, 2014.

[24] Armin Töpfer, Osvaldo Zagordi, Sandhya Prabhakaran, Volker Roth, Eran Halperin, and Niko Beerenwinkel. Probabilistic inference of viral quasispecies subject to recombination. *Journal of Computational Biology*, 20(2):113–123, 2013.

[25] Chunlin Wang, Yumi Mitsuya, Baback Ghazradasht, Mostafa Ronaghi, and Niko Beerenwinkel. Viral population estimation using pyrosequencing. *PLoS Comput Biol*, 4(5):e1000074, 2008.

[26] Laurent Excoffier, J. Neville, and Stefan Schneider. Computer note. simcoal: a general coalescent program for the simulation of molecular data in interconnected populations with arbitrary demography. *Journal of Heredity*, 91(6):506–509, 2000.

[27] Daniel P. Faith. Conservation evaluation and phylogenetic diversity. *Biological Conservation*, 61(1):1–10, 1992.

[28] Matthew R Helmus, Thomas J Bland, Christopher K Williams, and Anthony R Ives. Phylogenetic measures of biodiversity. *The American Naturalist*, 169(3):E68–E83, 2007.

[29] Yun Heo, Xiao-Long Wu, Deming Chen, Jian Ma, and Wen-Mei Hwu. Bless: Bloom filter-based error correction solution for high-throughput sequencing reads. *Bioinformatics*, 20(6):1–4, 2004.
[35] Osvaldo Zagordi, Lukas Geyrhofer, Volker Roth, and Niko Beerenwinkel. Deep sequencing of a genetically heterogeneous sample: local haplotype reconstruction and read error correction. *Journal of computational biology*, 17(3):417–428, March 2010.

[36] Osvaldo Zagordi, Rolf Klein, Martin Däumer, and Niko Beerenwinkel. Error correction of next-generation sequencing data and reliable estimation of HIV quasispecies. *Nucleic Acids Research*, 38:7400–7409, 2010.