

Supplementary

De novo haplotype reconstruction in viral quasispecies using paired-end read guided path finding - Supplementary

Jiao Chen¹, Yingchao Zhao² and Yanni Sun^{1,*}

¹Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA.

²School of Computing and Information Sciences, Caritas Institute of Higher Education, Hong Kong

1 Longest Common Substring (LCS) Distribution

1.1 Dynamic Programming for Calculating LCS Distribution

A number of existing haplotype reconstruction tools employ paired-end read information to distinguish different haplotypes. By estimating the size distribution of LCSs in viral quasispecies samples, we can evaluate whether paired-end reads are sufficient for haplotype reconstruction. Below we provide a probability model to estimate the LCS length distribution between a viral strain and its n th generation offspring.

Suppose initially there is only one virus strain x_0 of length L in the environment. Mistakes, such as insertions, deletions or substitutions, can happen each time the virus replicates. Finally there will be an equilibrium of multiple viral strains of constant abundances in the environment, which is described by the quasispecies theory. To simplify the LCS problem, we assume only substitutions can happen during replication and the mutation rate at each position is constant and independent.

Mutation rate accumulation for sequence replication Denote the mutation rate between two generations as μ , we ask what is mutation rate at each location between x_0 and its n th offspring x_n ?

This problem can be solved by dynamic programming. Define the subproblem $f[n]$ as the probability that $x_n[i]$ is different from $x_0[i]$, of which n denotes the n th generation and i is an arbitrary position on the genome. We have

$$f[n+1] = (1 - f[n])\mu + f[n](1 - \mu/3), \quad n = 1, 2, \dots \quad (1)$$

where $f[1] = \mu$. Equation 1 can be solved

$$f[n] = (1 - \frac{4\mu}{3})^{n-1}(\mu - \frac{3}{4}) + \frac{3}{4}, \quad n = 1, 2, \dots \quad (2)$$

Probability of LCS Let τ be the mutation rate at each position between x_0 and its n th generation offspring x_n , which can be calculated from equation 2. We can compute the distribution of LCSs between x_0 and x_n with dynamic programming. Let $x[1...i]$ be the prefix of x ending at position i . Define two sub-problems $f(i)$ and $g(i)$. $f(i)$ is the probability

that prefix $x_n[1...i]$ has $LCS \leq m$ with $x_0[1...i]$ and $x_n[i]$ mutated, $g(i)$ is the probability that prefix $x_n[1...i]$ has $LCS \leq m$ with $x_0[1...n]$ and $x_n[i]$ not mutated. We get the following recursive relationship:

$$\begin{aligned} m = 2 \\ f(i+1) &= \tau f(i) + \tau g(i) \\ g(i+1) &= (1 - \tau)f(i) + (1 - \tau)^2 f(i-1) \\ m = 3 \\ f(i+1) &= \tau f(i) + \tau g(i) \\ g(i+1) &= (1 - \tau)f(i) + (1 - \tau)^2 f(i-1) + (1 - \tau)^3 f(i-2) \\ &\vdots \\ m \\ f(i+1) &= \tau f(i) + \tau g(i) \\ g(i+1) &= \sum_{j=0}^{m-1} f(i-j)(1 - \tau)^{j+1} \end{aligned}$$

If $i \leq m$, the LCS will always be less or equal to m . Thus, $f(i) = \tau$, $g(i) = 1 - \tau$. Therefore, the recursive equations for calculating $f(i)$ and $g(i)$ are

When $1 \leq i \leq m$

$$\begin{cases} f(i) = \tau \\ g(i) = 1 - \tau \end{cases} \quad (3)$$

When $L \geq i \geq m+1$

$$\begin{cases} f(i) = \tau(f(i-1) + g(i-1)) \\ g(i) = \sum_{j=0}^{m-1} f(i-j-1)(1 - \tau)^{j+1} \end{cases} \quad (4)$$

The time complexity to calculate the probability for $LCS = m$ is $O(2L + mL)$. Since the sequence length is L , m can be any integers between 0 and L . To calculate the probability for each $m \in [0, L]$, the time complexity is $O(2L^2 + (L+1)\frac{L^2}{2}) = O(L^3)$. For HIV, its genome length is about 10^4 . Calculating directly with equations 4 for all the possible LCS is time consuming. It costs hours to calculate the probabilities for all the available LCS with a typical single-core CPU.

The time complexity can be reduced to $O(L^2)$ if we utilize the result of $g(i)$ to calculate for $g(i+1)$. Let $S(i) = f(i) + g(i)$, we will have

the recursive relationship (Proof omitted)

$$S(i+1) = S(i) - \tau(1-\tau)^{m+1}S(i-m-1), \quad i \geq m+2$$

Therefore, we get

$$S(i) = \begin{cases} 1, & \text{if } 1 \leq i \leq m \\ 1 - (1-\tau)^{m+1}, & \text{if } i = m+1 \\ 1 - (1+\tau)(1-\tau)^{m+1}, & \text{if } i = m+2 \\ S(i-1) - \tau(1-\tau)^{m+1}S(i-m-2), & \text{if } i \geq m+3 \end{cases}$$

It is linear to calculate the probability for one LCS. The time complexity for calculating $LCS = m$ is $O(L)$. To calculate each $m \in [0, L]$, the time complexity is $O(L^2)$.

1.2 Results of LCS Distribution

To estimate the LCS between two strains within a quasispecies, we calculated the probability distribution of LCS between two strains that are n generations apart based on Equation 4.

Let the mutation rate μ between two generations be $3e-5$, which is a commonly used average RNA viral mutation rate. The mutation rate between two haplotypes that are n generations apart can be calculated by equation 2. The genome size L is set as 10,000. The probability distribution for LCSs between two strains is shown in Figure S1. With the increase of replication cycles, the average LCS length keeps decreasing. According to the distribution, for haplotypes that are only 50 generations apart, the average LCS between them are very large and thus only paired-end sequencing with large insert size can be used to distinguish them. Figure S1 provides theoretical guidance about choosing appropriate insert sizes for given quasispecies data.

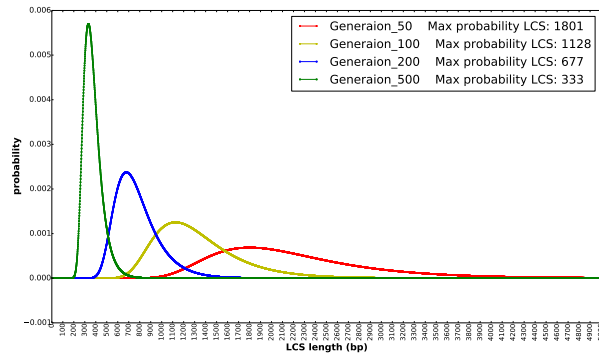


Fig. S1. Probability distribution for LCSs between two strains that are 50, 100, 200, and 500 generations apart. The x-axis is the length of LCS, with a range from 0 to 10,000. The y-axis is the corresponding probabilities for those LCS sizes.

2 Data Pre-processing

Error correction based on coverage information, usually improves *de novo* assembly. An alignment-based error correction tool Karect (Allam *et al.*, 2015) is used to correct substitution, insertion, and deletion errors. Besides adopting existing error correction tools, we also removed reads with very low abundance. Only reads that have at least n duplicates in the original data set were used for downstream analysis. Using a probability model, we show that this method can further filter out error containing reads while maintaining sufficient reads for *de novo* assembly.

Let N be the total reads number, r be the read length and L be the genome size. If we assume the reads are randomly sequenced from the

genome, the probability that k reads start from the same location on the genome can be estimated by a Poisson distribution:

$$Pr[k] = \frac{\lambda^k e^{-\lambda}}{k!}, \quad \lambda = N/L$$

$$Pr[k \geq n] = 1 - \sum_{i=0}^{n-1} \frac{\lambda^i e^{-\lambda}}{i!}, \quad n \geq 1$$

We denote μ as the probability of sequencing error at each base. For k reads originating from the same location on genome, the probability that they have one sequencing error on the same location is $Pr[N_\mu = 1] = r\mu^k$, the probability of two sequencing errors on the same locations is $Pr[N_\mu = 2] = \binom{r}{2}\mu^{2k}$, ..., the probability of at least one sequencing error on the same location can be calculated as $Pr = \sum_{i=1}^r \binom{r}{i}\mu^{ik} = (\mu^k + 1)^r - 1$. In the HIV MiSeq data set we used, $\sim 700k$ error corrected reads are left for 5 HIV strains. The genome length is $\sim 10k$ bp. $\lambda = 7 \times 10^5 / (5 \times 10^4) = 14$. For each base on the genome, the probability that at least 3 reads start from it is over 99.9%. If we assume $\mu = 0.01$, $r = 200$, the probability that a read sequence has at least 3 duplicates but contain one or more sequencing error is $Pr = (1e - 6 + 1)^{200} - 1 \approx 2.0e - 4$. The results reveal that when sequencing depth is deep enough, keeping reads with duplications is able to filter out error containing reads and will not reduce connectivity.

3 Reads orientation adjustment with overlap graph

Reads in the raw data set can come from both strands of the genome. The two reads of a read pair usually come from different strands. To better assemble the reads, we need to adjust the reads orientation so that they are from the same strand. We used Readjoinder (Gonnella and Kurtz, 2012) to construct the overlap graph and traverse each node of the graph for strand adjustment.

For two reads r_i and r_j , we denote r'_i and r'_j as their reverse complements. Readjoinder computes all possible overlaps between (r_i, r_j) , (r'_i, r_j) and (r_i, r'_j) , and label the overlaps with '+', '-', and '+' respectively if their suffix-prefix matches are longer or equal to the overlap threshold. A breadth-first search (BFS) traversal method is used for reads orientation adjustment. The traversal starts at a start node (with in-degree of 0) and label it as '+', then recursively labels all its successors and predecessors based on the edge type. The '+' type means current node and its neighbour come from the same strand, while '-' or '+' types mean a different strand. After traversing a connected subgraph, the corresponding paired-end of those labelled reads are checked and be assigned to a different strand if they are unlabelled. Although there are possible contradictions that two ends of a read pair are labelled with the same symbol during the BFS traversal step, in reality, we did not observe such cases. After the whole graph traversal, all the reads labelled with '-' will be replaced with their reverse complements.

4 Clique enumeration in PEHaplo

Several recently published virus assembly tools (Töpfer *et al.*, 2014; Baaijens *et al.*, 2017) have employed clique enumeration for haplotype reconstruction. They usually merge reads inside each clique to super-reads and iteratively apply this process to extend a local haplotype to a global one. However, directly merging reads in a clique to a super-read may incorrectly join reads from different haplotypes. An example is shown in Figure S2(A, B, C). In this overlap graph (Figure S2(B)), if $a.1$ overlaps with $a.2$, it should also overlap with $d.2$ because $d.2$ and $a.2$ share the same prefix longer than or equal to the overlap threshold

(Figure S2(A)). Similarly, $d.1$ will also overlap with $a.2$ if it overlaps with $d.2$. Therefore, this overlap graph has four cliques of size 6 (Figure S2(C)). However, two of these cliques, $a.1 \rightarrow b \rightarrow c \rightarrow e \rightarrow f \rightarrow d.2$ and $d.1 \rightarrow b \rightarrow c \rightarrow e \rightarrow f \rightarrow a.2$, have reads from two haplotypes, which are "chimeric super-read".

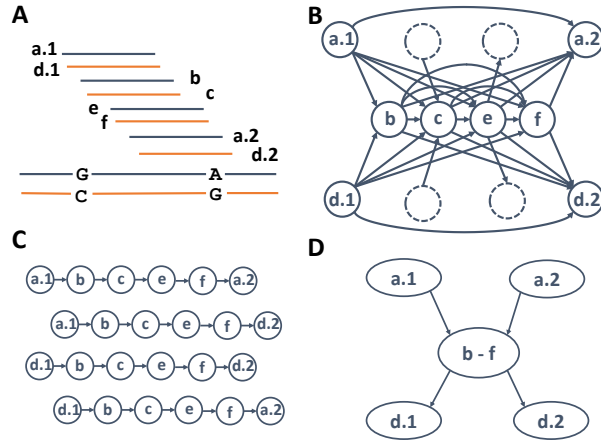


Fig. S2. (A) The bottom two long lines represent two haplotypes, which only differ by two mutations at two loci (G-C and A-G). Short lines represent reads sequenced from the two strains. The reads are sorted by their read mapping positions against their native strain. (B) Overlap graph. Nodes b , c , e , and f originate from the common region of the two strains. The dashed nodes have random overlaps with nodes c and e . (C) The overlap graph have 4 cliques of size 6. They can be merged to 4 super-reads by previous methods. But two of them have reads from two haplotypes. (D) The result of our clique enumeration.

In our methods, instead of directly converting each clique to a super-read, we are more cautious and use cliques mainly for graph pruning. When there is only a single clique, our processing is the same as others. However, the hard case comes from connected cliques, which is a cluster of cliques with two or more sharing nodes. Each clique cluster, denoted as G_c , is a connected subgraph. The nodes with zero in- or out degree in the subgraph of G_c are denoted as boundary nodes. All others are inside nodes. After we simplify the subgraph G_c with transitive reduction and node collapsing, the edges connecting inside nodes and nodes outside of G_c will be removed as they are likely to be random overlaps. As shown in Figure S2(D), the edges connecting dashed nodes and c , e in Figure S2(B) are removed after merging connected cliques. Those common nodes shared by cliques will be kept and paired-end information will be further applied to identify correct strains in path finding step. By merging connected cliques, we are able to prune the overlap graph while avoiding "chimeric super-read".

5 NP-complete (NPC) Proof for Path Finding in Paired-end Overlap Graph

Theorem: finding a path between two nodes in paired-end overlap graph with the most number of paired-end connections is NPC.

Proof: we can make a reduction from Hamiltonian Path problem, which is a famous NP-complete problem.

A Hamiltonian path in a directed graph is a directed path that goes through each node exactly once. The decision problem is: Given a directed graph $G = (V, E)$ and two nodes s and t in this graph, is there a Hamiltonian path from s to t in G ?

Because the path will go through each node exactly once and start from s , all the incoming edges to s will have no contributions in the solution.

Similarly, all the outgoing edges from t will have no contributions in the solution. Therefore we can assume that there are no incoming edges to s and no outgoing edges from t .

Given an instance of Hamiltonian path problem, we can create an instance of our problem as follows.

We construct a directed graph $G' = (V', E')$ by splitting each edge in G into two edges. For example, if (a, b) is an edge in G , then we add a new node v_{ab} between a and b so that edge (a, b) is replaced by two edges (a, v_{ab}) and (v_{ab}, b) in G' . Meanwhile we add a constraint pair (a, v_{ab}) for each new pair of edges. Additionally, we add an extra constraint pair (v, t) for each incoming edge from v to t in the new graph G' . To make it clear, we call nodes in V as old nodes, and the new added nodes as new nodes.

If G has $|E| = m$ edges and $|V| = n$ nodes, then there are m new nodes in G' and the number of constraints is $m + d$, where d is the in-degree of node t in G' . We can see that $|V'| = n + m$ and $|E'| = 2m$. Notice that there is exactly one old node appearing in each constraint.

Next we will show that the Hamiltonian path instance has a solution if and only if the instance of our problem has a path that satisfies n constraints.

If the instance of Hamiltonian path has a solution, i.e., there is a Hamiltonian path p from s to t in G , then there are $n - 1$ edges in p . We can find the corresponding path p' from s to t in G' , and there are $2(n - 1)$ edges in p' . Since the first $2n - 3$ edges satisfy $n - 1$ constraints, and the last edge is an incoming edge to t , which satisfies one constraint, we can see that p' satisfies n constraints. Therefore the instance of our problem has a path that satisfies n constraints.

If the instance of our problem has a path p' that satisfies n constraints, we need to show that path p' is from s to t and contains all the n old nodes, so that the corresponding path in G is a Hamiltonian path from s to t .

Because p' satisfies n constraints, and there is exactly one old node in each constraint, we can see that there are exactly n old nodes in p' . Since there are totally n old nodes and p' can go through each node at most once, path p' go through all the n old nodes. In our assumption, there are no incoming edges to s , and no outgoing edges from t . The graph G' is created from G , hence G' also has no incoming edges to s , and has no outgoing edges from t . But s and t must be somewhere on p' . Therefore, p' must be from s to t and contain all the n old nodes. Now we can find the corresponding path p in G , and p is a path that is from s to t visiting each node exactly once, which means that p is a Hamiltonian path in G .

6 Supplementary Figures and Tables

HXB2	AAAAATGGGAACCCAGATTGTAAGACTATTTTAAAGCATTGGGACGAGGCGCTACACT
NL43	AAAAATGGGAACCCAGATTGTAAGACTATTTTAAAGCATTGGGACGAGGCGCTACACT
YU2	AAAAATGGGAACCCAGATTGTAAGACTATTTTAAAGCATTGGGACGAGGCGCTACACT
JRCSF	AAAAATGGGAACCCAGATTGTAAGACTATTTTAAAGCATTGGGACGAGGCGCTACACT
89.6	AAAAATGGGAACCCAGATTGTAAGACTATTTTAAAGCATTGGGACGAGGCGCTACACT

Fig. S3. Partial multiple sequence alignment of five HIV-1 haplotypes. The alignment is produced using the program ClustalW (Thompson et al., 2002). For each column, the same color indicated the same bases.

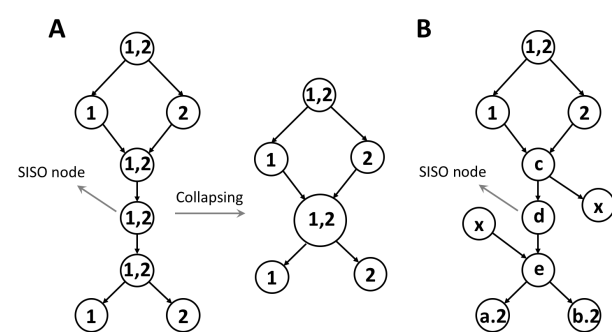


Fig. S4. SISO nodes usually represent one haplotype. (A) The conserved regions common to multiple haplotypes in the overlap graph are usually collapsed into one node, with in-degree and out-degree greater than 1. This is the case often being observed in the overlap graph. (B) The situation in which an SISO node comes from a common region of multiple haplotypes happens when both its predecessor and successor have incorrect/random edges with other nodes. Thus, the SISO node cannot be collapsed with them. Since error correction and graph pruning steps were applied before path finding, this situation is rare in our experiments. On the other hand, the SISO node in (B) do belong to two haplotypes, the extensions to both node 1 and node 2 are correct.

Table S1. Pairwise sequence similarity between 5 HIV-1 strains.

	89.6	HXB2	JRCSF	NL43	YU2
89.6		93.9	91.8	93.5	93.6
HXB2			92.8	97.4	95.2
JRCSF				92.6	92.9
NL43					94.9
YU2					

Table S2. Longest common substring (LCS) between 5 HIV-1 strains. These strains have similar lengths of about 10k bp.

	89.6	HXB2	JRCSF	NL43	YU2
89.6		195	201	164	234
HXB2			180	427	216
JRCSF				157	201
NL43					185
YU2					

Table S3. PEHaplo assembly results on simulated HIV data set with and without false edges removal step.

Assemble methods	Contigs num	N50	Genomes covered (%)	Unaligned length (bp)	Mismatch rate (%)	Indels (%)
Without removal	9	9,151	91.8	0	0.045	0.002
With removal	10	9,274	97.0	0	0.026	0.002

Table S4. Assembly results on pruned graph nodes for Ray Meta, Meta-IDBA, and SAVAGE. Contigs are aligned to the true haplotype sequences with a similarity cutoff of 98%.

Tools	Contig num	N50	Genomes covered (%)	Unaligned length	Mismatch rate (%)	Indels (%)
Ray-Meta	1	4,840	10.03	0	0.227	0
Meta-IDBA	27	1,237	57.23	0	0.007	0.004
SAVAGE	10	5,057	44.0	0	0	0

7 Supplementary algorithm

$e'_{SISO,v}$ represents a paired-end edge weight between SISO node(s) in the current path and v , which is a successor node of p_n . Function $C(v)$ denotes the read coverage of node v .

Algorithm 1 Greedy algorithm for path extension

```

1: if there exists a node  $v \in succ(p_n)$  with  $e'_{SISO,v} > 0$  then
2:   extend to the node with  $\text{argmax}_{v \in succ(p_n)}(e'_{SISO,v})$ 
3:   return
4: else if  $v \in succ(p_n)$  with  $e'_{SISO,w} > 0, w \in succ(v)$  in  $E$  then
5:   extend to the node with  $\text{argmax}_{v \in succ(p_n)}(e'_{SISO,w}), w \in succ(v)$  in  $E$ 
6:   return
7: else if  $v \in succ(p_n)$  with  $e'_{SISO,w} > 0, w \in succ'(v)$  in  $E'$  then
8:   extend to the node with  $\text{argmax}_{v \in succ(p_n)}(e'_{SISO,w}), w \in succ'(v)$  in  $E'$ 
9:   return
10: else if  $v \in succ(p_n)$  with  $e'_{Path,v} > 0$ 
11:   extend to the node with  $\text{argmax}_{v \in succ(p_n)}(e'_{Path,v})$  then
12:   return
13: else if  $v \in succ(p_n)$  with  $e'_{Path,w} > 0, w \in succ(v)$  in  $E$  then
14:   extend to the node with  $\text{argmax}_{v \in succ(p_n)}(e'_{Path,w}), w \in succ(v)$  in  $E$ 
15:   return
16: else
17:   extend to the node  $v \in succ(p_n)$  with  $\text{argmin}_{v \in succ(p_n)}(\text{abs}(C(v) - C(Path)))$ 
18:   return

```

8 Commands for running tools on simulated HIV data set

Input data sets: virus_1.fa, virus_2.fa or virus_1.fq, virus_2.fq

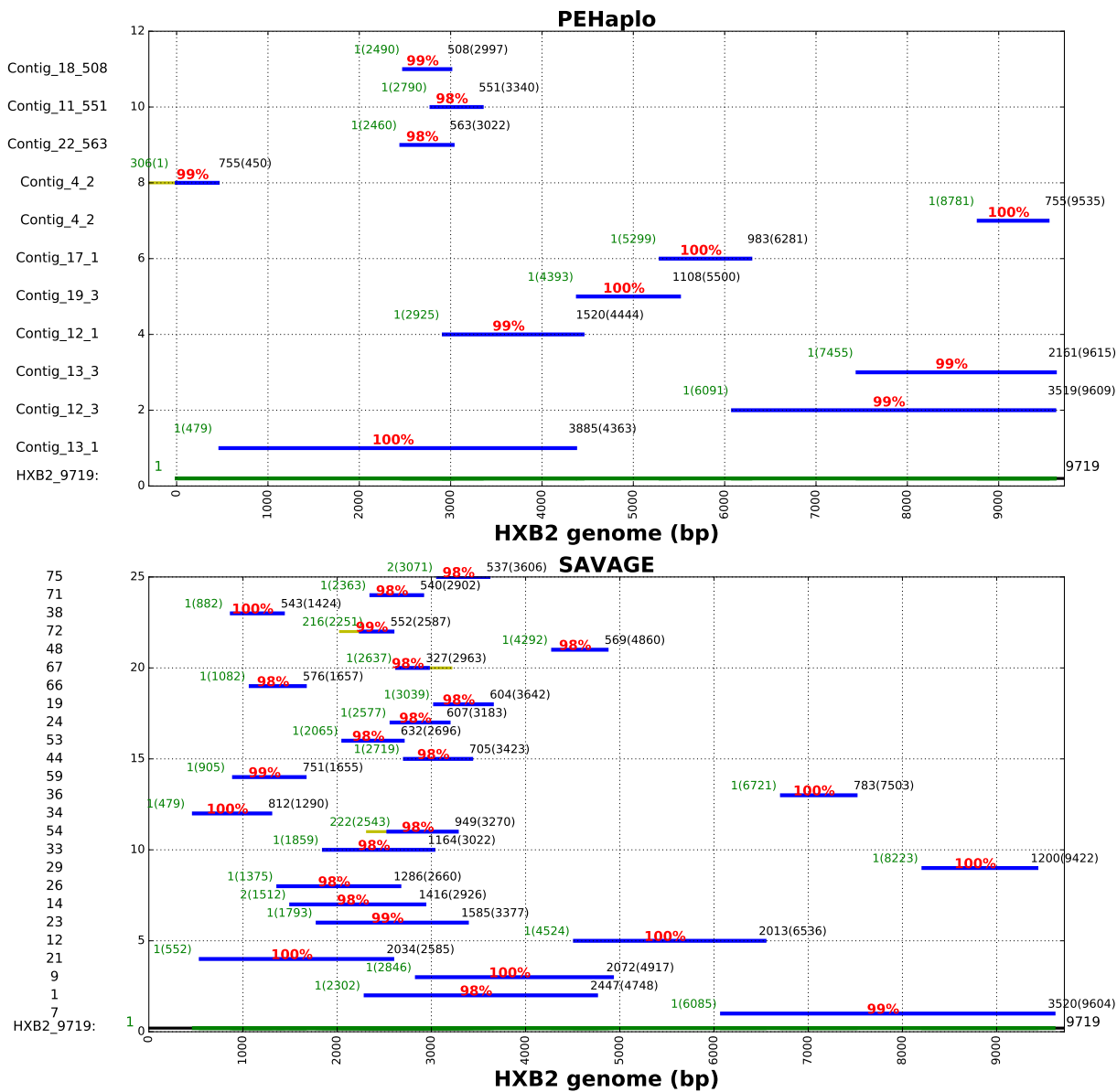


Fig. S5. Contigs alignment result on HXB2 strain for PEHaplo and SAVAGE. These contigs were produced from the real HIV MiSeq data and aligned to reference genome with MetaQuast. The x-axis is the coordinations of HXB2 genome, with the regions covered by contigs in green and others in black. The y-axis represents the number of contigs, with contig names listed on the left panel. On each contig, the green number at the left is the starting coordinate of the aligned contig and the number inside of the parenthesis shows the starting coordinate on the reference genome, the black value at the right is the ending coordinate of the aligned contig and the number inside of the parenthesis shows the ending coordinate on the reference. The red number at the middle is the sequence identity between the contig and reference genome.

```
IVA
iva -f virus_1.fq -r virus_2.fq --max_insert 800 --
  threads 4 result/

MLEHaplo
simulate_listoffiles: virus_1.fa, virus_2.fa
listofkmers: 55,45,35

../dsk-1.5655/multi-dsk simulate_listoffiles.txt
listofkmers.txt -d 10G -m 4G
../dsk-1.5655/parse_results solid_kmers_binary.55 >
simulate.55
python ../tools/join_pair_end_fasta.py virus_1.fa
virus_2.fa virus_whole.fa
```

```
perl ../construct_graph.pl virus_whole.fa simulate.55 0
    virus.55.graph "s"
perl ../construct_paired_without_bloom.pl -file1
    virus_1.fa -file2 virus_2.fa -paired -kmerfile
    simulate.55 -thresh 0 -wr virus_55.set.txt
```

```
perl ../dg_cover.pl -graph virus.55.graph -kmer
    simulate.55 -paired virus_55.set.txt -fact 5 -
    thresh 0 -IS 600 >virus.55.fact5.txt
perl ../process_dg.pl virus.55.fact5.txt >virus.55.
    fact5.fasta
perl ../get_paths_dgcover.pl -f virus.55.fact5.txt -w
    virus.55.fact5.paths.txt
```

```
perl ../likelihood_singles_wrapper_parallel.pl -
    condgraph virus.55.cond.graph -compset virus.55.
    comp.txt -pathsfile virus.55.fact5.paths.txt -back
    -gl 10000 -slow >virus.55.smxlik.txt
perl ../extract_MLE.pl -f virus.55.fact5.fasta -l virus
    .55.smxlik.txt >virus.55.MLE.fasta
```

SAVAGE

```
pear -f virus_1.fq -r virus_2.fq -o virus_join
```

```
python savage --split 20 --min_overlap_len 160 -s
    singles.fastq -p1 paired1.fastq -p2 paired2.fastq
    -t 16
```

PEHaplo

```
python pehaplo.py -f1 virus_1.fa -f2 virus_2.fa -l 180
    -ll 210 -r 250 -F 600 -std 150 -n 3 -correct yes
```

References

- Allam, A., Kalnis, P., and Solovyev, V. (2015). Karect: accurate correction of substitution, insertion and deletion errors for next-generation sequencing data. *Bioinformatics*, **31**(21), 3421–3428.
- Gonnella, G. and Kurtz, S. (2012). Readjoiner: a fast and memory efficient string graph-based sequence assembler. *BMC bioinformatics*, **13**(1), 82.
- Thompson, J. D., Gibson, T., Higgins, D. G., *et al.* (2002). Multiple sequence alignment using clustalw and clustalx. *Current protocols in bioinformatics*, pages 2–3.
- Töpfer, A., Marschall, T., Bull, R. A., Luciani, F., Schönhuth, A., and Beerenwinkel, N. (2014). Viral quasispecies assembly via maximal clique enumeration. *PLoS Comput Biol*, **10**(3), e1003515.
- Baaijens, J. A., El Aabidine, A. Z., Rivals, E., and Schönhuth, A. (2017). De novo assembly of viral quasispecies using overlap graphs. *Genome Research*, **27**(5), 835–848.