The Tiling Algorithm – A general method for structural characterization of accurate long DNA sequence reads: application to AAV genome sequences

Robert E. Bruccoleri[1,2*], David Rouleau[2], Celia Slater[2], Dimpal Lata[2], Christopher Phillion[2], Samuel Adjei[2], Kiran Adhikari[2], Serena Dollive[2]

1 Congenomics, LLC, Glastonbury, CT, USA
2 Oxford Biomedica (US) LLC, Bedford, MA, USA

* Corresponding author: Robert E. Bruccoleri: bruc@congen.com

Abstract

Adeno-associated virus (AAV), a common vector used in human gene therapy, is a challenging organism for DNA sequencing because its replication cycle results in structural rearrangements. In addition, the AAV manufacturing process can produce small fractions of viral particles containing host cell DNA and/or fragments of the helper plasmids. Pacific Biosciences (PacBio) long-read sequencers are capable of full length, single molecule sequencing of AAV viral genomes, but the analysis of the data is challenging. We present a simple algorithm for determining the arrangement of functional elements of single DNA molecules which can be aggregated to provide a sensitive measure of the population of sequences in a sample, including minor species. Using data from four publicly available datasets, we demonstrate our algorithm is able to characterize nearly all of the species in the AAV samples.

Introduction

Adeno-associated virus (AAV) is a common vector for human gene therapy [1]. Its use in clinical settings requires proof of the purity of any manufactured lot of the virus [2,3]. Determination of the DNA sequence via next-generation sequencing (NGS) is one appropriate method for assessing the purity and enumerating contaminants.

Sanger sequencing is not appropriate for assessing the broad contents of a manufactured lot because PCR amplification is required, and therefore the Sanger chromatograms represent the macroscopic summation of all the sequences that are amplifiable with a specified set of PCR probes. Thus, Sanger sequencing cannot capture low abundance or non-targeted DNA molecules, including both sequence variation in the vector and residual contaminants [4,5]. In addition, the read length for Sanger sequencing is too short to capture the entire viral genome, and thus assembly would be required. Any structural variants covered by separate reads would be difficult to assemble, because it would be difficult to determine if any relevant variants occurring in cis or trans.

Since the AAV genome is approximately 4.8 kb in length [6], Pacific Biosciences (PacBio) sequencers are capable of accurately sequencing individual AAV molecules using their Circular Consensus algorithm [7]. Thus, it is possible to analyze any lot of manufactured AAV as a population of individuals without PCR or assembly bias.

Analyzing AAV genomes is difficult because of the virus's unique replication process [8]. The AAV genome consists of a single stranded DNA molecule with the

genetic payload sandwiched between two inverted terminal repeats (ITRs) [9]. The replication process does not conserve the strandedness of the ITRs and the payload resulting in a variety of functional and acceptable vector genomes varying by ITR orientation and strandedness of the expression cassette for each AAV preparation [10].

Additional complications in the analysis of AAV sequences arise from the PacBio library preparation protocol for virus particles [11]. PacBio requires double stranded DNA for adapter ligation, and the capsid needs to be lysed at the beginning of library preparation. After lysis, the viral DNA molecules must anneal to make them double stranded. Thus, it is possible for two viral DNAs with different, but similar sequences to anneal [12]. In particular, two non complementary ITR sequences at each end can facilitate heteroduplex formation. The `--hd-finder` option in the PacBio CCS program is normally used to find heteroduplexes, but this possibility can confound accurate counting of the viral DNA sequences.

Besides the annealing step, another complication arises from the gap filling step whereby a DNA polymerase is used. If there is extended incubation time in the protocol, the 3' ITR end can serve as a primer for extending the single stranded viral genome into a double stranded DNA hairpin [13].

Also, the manufacturing process of AAVs can produce viruses containing partial genomes, DNA from the plasmids used in the manufacturing process, or DNA from the host cells [14].

Because of these complications, sequence analysis methods used for human genome resequencing do not apply for AAV. In particular, the alignment and variant calling algorithms will not properly account for the variation in strandedness of the connected ITRs and payloads or structural variation.

In addition to the expected full-length vector genomes, the viral production process is known to generate a diverse array of rare and unexpected genome structures. To comprehensively capture this variability, we elected to analyze all DNA molecules obtained in a sequencing run, rather than focusing solely on the predominant species. This approach allows us to address the key question: "What are the components of each DNA molecule present in the sample, and how are they arranged?". The tiling algorithm was developed specifically to answer this question.

Methods

The overarching principle of our methodology is to analyze each sequence individually and label them using a high-level description of the structure. These high-level descriptions are then grouped together, counted, and presented in descending order of abundance.

We use the verb, "tile", in the one-dimensional abstraction of the tiling of a two dimensional surface. The tiles should cover the surface as completely as possible with a user-modifiable allowance of inter-tile gaps. That is, for a one-dimensional sequence, we want to maximally cover the sequence with subsequences derived from known sequences, see Fig 1, with minimal or no gaps.

Fig 1 caption: Fig 1 illustrates the tiling concept. The read is shown as a horizontal line from the $5'$ to $3'$ end, with colored blocks (HSP 1 to HSP 5) representing matches to known components. Ideally, these HSPs should fully span the read.

In the context of AAV sequencing, it is also very important to understand the relationship between the "Flip" and "Flop" orientation of the ITR [15]. Specifically, if the 3' ends of Flip and Flop are extended by complementing the base paired 5' ends for 21 bases, i.e. the D region, Flip and Flop are revealed to be the exact reverse complement of each other. Thus, they can be treated as equivalent from a DNA sequence information perspective. We take advantage of this equivalence to simplify the

analysis by using the single extended ITR sequence as a single Basic Local Alignment Search Tool (BLAST) [16] reference sequence.

Because the AAV replication process randomizes the orientation of the ITRs at each end of the viral genome combinatorially [10], conventional read alignment approaches, which rely on a fixed reference orientation, are not suitable. Instead, the tiling methodology uses the BLAST algorithm [16] to find regions of each read that match the expected components of a viral genome separately, and then uses this information to find coverage of the read with minimum gaps or overlaps.

Specifically, the reference BLAST database consists of the extended ITR sequence described above, the payload, and the backbone subsequences of the packaging plasmid for the viral genome. These subsequences in the reference BLAST database are referred to as 'components' in the succeeding text. Every read from the PacBio sequencing run is aligned against this reference and high scoring pairs (HSPs) of reference component and coordinates of the alignment are generated. All strandedness possibilities are included.

Thus, a more precise statement of the goal of the tiling algorithm is to find a set of HSPs for every read, sorted by their coordinates on the read, where the end of one HSP is juxtaposed against the start of the next HSP with no overlap or gap. In addition, the first HSP should have a starting read coordinate of 1, and the last HSP should have ending read coordinate equal to the length of the read.

Because the ITRs are self-complementing and many payloads are self-complementing or have repetitive sequences, many possible HSPs can be generated from the alignment of a read against the reference sequences. In addition, sequencing errors will complicate the analysis.

These issues are addressed in three steps. The first step is to assess if a tiled coverage is possible. This is done by creating a character string of blanks with the same length as the read. Then, "X"s are written into this string using string coordinates equal to the read coordinates of all the HSPs. If gaps of blank characters are greater in length than the `min_gap_search` parameter, then it is known that the read cannot be tiled successfully based on the reference components used. This coverage test can be modified by an optional set of reference sequences which will be described later. The default setting for the `min_gap_search` parameter is 11 bases. (N.B. the `min_gap_search` parameter is unrelated to the gap search in the BLAST algorithm.)

The second step is redundant HSP filtering. In this step, all the HSPs are sorted by the starting coordinates of the read and then by sorting on the ending coordinates of the read. Then, a search is made of any HSP that is fully contained within another HSP with respect to the read coordinates, regardless of the strandedness. There are two cases of containment to consider. The simplest case is where the containing HSP is larger than the contained HSP, in which case the contained HSP is then deleted from consideration.

The more complex case is when the two HSPs cover exactly the same coordinates of the read. In that case, the HSPs have different strandedness. This occurs frequently with ITRs due to their palindromic and repetitive composition. Here, we examine how many mismatches and gaps are found in each HSP and we pick the HSP that has the smallest sum of mismatches and gaps. If these sums are identical for both HSPs, then we just pick the HSP that uses the positive strand of the reference for the alignment so the process is deterministic.

The final step is to exhaustively search for a set of HSPs that have the best coverage score. In this context, "best" means the minimization of the following score, S

$$S = \sum_{i=1}^{N_h} (\mathrm{NM}_i + \mathrm{NG}_i) + \sum_{j=1}^{N_h - 1} |b_{i+1} - e_i + 1| + G_e \tag{1}$$

where N_h or is the number of HSPs, NM_i is the number of mismatches in HSP_i, NG_i is the number of gaps in HSP_i, e_i is the end coordinate of the read in HSP_i, b_i is

beginning coordinate of the read in HSP_i and G_e is the difference between the read length and the end coordinate of the read in the final HSP.

The first term in S captures the quality of the HSPs, and the second term captures the quality of the tiling coverage between HSPs. The third term captures any uncovered bases at the 3' end. Any gap at the 5' end of the read is constant because the HSPs are sorted by the beginning coordinate for the read in the HSP and overlapping HSPs are reduced to a single, largest possible HSP. Thus, any 5' gap need not be considered in the minimization of the tiling score.

The exhaustive search is performed on a directed graph of HSPs where the root of the graph is the first HSP in the sorted order. Children of any node in the graph are those HSPs where the magnitude of the difference of parent node's ending coordinate from the child node starting coordinate is less than the `gap_limit` parameter. Fig 2 illustrates a simple, example graph.

Fig 2 Caption: Fig 2 shows a simple graph of possible HSPs and the paths that can be taken to generate a complete tiling pattern.

If the HSP nodes in this graph have many linked HSPs, the exhaustive search can take a very long time. Thus, the algorithm counts the number of nodes visited in the search and aborts the search if the count exceeds the `max_iter` parameter. For good quality, less heterogeneous sequencing runs, the HSP search graph is usually very narrow, and can be searched very quickly.

Once the best coverage path through the HSPs is determined, then the code generates a summary description of the read by concatenating the BLAST database sequence name, the coordinates of the HSP match on the component along with the HSP strandedness. The summary description consists of a concatenation of all HSPs and which reference coordinates and strand each HSP aligns to. Note that unaligned bases in the original read are not represented in the summary description.

Here's an example of a summary description of a read (from the 2021 dataset below, barcode bc1001):

```
ITR-FLIP[1-145](-) Payload[1-1831](-) ITR-FLIP[25-141](-) Payload[1-1831](+) ITR-FLIP[1-145](+)
```

From this example, one can see that this DNA sequence has self-complementary payload with complementary ITRs at the ends and a shortened ITR in the middle joining the double stranded payload.

The code then counts the number of occurrences of each of these tiling patterns and sorts them by abundance in reverse order.

Adjustments to the tiling process

There are some optional adjustments that can be made to the tiling algorithm's process. First, the code can generate "pseudo" HSPs out of homopolymers. These "pseudo" HSPs are identified by applying a simple regular expression which looks for repeats of the same base. The minimum number of bases in the homopolymer can be set by the `min_poly` parameter. These HSPs have a component name of "poly-" followed by the base. Optionally, the length of the homopolymer can be added. These homopolymer HSPs are especially useful when poly-adenylation is used in a library preparation protocol.

Another adjustment is the use of multiple reference sequences after the initial reference database BLAST search. During the coverage checking step, if a large gap is found, the code will check other references for HSPs to see if the gaps can be covered. One such use case for additional reference sequences is the detection of additional plasmids used in AAV manufacturing. Occasionally, DNA from these plasmids will be

incorporated into the capsid, and this additional reference check will identify them. In addition, host cell DNA can also be incorporated into AAV DNA. By adding a human genome assembly as a reference database, HSPs with host cell genome sequence can also be detected and included in the tiling patterns.

Strand counting

A challenge with counting DNA molecules that is specific to AAV is the annealing protocol typically used to prepare sequencing libraries. One Zero Mode Waveguide (ZMW) in the PacBio SMRT cell will sequence two viral DNAs at once. If both strands of the single molecule circle in a ZMW are the same, the CCS algorithm can successfully execute with just three passes of the polymerase, whereas if they are different, six passes are needed. More passes are less likely in any ZMW. With five passes, only one strand can be corrected and the other strand will be missed. Thus, such heteroduplexes will be undercounted relative to homoduplexes.

Hairpin extension artifacts also complicate DNA molecule counting. In this case, only one SMRT Bell will be ligated onto the end of the hairpin, and the ZMW is only sequencing one viral genome, not two.

We have not attempted to resolve this counting problem. Instead, the algorithm can generate two possible counts of tiling patterns. It will always generate a count based on the CCS reads. Optionally, it can generate a count based on what's seen in each ZMW. If the ZMW reports just one read, then the output is just the tiling pattern for that read. However, if there are two reads reported, the tiling computes the pattern for both and reports them both.

Implementation

The tiling algorithm is implemented in Perl using the Bio::Frescobi [17] Perl module which uses a relational database to store all the BLAST results and read sequences. The relational database greatly simplifies the retrieval of relevant HSPs for each read in an analysis. Frescobi supports either PostgreSQL or SQLite as the database engine. The code is available in Github [18].

An important property of the sequence database constructed by Frescobi is the 1:1 mapping of sequences to a short identifier, i.e. the `seqid`. Even if a sequence dataset has multiple read with identical sequences, each identical sequence will have the same identifier. Thus, any sequence based computation is only computed for unique sequences. Frescobi maintains a table to relate the name of a sequence to its identifier, so the abundance of sequences can be counted by finding all the names for a given `seqid`.

The script takes the name of sequence library in Frescobi as an input and can generate a large number of output files, see Table 1 for a list of commonly used output files. All of these files begin with the library name, and the extension indicates the type of data in the file. Certain execution options can generate additional files which are described with the tiling code documentation.

Material

We analyzed four AAV datasets [Table 2] available on the PacBio web site, https://downloads.pacbcloud.com/public/dataset/AAV/ using the tiling algorithm.

Table 1. Common Output files

Extension	Description of file contents
.tiles	Tile pattern for each unique read sequence.
.gap.seq	Subsequences in each read that were not identifiable.
.unmatched.seq	Full sequence of unique reads that had some sequence that was not identifiable.
.summary	Summary statistics of the tiling run.
.split.reads	Listing of sequences of each of the components for each unique read.
.tile.err	Any error messages.
.hsps	List of all HSPs for each unique read.
.tile.details	More details on each tiling pattern as a function of each unique read.
.name2seqid	The correspondence between `seqid` and the name of the sequence in the PacBio sequence file. This name identifies the ZMW number on the SMRT Cell.
.tile.zmw.counts	Tile patterns for the sequences per ZMW.
.feats	Counts for the features seen in the tiling run.
.tile.counts	Abundances for all the tile patterns sorted most abundant first.
.gap.counts	Abundances for the sequences that were not covered by reference sequences. This file is especially important in diagnosing what's unidentified in a sample.

Table 2. Listing of PacBio AAV datasets

Folder Name	Sample Name	Source
2021-scAAV-CBA-eGFP	2021-scAAV/rh10-CBA-eGFP	Vector Biolabs
2022-ssAAV-pAV-CMF-GFP	2022-pAV-CMV-GFP	Vigene Biosciencess
2022-ssAAV-scAAV-mix	Same as folder	Not specified
2023-Revio-ssAAV-pAV-CMF-GFP	Same as folder	Charles River

Results

The tiling algorithm was applied to the four AAV datasets, and the complete results were uploaded to Zenodo [19]

2021-scAAV-CBA-eGFP

This experiment had eight different samples that were multiplexed into one run. There is no information available on the differences between each sample. The percentage of sequences that could be characterized was very high across all eight samples with values between 99.4113% and 99.4849%.

We picked one sample arbitrarily for the display of detailed results. The sixteen most common patterns in the sample barcoded bc1002 were as follows:

```
41389 0.1288309 ITR-FLIP[1-145](-) Payload[1-1831](-) ITR-FLIP[25-141](+) Payload[1-1831](+) ITR-FLIP[1-145](+)
41325 0.1286317 ITR-FLIP[1-145](-) Payload[1-1831](-) ITR-FLIP[25-141](-) Payload[1-1831](+) ITR-FLIP[1-145](+)
40794 0.1269789 ITR-FLIP[21-165](+) Payload[1-1831](-) ITR-FLIP[25-141](+) Payload[1-1831](+) ITR-FLIP[21-165](-)
40623 0.1264466 ITR-FLIP[21-165](+) Payload[1-1831](-) ITR-FLIP[25-141](-) Payload[1-1831](+) ITR-FLIP[21-165](-)
18677 0.0581356 ITR-FLIP[1-145](-) Payload[1-1831](-) ITR-FLIP[25-141](-) Payload[1-1831](+) ITR-FLIP[21-165](-)
18535 0.0576936 ITR-FLIP[1-145](-) Payload[1-1831](-) ITR-FLIP[25-141](+) Payload[1-1831](+) ITR-FLIP[21-165](-)
17436 0.0542728 ITR-FLIP[21-165](+) Payload[1-1831](-) ITR-FLIP[25-141](+) Payload[1-1831](+) ITR-FLIP[1-145](+)
17362 0.0540424 ITR-FLIP[21-165](+) Payload[1-1831](-) ITR-FLIP[25-141](-) Payload[1-1831](+) ITR-FLIP[1-145](+)
 5859 0.0182372 ITR-FLIP[1-144](-) Payload[1-1831](-) ITR-FLIP[25-141](+) Payload[1-1831](+) ITR-FLIP[1-144](+)
 5816 0.0181034 ITR-FLIP[1-144](-) Payload[1-1831](-) ITR-FLIP[25-141](-) Payload[1-1831](+) ITR-FLIP[1-144](+)
 5767 0.0179509 ITR-FLIP[22-165](+) Payload[1-1831](-) ITR-FLIP[25-141](+) Payload[1-1831](+) ITR-FLIP[22-165](-)
 5458 0.0169890 ITR-FLIP[22-165](+) Payload[1-1831](-) ITR-FLIP[25-141](-) Payload[1-1831](+) ITR-FLIP[22-165](-)
 1950 0.0060697 ITR-FLIP[1-144](-) Payload[1-1831](-) ITR-FLIP[25-141](-) Payload[1-1831](+) ITR-FLIP[22-165](-)
 1884 0.0058643 ITR-FLIP[1-144](-) Payload[1-1831](-) ITR-FLIP[25-141](+) Payload[1-1831](+) ITR-FLIP[22-165](-)
 1799 0.0055997 ITR-FLIP[22-165](+) Payload[1-1831](-) ITR-FLIP[25-141](+) Payload[1-1831](+) ITR-FLIP[1-144](+)
 1731 0.0053881 ITR-FLIP[22-165](+) Payload[1-1831](-) ITR-FLIP[25-141](-) Payload[1-1831](+) ITR-FLIP[1-144](+)
```

The numbers on the first column represent how many times a pattern was tiled or the species existed in the sample. The second column represents the proportion of reads that have the particular pattern. As expected, we found majority of the patterns show a self complementary payload with a shortened ITR in the middle (no D region) and full length ITRs at the ends. The sequence differences between the patterns of similar abundance occur because of Flip or Flop ITR orientation or the strandedness of the shortened, interior ITR. The less abundant patterns show one missing base at the end of the ITR, i.e. 144 bases instead of 145. There are a total of 15,830 patterns seen, although many are just single occurrence patterns.

2022-ssAAV-pAV-CMF-GFP

We were able to tile 93.7% of the reads from this dataset. The most abundant pattern is

```
9959 0.0142946 ITR-FLIP[21-165](+) Payload[1-2164](-) ITR-FLIP[21-165](-)
```

Unlike the previous sample where the most abundant pattern represented about 13% of the reads, the most abundant pattern in this sample represented only about 1.4% of the species present. Interestingly, we observed a substantial number of patterns which indicate strand extension during library preparation. Here is the most common such pattern:

```
2703 0.0038797 ITR-FLIP[22-165](+) Payload[1-2164](+) ITR-FLIP[1-165](-) Payload[1-2164](-) ITR-FLIP[22-165](-)
```

There are a total of 173,677 patterns seen in this sample, although 135,615 are single occurrence patterns (78%)

2022-ssAAV-scAAV-mix

This run was apparently a mixture test using the two samples above. The reference sequence used for tiling was ITR-FLIP plus the payload and backbones for the above two viral genomes. The percentage of reads that could be tiled was high: 98.949%.

The 24 most abundant tiling patterns are as follows:

```
27416 0.0910731 ITR-FLIP[1-145](-) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](+) Payload_scAAV[1-1831](+) ITR-FLIP[1-145](+)
27051 0.0898606 ITR-FLIP[1-145](-) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](-) Payload_scAAV[1-1831](+) ITR-FLIP[1-145](+)
26806 0.0890467 ITR-FLIP[21-165](+) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](+) Payload_scAAV[1-1831](+) ITR-FLIP[21-165](-)
26598 0.0883558 ITR-FLIP[21-165](+) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](-) Payload_scAAV[1-1831](+) ITR-FLIP[21-165](-)
14035 0.0466228 ITR-FLIP[1-145](-) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](-) Payload_scAAV[1-1831](+) ITR-FLIP[21-165](-)
13948 0.0463338 ITR-FLIP[1-145](-) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](+) Payload_scAAV[1-1831](+) ITR-FLIP[21-165](-)
13371 0.0444171 ITR-FLIP[21-165](+) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](+) Payload_scAAV[1-1831](+) ITR-FLIP[1-145](+)
13362 0.0443872 ITR-FLIP[21-165](+) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](-) Payload_scAAV[1-1831](+) ITR-FLIP[1-145](+)
 5493 0.0182472 ITR-FLIP[1-148](-) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](+) Payload_scAAV[1-1831](+) ITR-FLIP[1-148](+)
 5265 0.0174898 ITR-FLIP[1-148](-) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](-) Payload_scAAV[1-1831](+) ITR-FLIP[1-148](+)
 5122 0.0170147 ITR-FLIP[18-165](+) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](+) Payload_scAAV[1-1831](+) ITR-FLIP[18-165](-)
 5058 0.0168021 ITR-FLIP[18-165](+) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](-) Payload_scAAV[1-1831](+) ITR-FLIP[18-165](-)
 2555 0.0084874 ITR-FLIP[1-148](-) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](+) Payload_scAAV[1-1831](+) ITR-FLIP[18-165](-)
 2490 0.0082715 ITR-FLIP[1-148](-) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](-) Payload_scAAV[1-1831](+) ITR-FLIP[18-165](-)
 2428 0.0080656 ITR-FLIP[18-165](+) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](+) Payload_scAAV[1-1831](+) ITR-FLIP[1-148](+)
 2377 0.0078961 ITR-FLIP[18-165](+) Payload_scAAV[1-1831](-) ITR-FLIP[25-141](-) Payload_scAAV[1-1831](+) ITR-FLIP[1-148](+)
 1802 0.0059861 ITR-FLIP[1-145](-) Payload_pAV_CMV_GFP[1-2164](+) ITR-FLIP[1-145](+)
 1790 0.0059462 ITR-FLIP[21-165](+) Payload_pAV_CMV_GFP[1-2164](+) ITR-FLIP[21-165](-)
 1748 0.0058067 ITR-FLIP[21-165](+) Payload_pAV_CMV_GFP[1-2164](-) ITR-FLIP[21-165](-)
 1731 0.0057502 ITR-FLIP[21-165](+) Payload_pAV_CMV_GFP[1-2164](+) ITR-FLIP[1-145](+)
 1698 0.0056406 ITR-FLIP[1-145](-) Payload_pAV_CMV_GFP[1-2164](+) ITR-FLIP[21-165](-)
 1697 0.0056373 ITR-FLIP[1-145](-) Payload_pAV_CMV_GFP[1-2164](-) ITR-FLIP[1-145](+)
 1696 0.0056339 ITR-FLIP[1-145](-) Payload_pAV_CMV_GFP[1-2164](-) ITR-FLIP[21-165](-)
 1685 0.0055974 ITR-FLIP[21-165](+) Payload_pAV_CMV_GFP[1-2164](-) ITR-FLIP[1-145](+)
```

The tiling patterns were analyzed by a short Perl script which counted the number of reads that had either one of the two AAV payloads, both, or neither. The self complementary AAV percentage was 85.814% and the single stranded percentage was 14.061%, with the ratio being 6.103. A small fraction (0.074%) were ITR fragments, and another small fraction (0.052%) had both sequences from both libraries, although the maximum number of copies seen for this category was 2 copies out of 301,033 total reads.

For experiments where assessment of mixture ratios for different AAVs is important, the tiling algorithm can be very useful, because each read is characterized independently. As long as each AAV has unique sequences relative to the other AAVs in the mixture, the ratio of counts is definitive, subject to the strand counting caveat above.

2023-Revio-ssAAV-pAV-CMF-GFP

Two replicates were sequenced, but there's no indication that they are different.

The percentage of sequences that could be tiled was lower than the previous three datasets with replicate 1 being 89.054% and replicate 2 being 87.4374%. The number of unique tiling patterns for each replicate are 904,153 for replicate 1 and 1,488,692 for replicate 2.

The most abundant 24 tile patterns for the two replicates are listed below:

Replicate 1 most abundant tiling patterns

```
14561 0.0089000 ITR-FLIP[21-165](+) Payload[1-2164](-) ITR-FLIP[21-165](-)
14496 0.0088603 ITR-FLIP[1-145](-) Payload[1-2164](-) ITR-FLIP[1-145](+)
11057 0.0067583 ITR-FLIP[22-165](+) Payload[1-2164](-) ITR-FLIP[22-165](-)
10774 0.0065853 ITR-FLIP[1-144](-) Payload[1-2164](-) ITR-FLIP[22-165](-)
10729 0.0065578 ITR-FLIP[21-165](+) Payload[1-2164](+) ITR-FLIP[21-165](-)
10707 0.0065444 ITR-FLIP[1-144](-) Payload[1-2164](-) ITR-FLIP[1-144](+)
10667 0.0065199 ITR-FLIP[1-145](-) Payload[1-2164](+) ITR-FLIP[1-145](+)
10513 0.0064258 ITR-FLIP[1-145](-) Payload[1-2164](-) ITR-FLIP[21-165](-)
 9666 0.0059081 ITR-FLIP[22-165](+) Payload[1-2164](-) ITR-FLIP[1-144](+)
 9598 0.0058665 ITR-FLIP[21-165](+) Payload[1-2164](-) ITR-FLIP[1-145](+)
 8350 0.0051037 ITR-FLIP[1-144](-) Payload[1-2164](+) ITR-FLIP[22-165](-)
 7987 0.0048818 ITR-FLIP[1-145](-) Payload[1-2164](+) ITR-FLIP[21-165](-)
 7821 0.0047804 ITR-FLIP[1-144](-) Payload[1-2164](+) ITR-FLIP[1-144](+)
 7761 0.0047437 ITR-FLIP[22-165](+) Payload[1-2164](+) ITR-FLIP[22-165](-)
 7621 0.0046581 ITR-FLIP[21-165](+) Payload[1-2164](+) ITR-FLIP[1-145](+)
 7560 0.0046208 ITR-FLIP[22-165](+) Payload[1-2164](+) ITR-FLIP[1-144](+)
 2194 0.0013410 ITR-FLIP[1-144](-) Payload[1-2164](-) ITR-FLIP[21-165](-)
 2142 0.0013092 ITR-FLIP[1-144](-) Payload[1-2164](-) ITR-FLIP[1-145](+)
 2036 0.0012445 ITR-FLIP[22-165](+) Payload[1-2164](-) ITR-FLIP[21-165](-)
 2026 0.0012383 ITR-FLIP[22-165](+) Payload[1-2164](-) ITR-FLIP[1-165](-) Payload[1-2164](+) ITR-FLIP[22-165](-)
 1983 0.0012121 ITR-FLIP[1-144](-) Payload[1-2164](+) ITR-FLIP[1-165](-) Payload[1-2164](-) ITR-FLIP[1-144](+)
 1958 0.0011968 ITR-FLIP[1-144](-) Payload[1-2164](-) ITR-FLIP[1-165](-) Payload[1-2164](+) ITR-FLIP[1-144](+)
 1942 0.0011870 ITR-FLIP[1-144](-) Payload[1-2164](-) ITR-FLIP[1-165](+) Payload[1-2164](+) ITR-FLIP[1-144](+)
 1910 0.0011674 ITR-FLIP[22-165](+) Payload[1-2164](-) ITR-FLIP[1-165](+) Payload[1-2164](+) ITR-FLIP[22-165](-)
```

Replicate 2 most abundant tiling patterns

```
17604 0.0064796 ITR-FLIP[21-165](+) Payload[1-2164](-) ITR-FLIP[21-165](-)
17030 0.0062683 ITR-FLIP[1-145](-) Payload[1-2164](-) ITR-FLIP[1-145](+)
13788 0.0050750 ITR-FLIP[21-165](+) Payload[1-2164](+) ITR-FLIP[21-165](-)
13523 0.0049775 ITR-FLIP[1-145](-) Payload[1-2164](+) ITR-FLIP[1-145](+)
12799 0.0047110 ITR-FLIP[22-165](+) Payload[1-2164](-) ITR-FLIP[22-165](-)
12770 0.0047003 ITR-FLIP[1-144](-) Payload[1-2164](-) ITR-FLIP[22-165](-)
12383 0.0045579 ITR-FLIP[1-144](-) Payload[1-2164](-) ITR-FLIP[1-144](+)
12366 0.0045516 ITR-FLIP[1-145](-) Payload[1-2164](-) ITR-FLIP[21-165](-)
11335 0.0041721 ITR-FLIP[21-165](+) Payload[1-2164](-) ITR-FLIP[1-145](+)
11173 0.0041125 ITR-FLIP[22-165](+) Payload[1-2164](-) ITR-FLIP[1-144](+)
10391 0.0038247 ITR-FLIP[1-145](-) Payload[1-2164](+) ITR-FLIP[21-165](-)
10341 0.0038063 ITR-FLIP[1-144](-) Payload[1-2164](+) ITR-FLIP[22-165](-)
 9959 0.0036657 ITR-FLIP[22-165](+) Payload[1-2164](+) ITR-FLIP[22-165](-)
 9802 0.0036079 ITR-FLIP[1-144](-) Payload[1-2164](+) ITR-FLIP[1-144](+)
 9597 0.0035324 ITR-FLIP[22-165](+) Payload[1-2164](+) ITR-FLIP[1-144](+)
 9549 0.0035147 ITR-FLIP[21-165](+) Payload[1-2164](+) ITR-FLIP[1-145](+)
 3781 0.0013917 ITR-FLIP[22-165](+) Payload[1-2164](-) ITR-FLIP[1-165](-) Payload[1-2164](+) ITR-FLIP[22-165](-)
 3731 0.0013733 ITR-FLIP[1-144](-) Payload[1-2164](+) ITR-FLIP[1-165](-) Payload[1-2164](-) ITR-FLIP[1-144](+)
 3720 0.0013692 ITR-FLIP[22-165](+) Payload[1-2164](-) ITR-FLIP[1-165](+) Payload[1-2164](+) ITR-FLIP[22-165](-)
```

```
3711 0.0013659 ITR-FLIP[1-144](-) Payload[1-2164](-) ITR-FLIP[1-165](+) Payload[1-2164](+) ITR-FLIP[1-144](+)
3677 0.0013534 ITR-FLIP[1-144](-) Payload[1-2164](-) ITR-FLIP[1-165](-) Payload[1-2164](+) ITR-FLIP[1-144](+)
3598 0.0013243 ITR-FLIP[22-165](+) Payload[1-2164](+) ITR-FLIP[1-165](-) Payload[1-2164](-) ITR-FLIP[22-165](-)
3544 0.0013045 ITR-FLIP[1-144](-) Payload[1-2164](+) ITR-FLIP[1-165](+) Payload[1-2164](-) ITR-FLIP[1-144](+)
3437 0.0012651 ITR-FLIP[22-165](+) Payload[1-2164](+) ITR-FLIP[1-165](+) Payload[1-2164](-) ITR-FLIP[22-165](-)
```

Compared to the previous samples, these sequencing runs show much more heterogeneity in the tiling results, as indicated by the number of unique tiling patterns. In addition, the most abundant patterns have a smaller frequency of occurrence relative to the tiling patterns for the other datasets (e.g. 0.9% for the first replicant versus 12.9% in the 2021-scAAV-CBA-eGFP for the most abundant pattern in the sample with the bc1002 barcode). We also see examples of strand extension on the ITRs in the last eight tiling patterns above.

The analysis of these AAV sequences in these sequencing libraries using the tiling algorithm reveals a very high level of diversity in the population of viral DNAs. Many of them are low abundance, but not single occurrence. Here's an example from the second replicate of the 2023 Revio run:

```
394 0.0001450 ITR-FLIP[1-145](-) Payload[543-2164](-) Payload[614-2164](+) ITR-FLIP[1-145](+)
```

The payload DNA has folded back on itself and generated a hairpin. The fact that there were 394 copies of this DNA structure strongly suggests that it is not an computational artifact.

Discussion

A key strength of our approach is its ability to identify noncanonical and complex genome arrangements by exhaustively parsing tiling patterns using high-scoring pairs (HSPs) from BLAST alignment against annotated reference components. This allows us to map every segment within each CCS read including strand orientation and position to reconstruct a comprehensive genome layout. By segregating repetitive elements in the reference BLAST database, we can properly annotate repetitive elements in a small genome.

Alternative AAV genome classification frameworks were developed to address similar challenges [20]. The pipelines typically combine HiFi CCS read generation with alignment via Minimap2 and use BLAST-based parsing in Python to classify sequences into structural categories such as full genomes, snapback genomes, genome deletion mutants, and other noncanonical forms [21]. In contrast, our pipeline provides a more granular, tile-based reconstruction of each AAV genome by aligning and identifying full-length reference components including ITRs, payloads, and backbones using BLAST. Rather than relying solely on coordinate overlaps, our method maps all segments within each read to reconstruct the complete genomic configuration. This includes accurate determination of components, their count, strand orientation, and completeness (full or partial).

For instance, while other methods may flag an ITR based on a short alignment, our approach ensures that only biologically meaningful and structurally complete ITRs are recognized. Furthermore, by reporting all unique tiling patterns and their abundances, our pipeline enables high-resolution insight into both dominant and low-frequency structural variants, offering a better view of the diversity within AAV vector preparations and reducing the data complexity by approximately two orders of magnitude.

In addition to identifying well-matched patterns, the tiling algorithm also reports reads or parts of reads that don't match anything in the reference. These "untiled" reads and unmatched parts are saved in output files (like .gap.seq and .unmatched.seq), allowing scientists to look more closely at unexpected or unknown sequences that may come from contamination or missing reference information.

The use of an exhaustive search to find the best tiling pattern is potentially a problem for the general use of the tiling algorithm. In particular, repetitive elements in the reference sequence will result in many overlapping HSPs, and the search will not complete in a reasonable time.

The challenge with the tiling analysis is high level of detail. Typical tiling runs can generate tens of thousands of patterns, but many of them are from very low abundance DNAs (literally parts per million). In addition, the coordinates in the HSPs are affected by sequencing errors, and small differences in the endpoints are typically not significant. Thus, there is great value in processing the tiling output to summarize the patterns that are seen. D. Rouleau et. al. have a manuscript in preparation which describes analytic methods that provide additional insights into the composition of a sequenced viral genome sample.

Although demonstrated here in the context of adeno-associated virus (AAV) genome analysis, the tiling algorithm is generalizable and can be applied to other DNA molecules with known component structures. Nonetheless, certain design features may limit its broader applicability as discussed above. It is important to note that the algorithm does not perform base-pair level sequence analysis or variant calling. Instead, it provides high-level structural annotations of each DNA molecule, identifying the order, orientation, and boundaries of predefined genomic components such as inverted terminal repeats (ITRs), payloads, plasmid backbone elements, etc. These structural annotations form a framework for downstream analyses. When nucleotide-level resolution is required, the algorithm's output coordinates can be used to extract relevant subsequences from individual reads. The software implementation includes functionality to generate these coordinates and subsequences for each annotated molecule.

Conclusion

We have presented an algorithm for structural analysis of DNA sequences that can handle rearrangements on individual AAV molecules. The algorithm also provides the mechanism to identify unexpected components in a DNA sample as well as provide some quantification of proportions in a mixture. Designed originally for AAV genome analysis, our algorithm can be utilized with any DNA molecules with known component substructures and structural variability.

Acknowledgments

We thank James McGivney for his leadership in supporting AAV sequencing at Oxford Biomedica. We thank Jeong-Ah Kwon, Brianna Uga, Esra D. Camci, and Conner Traugot for their laboratory work in preparing samples and running the Sequel II instrument to provide high quality sequence data for analysis.

We thank Pacific Biosciences for technical assistance with processing sequence data and smooth functioning of our Sequel II instrument.

References

1. Lisowski L, Tay SS, Alexander IE. Adeno-associated virus serotypes for gene therapeutics. Current opinion in pharmacology. 2015;24:59–67. doi:10.1016/j.coph.2015.07.006.

2. Rosas LE, Grieves JL, Zaraspe K, La Perle KM, H F, M MD. Patterns of scAAV vector insertion associated with oncogenic events in a mouse model for genotoxicity. Molecular Therapy. 2012;20:2098–2110. doi:10.1038/mt.2012.197.

3. Schmidt M, Foster GR, Coppens M, Thomsen H, Cooper D, Dolmetsch R, et al.. Liver safety case report from the phase 3 HOPE-B gene therapy trial in adults with hemophilia B.; 2021. Available from: `https://genetherapy.isth.org/liver-safety-case-report-from-the-phase-3-hope-b-gene-therapy-trial-in-adults-with-hemophilia-b`.

4. Lecomte E, Tournaire B, Cogné B, Dupont JB, Lindenbaum P, Martin-Fontaine M, et al. Advanced Characterization of DNA Molecules in rAAV Vector Preparations by Single-stranded Virus Next-generation Sequencing. Molecular Therapy Nucleic Acids. 2015;4. doi:10.1038/mtna.2015.32.

5. Maynard LH, Smith O, Tilmans NP, Tham E, Hosseinzadeh S, Tan W, et al. Fast-Seq: A Simple Method for Rapid and Inexpensive Validation of Packaged Single-Stranded Adeno-Associated Viral Genomes in Academic Settings. Human Gene Therapy Methods. 2019;30(6):195–205. doi:10.1089/hgtb.2019.110.

6. Naso MF, Tomkowicz B, Perry WL, Strohl WR. Adeno-Associated Virus (AAV) as a Vector for Gene Therapy. BioDrugs. 2017;31:317–334. doi:10.1007/s40259-017-0234-5.

7. Wenger AM, Peluso P, Rowell WJ, Chang PC, Hall RJ, Concepcion GT, et al. Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. Nature Biotechnology. 2019;37:1155–1162. doi:10.1038/s41587-019-0217-9.

8. McCarty D. Self-complementary AAV vectors; advances and applications. Molecular therapy : the Journal of the American Society of Gene Therapy. 2008;16:1648–56. doi:10.1038/mt.2008.171.

9. Berns KI. The Unusual Properties of the AAV Inverted Terminal Repeat. Human Gene Therapy. 2020;31(9-10):518–523. doi:10.1089/hum.2020.017.

10. Chen Y, Hu S, Lee W, Walsh N, Iozza K, Huang N, et al. A Comprehensive Study of the Effects by Sequence Truncation within Inverted Terminal Repeats (ITRs) on the Productivity, Genome Packaging, and Potency of AAV Vectors. Microorganisms. 2024;12(2). doi:10.3390/microorganisms12020310.

11. Pacific-Biosciences. Preparing multiplexed AAV SMRTbell® libraries using SMRTbell prep kit 3.0; 2025.

12. Wright JF, Le T, Prado J, Bahr-Davidson J, Smith PH, Zhen Z, et al. Identification of factors that contribute to recombinant AAV2 particle aggregation and methods to prevent its occurrence during vector purification and formulation. Molecular therapy : the journal of the American Society of Gene Therapy. 2005;12:171–178. doi:10.1016/j.ymthe.2005.02.021.

13. Lkharrazi A, Tobler K, Marti S, Bratus-Neuenschwander A, Vogt B, Fraefel C. AAV2 can replicate its DNA by a rolling hairpin or rolling circle mechanism, depending on the helper virus. Journal of Virology. 2024;98(11):e01282–24. doi:10.1128/jvi.01282-24.

14. McColl-Carboni A, Dollive S, Laughlin S, Lushi R, MacArthur M, Zhou S, et al. Analytical characterization of full, intermediate, and empty AAV capsids. Gene Therapy. 2024;31:285–294. doi:10.1038/s41434-024-00444-2.

15. Willmot P, Lisowski L, Alexander IE, Logan GJ. A User's Guide to the Inverted Terminal Repeats of Adeno-Associated-Virus. Human Gene Therapy Methods. 2019;30(6):206–213.

16. Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, et al. BLAST+: architecture and applications. BMC Bioinformatics. 2009;10:421. doi:10.1186/1471-2105-10-421.

17. Bruccoleri RE. Free Relational Eclectic System for the Collection Of Biological Information; 2025. Available from: `https://github.com/bruc/BioFrescobi`.

18. Bruccoleri RE. The Tiling algorithm used for DNA sequence analysis; 2025. Available from: `https://github.com/bruc/tiling`.

19. Bruccoleri RE. Tiling Results of 4 AAV sequencing datasets produced on a Pacific Biosciences Sequel II Sequencer; 2025. Available from: `https://doi.org/10.5281/zenodo.16064304`.

20. Zhang J, Yu X, Chrzanowski M, Tian J, Pouchnik D, Guo P, et al. Thorough molecular configuration analysis of noncanonical AAV genomes in AAV vector preparations. Molecular Therapy - Methods & Clinical Development. 2024;32(1):101215. doi:10.1016/j.omtm.2024.101215.

21. Talevich E, Tseng E, Diallo A, Sellami N, Elliott A, Cantarel B, et al. Standardized Nomenclature and Reporting for PacBio HiFi Sequencing and Analysis of rAAV Gene Therapy Vectors. bioRxiv. 2024;doi:10.1101/2024.05.07.592296.

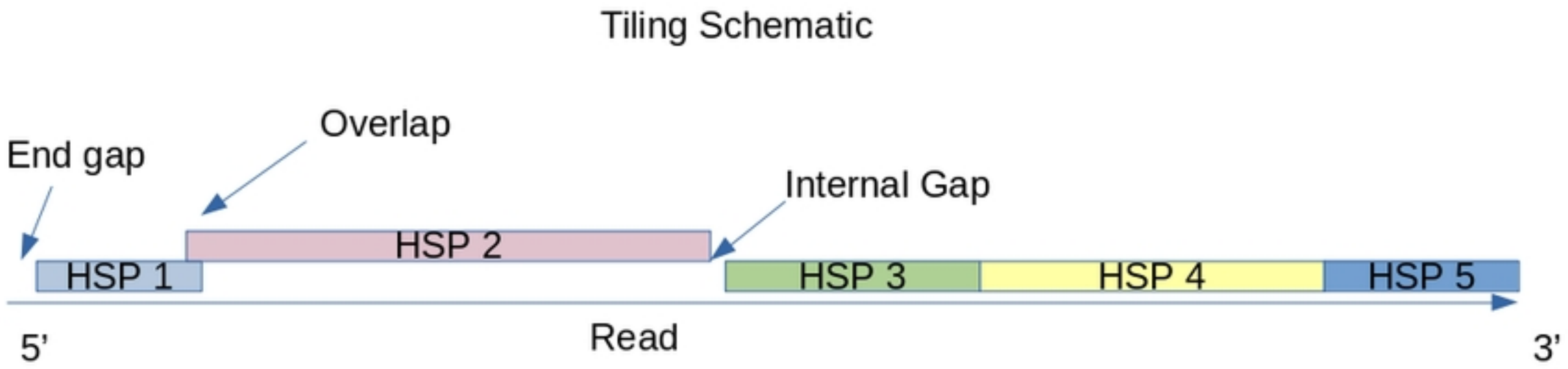
Tiling Schematic
Overlap
End gap
Internal Gap
HSP 2
HSP 1
HSP 3
HSP 4
HSP 5
5'
Read
3'

Figure 1

Tile Search

Possible tiling patterns:
1 2 3 4 5
1 6 7 4 5
1 6 8 9 4 5

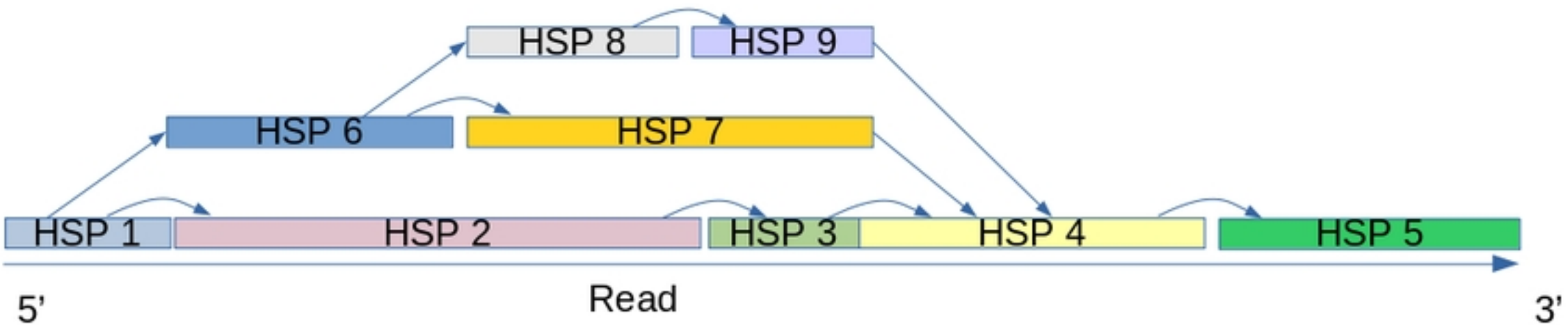

Figure 2