Development of Fast, Slim and Accurate Amplicon Variant Detection Algorithm for Next Generation Sequencing

Wei-min Liu[1], Yan Li[1]
[1]Roche Molecular Systems, Inc., 4300 Hacienda Dr, Pleasanton, CA 94588

Abstract
Amplicon variant detection (AVD) is one of the important applications for next generation sequencing (NGS). The enormous amount of sequencing data generated by the NGS technology necessitates the development of fast, slim and accurate AVD algorithm. Here, we present our approaches to develop and implement an AVD algorithm with fast processing speed and efficient use of memory space. The hash function and Burrows-Wheeler transformation are customized to speed up AVD so that processing the large amount of sequencing reads can be completed in a relative short time. We also propose a special way to compress the alignments of reads with reference sequences. Our format uses significantly smaller memory space in comparison with some widely used formats such as SAM/BAM. Moreover, it is useful to automatically identify whether the newly found variants exist in a public database such as COSMIC. We also discuss accuracy and other issues in evaluation of AVD algorithms for NGS.

Key Words: alignment, cancer, complex mutation, next generation sequencing, NGS, sequencing

1. Introduction

The fast progress of the next generation sequencing (NGS) technology opens exciting opportunities for clinical applications. The deep sequencing of target gene amplicons can identify disease related mutations even if they appear in low proportions in specimens. Since the amount of sequencing read data is large, development of fast and slim AVD algorithm is required. Moreover, sequencing noises can occur in many steps, such as contamination during sample preparation, improper primer design, infidelity of DNA amplification, imperfect base calling algorithm, problematic alignment algorithm and variant calling algorithm. In order to reduce sequencing noise, a common practice is to apply various filters to remove low quality reads or suppress the report of unreliable variants. In this paper, we remark on several approaches that can make AVD algorithm fast and save memory space. This is especially important when we apply NGS in clinical settings where reduced turn-around time means saving patients' lives in certain circumstances. We also discuss accuracy and other issues in evaluation of AVD algorithms.

2. Amplicon Recognition

Amplicon recognition includes de-multiplexing and recognizing the target gene segments based on the gene-specific primer sequences.

To reduce costs or to compare various sample preparation methods, multiple samples are usually included in one sequencing run. Specially designed short sequence segments are attached to the sample sequences as their identifier. They are known as molecular barcodes, or multiplex identifiers (MIDs, in 454 terminology), or indices (in Illumina terminology). Due to sequencing errors, algorithms should allow small variations from the designed barcode sequence.

A naïve implementation is to check all used barcodes for every read. Obviously, the speed of this implementation is $O(N)$, where N is the number of all used barcodes. A better implementation is to use hash so that it can be done in constant time $O(1)$ for reads with exactly matching barcode. For reads without exactly matching barcode, slower algorithm can be used. Since most reads have exactly matching barcode, it is worthwhile to first try hashing and then a slower algorithm. However, if the length of barcode is relatively long, e.g., 10 or more bases, then the hash table becomes large. For 10-base barcodes, to avoid complex collision resolving function, one may use 4 ^ 10 = 1,048,576 buckets. In this case, we can consider cascade hash tables. For example, we can use two hash tables, one for the first 5 bases and the other for the last 5 bases. In each table, a non-empty entry is a linked list of barcode indices with the corresponding 5 bases. The total number of required entries becomes 2 * 4 ^ 5 = 2,048. The barcode of a read is recognized if the intersection of the entry in the first table and the entry in the second table is not empty. The intersection can contain at most one barcode when the original barcode list does not include replicate barcodes. The algorithm is still $O(1)$ as long as not many barcodes share the same first 5 bases or the same last 5 bases (which is true in real applications).

When the number of samples is large, dual barcodes can be used. In some platforms, the starting position of the ending barcode may not be certain due to sequencing errors in the ending adaptor part. To take care of this situation, the Burrows-Wheeler transformation (BWT, Burrows and Wheeler, 1994) can be used. To make the algorithm more efficient, we can first make a conservative estimation of the ending part of the read that should contain the ending barcode and then use the BWT only to the ending part.

Alignment of a read to a reference sequence is usually the bottle neck of speed for the whole AVD algorithm. Therefore, it is helpful to first recognize the primers used in a read, and then the algorithm only needs to align the read to a particular reference sequence instead of every reference sequence. One may also first try a quick algorithm that allows a small number of substitutions in primer part, and then try a more complicated algorithm that allows a small indel in primer part.

3. Alignment and Variant Detection

To speed up alignment between a read and a reference sequence, we use an adaptive banded semi-global algorithm. The semi-global alignment is described in Liu et al. (2011). It can give different penalties to the ending indels, and use different initialization methods to fit different purposes. The adaptive banded alignment first limits the horizontal and vertical moves in the semi-global alignment in certain upper band and lower band of the diagonal. If the edit distance of the alignment result is within the predefined bands, the alignment is accepted. Otherwise, the alignment is repeated with increased bands until the results are acceptable. We set the initial upper and lower bands to 6, and the next levels are 12, 30, 60 and full size. Most reads can be successfully

aligned when upper and lower bands are 6. This banded alignment significantly saves time for the AVD algorithm.

After alignment, it is not difficult to get variant calls, counts and proportions for simple mutations defined in Liu et al. (2011). We also implemented computation of the joint counts and frequencies of complex mutations consisting of two or more simple mutations. If all possible combinations of simple mutations are considered, the speed and memory requirement will be impractical. We can first tally the simple mutations, find those satisfying certain abundance condition and then use a hashing function to record the complex mutations contained in a read and count their occurrences in all relevant reads. The joint frequencies of complex mutations does not only generate the statistics of these mutations, but also provide cases for further studying whether they represent alternative targets instead of the originally designed targets.

4. Memory Saving Approaches

Liu et al. (2011) proposed the S3B format to record in information of 3 bases (allowing ambiguous base N) with a byte. Here, we suggest a slim approach to record the alignment results in AVD algorithm. SAM/BAM format is successfully used by many software packages to record the alignment results. For deep sequencing of AVD, we can further save the memory space required to store the alignment results. First, because of the depth, we can save space by recording the reference sequence information only once for many reads associated with the same reference sequence. Second, we can modify the compact idiosyncratic gapped alignment report (CIGAR) format used by SAM/BAM to make it even more compact. Since in AVD applications most frequent pairs in the alignment are matches, it is helpful to record a run of matches more efficiently. We can use a byte pair to record a run of matches of length 1 to 256. We can use 0 in the first byte to denote it as a match (and use other byte code to denote substitutions and indels), and use the second byte with value n in 0 to 255 to denote the run length (n + 1) in 1 to 256. If the run length is larger than 256, we can use multiple byte pairs to denote this situation. For mutations, we can use a non-zero byte to denote 12 possible substitutions, 4 possible insertions and 4 possible deletions. We may also use a byte pair to denote a run of deletions.

5. Usage of Databases

For clinical applications, it is helpful to verify whether a reported variant is a known mutation in a disease-related database such as the Catalogue of Somatic Mutations in Cancer (COSMIC). It is also useful to check the frequency of the mutation in relevant tissue.

6. Algorithm Assessment

To reduce sequencing noise, AVD algorithms usually use filters to remove low quality reads or suppress the report of unreliable variants. To assess an AVD algorithm thoroughly, it is necessary to get the statistics of these filters. We provide comprehensive reports of the filter status for every read and the filter statistics summary for all reads.

Alignment is a critical step in the AVD algorithm. It is important for an algorithm to generate consistent results for the same variant. If an algorithm generates different

alignments for the same variant, it will cause inaccurate computation of variant frequency.

It is also important for an algorithm to report total counts, variant counts and frequencies in both forward and reverse reads. Significantly unbalanced results in forward and reverse reads may indicate unreliable results.

It may be useful to provide a quality score for the variant call.

Some algorithm put a limit of number of bases in reporting indels, and the user should be aware of these restrictions and find ways to overcome these restrictions if necessary.

As pointed out in Liu et al. (2011), the concept of simple mutation including matching bases at both ends of a mutation can help accurately report multi-base indels, e.g., a 15-base deletion should not be reported as a 14-base or 12-base deletions.

The capability to report joint counts and frequencies of complex mutations consisting of multiple simple mutations is also valuable. It is also important to identify whether some reads may belong to an alternative target.

References

Burrows, M. and D. Wheeler (1994). A block sorting lossless data compression algorithm. Technical Report 124. Digital Equipment Corporation

Liu, W., Y. Li,, Y. C. Tai, J. Tsai, M. Christensen, and W. Wen (2011). Notes on algorithms for detection of amplicon variants in next-generation sequencing. In *JSM Proceedings*. Alexandria, VA: American Statistical Association. 4268-4273.