ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 4, Issue 3, pp: (4-10), Month: July - September 2016, Available at: www.researchpublish.com

Alignment Algorithms for RNA-Seq

Anshul Vyas[1], Prof. Varshapriya Jyotinagar[2]

[1]M.Tech student, [2]Professor
[1,2]Department of Computer Engineering and IT , Veermata Jijabai Technological Institute, Mumbai, India.

Abstract: **In bioinformatics field, sequencing technologies evolving rapidly. And it generates so much unclassified and random data. In the process of analyzing this data, most difficult part is, sequence alignment, where we compare sequence reads with a reference. Over the past two years, so many different algorithms and software has been developed for this purpose. One can use cloud computing for that purpose. We can also go for different algorithms for mapping purpose. After study of different algorithms over different data sets, we can conclude that short-read sequences are not bottleneck for data analysis. With the help of long sequence reads and cloud computing, We can develop much faster and accurate algorithms which can be used by different sequence alignment software.**

Keyword: **Next generation sequencing, RNA-sequence, tophat, splice junction, Alignment algorithms.**

I. INTRODUCTION

Advance development in sequencing techniques made so many bioinformatics software and algorithms more efficient. For example genome-wide variation [1] and assembly of new genomes or transcriptome become more easy to use. And most important thing, identification of protein binding sites (coding sequences) and analysis of transcriptome become more accurate, fast and easy. There are so many new technologies available in production houses like Roche/454 and illumina can generate the data in terms of gigabyte base-pairs per machine [2]. AS data is growing with such a exponential way researchers realize that even the fastest algorithms are not efficient for sequencing purpose[3].IN last two years so many new algorithms and tools came into the market to keep the fast pace of result generation. These tools can work with either short base pair length or long base pair length. But output of these tools is highly dependent on type of input. Some new tools show very high compatibility with Solid Helicos reads and short length base pair of illumina. Now days, short read aligners are used for most of the aligner because it takes less time and high accuracy but with lower chance of bottleneck. A systematically review of different advancement in alignment algorithms has been done in this paper. We will discuss all different types of available algorithms in alignment field, their classification on the basis of different techniques and methods.

II. OVERVIEW OF ALIGNMENT ALGORITHMS

There are two type of sequences read sequences and reference sequences. For these references mostly indices are used. Indices are the auxiliary data structures which are constructed by different fast alignment algorithms. These indices have different properties. On the basis of these properties type of algorithm can be divided into three categories. First one is slide [4] and its descendant slider [5].Second one is based on hash table and slicing concept. This is the mostly used algorithm in next sequence generation techniques. And third one, which is also equally important, is based on tree structure.i.e based upon suffix tree and sorting. But this review paper will be focused on only second and third category.

III. ALGORITHM BASED ON HASH TABLE

Hash table algorithms are most used algorithms in rna sequence alignment process. All hash table based algorithms follows seed-and-extend concept. In BLAST, position of each k-mer subsequence holds the each position of k-mer query in hash table and that query will act as the key for that hash table. Then it will scan the whole database sequence for exact matches. This K-mer called seed. BLAST will extend and join all these seeds in two ways. First they will find seeds without gaps and in next step smith-Waterman alignment used to refine [6,7]. It will generate local alignment as a final result. Performance of these hash-table based algorithms are depends upon quality of seed. We can improve the performance by using various seed improvement techniques. In FIGURE:1 performance of different hash table shown.

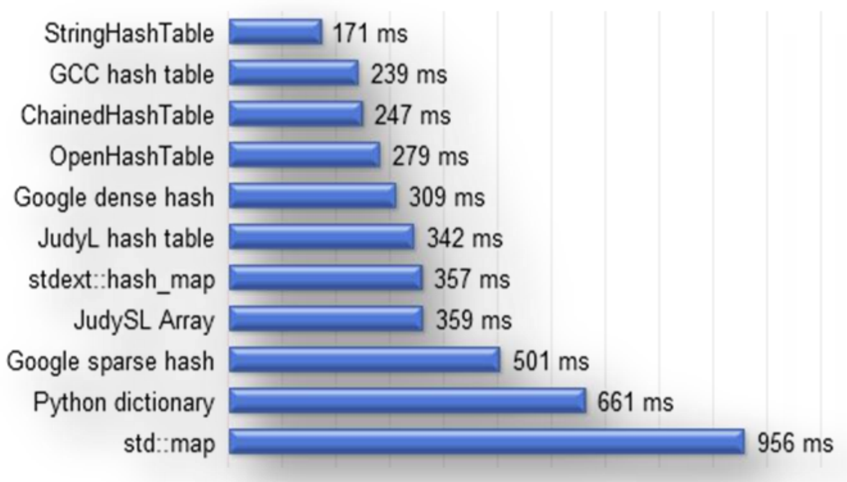

Figure:1

IV. IMPROVEMENT ON SEEDING

A. Spaced Seed

By default, seed alignment used in BLAST matches 11 consecutive places. But research shows that seeding with non-consecutive matches improve sensitivity. For example, if we take a template ‘0001011010110010000’ requiring 11 matches at the ‘0’ positions are more sensitive than default template of BLAST’s default template ‘00000000000’.

Spaced seed means a seed which allows internal mismatches. Another important term in this seed based algorithms is weight. The number of matches in the seed is its weight. Spaced seed can be used in short-read alignment. A software named, Eland, was used for this task. This software uses total six seed templates. These templates used in such a way that for short read , it is guaranteed that if two consecutive or nonconsecutive mismatch occurs, they will be identified. There is only one difference between BLAST and SOAP [8] i.e. BLAST uses short reads whereas SOAP uses reference genome. SeqMap [9] and MAQ [10] both extend the method to allow K-mismatches. For fully responsive k-mismatches hits, It requires no. of templates which are exponentially in K.Clearly, due to exponential growth of K it is insufficient to use for large K values. When it comes to speed, result says that MAQ guarantee two-mismatch hit in the first 28 base-pair of each read. This property makes MAQ the most reliable reads of Illumina. When a seed match is found, it extends the partial match.

Another tool RMAP[11], which is based on Baeza–Yates–Perleberg algorithm [12],uses different type of template as seed. It requires k+1 template to find k mismatches. Thus we can say that RMAP minimize the requirement of templates but create problem when value of K will increase because it will decrease the weight of each template. This technique can’t be useful in hash table indexing.

Lin proposed a simple yet optimized model for designing templates with minimum number of space seeds, appropriate sensitivity requirement, proper use of memory and specific length.

B. Q-gram filter and multiple seed hits

In spaced seed technique gap is not allowed within the seed. BY using dynamic programming, Gaps can be found in the extension step or it can be done by using small gaps at each read positions [13].RazerS and SHRiMP implemented with q-gram filter and can create an index natively allowing gaps.

SSAHA[14] and BLAT[15] use a q-gram index of the target sequence, containing however only the positions of the non-overlapping q-grams.This reduces the index size and the expected number of q-gram seeds by a factor of q, but at the cost of a loss in sensitivity. At query time the set of matching q-gram positions between the query and target sequences is collected and sorted by diagonal. A linear scan locates stretches of hits on identical diagonals. These stretches are then sorted by target sequence position. Another linear scan identifies contiguous stretches of matching positions in the target sequence. These stretches are extended in traditional blast style to produce the final alignments.[16]

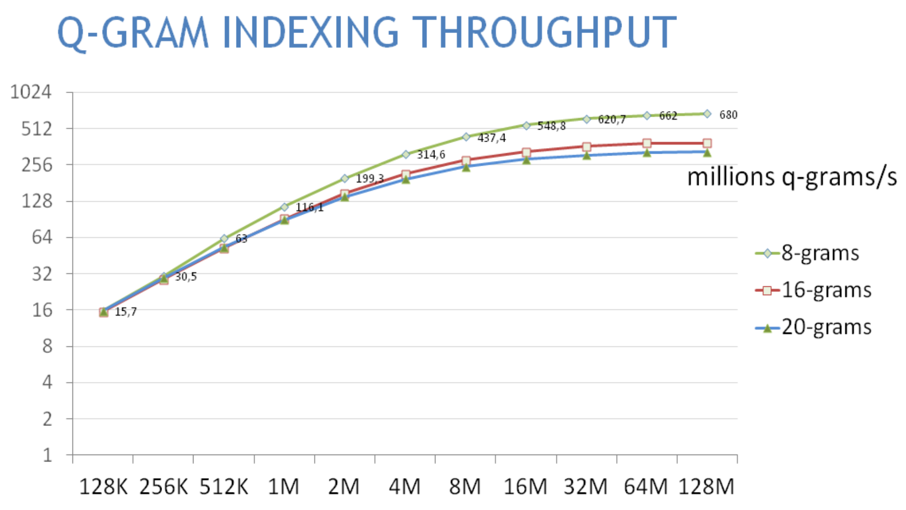

Figure:2

These both methods, spaced seeds and the q-gram filter both works for the fast lookup in hash table. Figure:2 clearly explains that if weightage of q in q-gram is less performance will be more. Even though the purpose of these tool is same but the way of using seeds is different. In spaced seed method, it extends the seed, only after a long match occurs, while after occurrence of multiple short read matches extension in seed happens.

V. ALGORITHM BASED ON SUFFIX AND PREFIX TRIES

There is one major problem in alignment algorithms. They frequently perform inexact matching. These tries based algorithms avoid that type of mismatches and gives the most accurate results. The process of this algorithm is divided into two major steps. In first step it identifies exact matches and then builds inexact alignments supported by exact matches. Certain representation of suffix/prefix trie is used by these algorithms to find exact matches. Example of these tries are suffix tree, FM-index[17] and enhanced suffix array[18].These algorithms are efficient because alignment of multiple copy of a sub string is done only once because next identical identities will be saved in a single path of a trie. And here is the reason that hash table based algorithms are not performed well as compare to trie based algorithm because they align common string each time. A lot of wastage of computing power as well as time also. Data structure can be chosen independently for finding exact matches.

A. Trie, prefix/suffix trie and FM-index

All suffixes of a string are stored in a suffix trie and this is the main reason for fast string matching. Burrows-Wheeler Transform (BWT) is a type of data structure which is used to create a link between a trie and a FM index. Our main concern is prefix trie which is trie of the reverse string.

Performance of an algorithm is considered best if it has a good results for its time complexity. And for this algorithm, time complexity is linearly dependent on the length of input query. It does not matters that what is the length of reference. Space complexity for this algorithm is O (L2) and L is the length of the reference. It is impractical to build a trie even for a bacterial genome. Several data structures are proposed to reduce the space. Among these data structures, a suffix tree is most widely used. It achieves linear space while allowing linear-time searching. Although it is possible in theory to represent a suffix tree in [L(logL) + O(L)] bits using rank selection operations[41].Even the most space efficient implementation of bioinformatics tools requires 12–17 bytes per nucleotide [19], making it impractical to hold the suffix tree of the human genome in memory. For human race we cannot build a trie. It is impractical to build a trie even for a bacterial genome. To reduce the space, so many data structures have been proposed. A suffix tree is most widely used data structure. Reference sequence[20], even the most space efficient implementation of bioinformatics tools requires 12–17 bytes per nucleotide, making it impractical to hold the suffix tree of the human genome in memory. Mohamed Abouelhoda proposed a solution for this. He derived an enhanced suffix array that consists of a suffix array and several auxiliary arrays, taking 6.25 bytes per nucleotide. It can be regarded as an implicit representation of suffix tree, and has an identical time complexity to suffix tree in finding exact matches, better than the suffix array originally invented by Manber and Myers [21].

A further improvement on memory is achieved by Ferragina and Manzini who proposed the M-index and found that locating a child of a parent node in the prefix trie can be done in constant time using a backwards search on this data structure. Thus the time complexity of finding exact matches with an FM-index is identical to that with a trie. With respect to memory, the FM-index was originally designed as a compressed data structure such that the theoretical index size can be smaller than the original string if the string contains repeats (equivalently, has small entropy). The FM-index is usually not compressed for better performance during alignment since DNA sequences have a small alphabet. The practical memory footprint of an FM-index is typically 0.5–2 bytes per nucleotide, depending on implementations and the parameters in use. The index of the entire human genome only takes 2–8 GB of memory.It is worth noting that we only focus on the data structures having been used for DNA sequence alignment. There is a large volume of literature in computer science on general theory of string matching, especially on short string matching.

B. Prefix/suffix trie used to find inexact matches:

There are two aligners which are based on suffix tree MUMmer and OASIS. Some are based on enhanced suffix tree Match and Segemehl and finally third category is based on FM index Bowtie, SOAP2 , BWA , BWT-SW and BWA-SW on FM-index.

A program which is built upon a specific suffix/prefix trie can be migrated or shifted to other suffix/prefix trie.Only FM index is the algorithm which takes very less footprint as compare to others. MUMmer and Vmatch generate maximum matches. Similarly, Segemehl initiates the alignment with the longest prefix match of each suffix, but it may also enumerate mismatches and gaps at certain positions of the query to reduce false alignments.OASIS and BWT-SW essentially sample substrings of the reference by a top–down traversal on the trie and align these substrings against the query by dynamic programming.

BWA-SW and BWT-SW represents all query as a directed word graph (DAWG), this representation accelerates the occurrence of exact matching. Bowtie and BWA also sample short substrings of the reference, but instead of performing dynamic programming, they compare the query and sampled substrings only allowing a few differences. Bowtie and BWA can be considered to find out all possible combinations of different mismatches and remaining gaps in the present query sequence such that the altered query can be aligned exactly.

VI. ALLIGNMENT BASED UPON DIFFERENT TYPES OF READ

a) Using Base Quality In Alignment

b) Aligning Long Sequences Reads

c) Aligning Solid Reads

a) Using Base Quality In Alignment:

Smith discovered that if base quality will be used in the alignment, accuracy of the alignment can be increased. As probability of error for each base will be known. In such case aligner will assign lower penalty for those mismatch that have high probability of mismatch.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 4, Issue 3, pp: (4-10), Month: July - September 2016, Available at: www.researchpublish.com

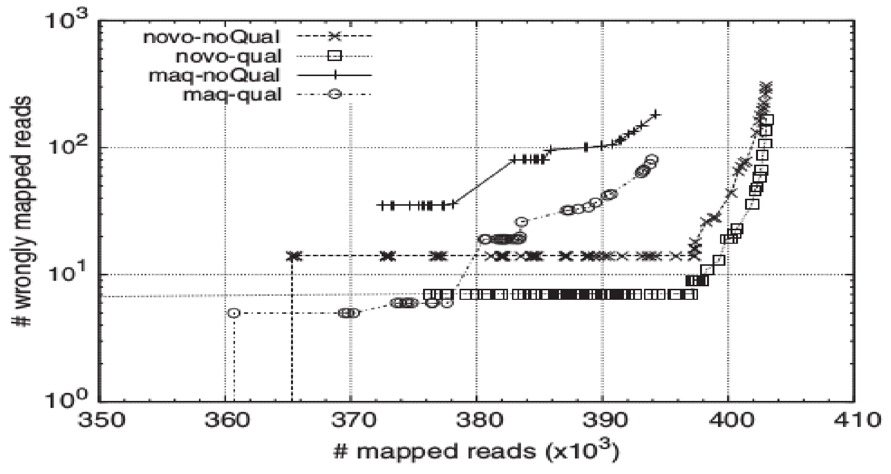

FIGURE:3

Figure 3 shows that using base quality score halves alignment errors when the quality score is accurate. In practice, however, accurate quality score is not always available from the base calling pipeline. Recalibration of quality score is recommended to make this strategy more effective.

b) Aligning long sequences reads

In the reference genome long reads can produce more appropriate result as compare to short reads. Currently, all available too all programs capable of genome-wide long-read alignment follow the seed-and-extend paradigm, seeding the alignment using hash table index [8, 9] or more recently FM-index, and extending seed matches with the banded Smith–Waterman algorithm. This allows for sensitive detection of indels as well as allowing for partial hits.

c) Aligning solid reads

This technique, called SOLID sequencing technique, identifies two consecutive bases at a time. Each and every dinucleotide can be assigned or encoded as one of four types of color.This encoding referred as a colour space(Figure 4A). A known base always allows decoding a colour read to bases (Figure 4B).

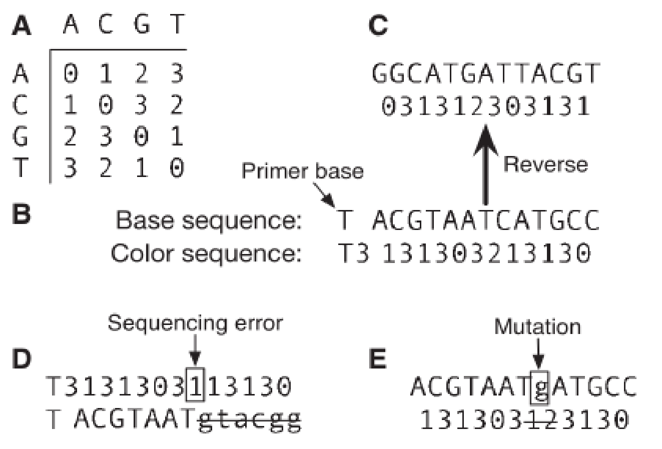

Figure 4: Color-space encoding. **(A)** Color space encoding matrix. **(B)** Conversion between base and color sequence. **(C)** The color encoding of the reverse complement of the base sequence is the reverse of the color sequence. **(D)** A sequencing error leads to contiguous errors when the color sequence is converted to base sequence. **(E)** A mutation causes two contiguous color changes.

Figure 4D,explained that if someone using only one colour than few contiguous errors will arise due to the use of that single colour. Given the fact that reverse complementing a base sequence is equivalent to reversing the colour sequence (Figure 4C), the proper solution is to encode the reference as a colour sequence and align colour reads directly to the colour reference as if they are base sequences with the exception of the complementing rule. By using dynamic programming, Base sequences can be taken as output by converting colour sequence [48].Although alignment done in the colour sequences may or may not be optimal. Performing alignment entirely in the colour space may not be optimal, though. With colour encoding, one base mutation leads to two contiguous colour changes with some restrictions (Figure 4E). Over two discontinuous changes are preferred over two adjacent same colours. To handle this confusing situation, special colour oriented algorithm should be used. Colour aware algorithm like Smith–Waterman alignment can be used. Some other colour oriented algorithms are used in BFAST and SHRiMP. By using this Smith-Waterman algorithm which is based on colour scheme can be used for post alignment analysis. Approx all algorithm can be used with SOLID technique.

VII. CONCLUSION

As traditional perception about alignment is, if short reads are used it will generate bottleneck in performance for new sequence data. But after a lot of research we can say that by using new and different algorithms, we can remove this bottleneck problem from the alignment process. But different researcher says that in upcoming years algorithms will be designed and implemented in such way so that use of long reads will be back in use. Short reads are not fully efficient for alignment. There are few things which better in in long reads also. Researches are trying to apply these existing algorithms to make log read more efficient, But till then short reads giving better result as compare to long read .After researching about various alignment algorithms. We have two option to increase the performance of a alignment software either we can use cloud computing for better resource utilization or we can choose basic algorithm of a software to improve its performance by trying with different existing algorithms.

REFERENCES

[1] Dalca AV, Brudno M. Genome variation discovery with high-throughput sequencing data. Brief Bioinform 2010;11:3–14.

[2] Metzker ML. Sequencing technologies – the next generation. Nat Rev Genet 2010;11:31–46.

[3] Ning Z, Cox AJ, Mullikin JC. SSAHA: a fast search method for large DNA databases. Genome Res 2001;11:1725–9.

[4] Malhis N, Butterfield YSN, Ester M, et al. Slider–maximum use of probability information for alignment of short sequence reads and SNP detection. Bioinformatics 2009;25:6–13.

[5] Malhis N, Jones SJ. High quality SNP calling using Illumina data at shallow coverage. Bioinformatics 2010;26:1029–35.

[6] Smith TF, Waterman MS. Identification of common molecular subsequences. J Mol Biol 1981;147:195–7.

[7] Gotoh O. An improved algorithm for matching biological sequences. J Mol Biol 1982;162:705–8.

[8] Li R, Li Y, Kristiansen K, etal. SOAP: short oligonucleotide alignment program. Bioinformatics 2008;24:713–4

[9] Jiang H, Wong WH. SeqMap: mapping massive amount of oligonucleotides to the genome. Bioinformatics 2008;24:2395–6.

[10] Li H, Ruan J, Durbin R. Mapping short DNA sequencing reads and calling variants using mapping quality scores.Genome Res 2008;18:1851–8.

[11] Smith AD, Xuan Z, Zhang MQ. Using quality scores and longer reads improves accuracy of Solexa read mapping.BMC Bioinformatics 2008;9:128

[12] Baeza-Yates RA, Perleberg CH. Fast and practical approximate string matching. In: Apostolico A, Crochemore M,Galil Z, Manber U, (eds). CPM, Lecture Notes in Computer Science, Vol. 644. Berlin: Springer, 1992:185–92

[13] Li R, Li Y, Kristiansen K, etal. SOAP: short oligonucleotide alignment program. Bioinformatics 2008;24:713–4.

[14] Ma B, Tromp J, Li M. PatternHunter: faster and more sensitive homology search. Bioinformatics 2002;18:440–5.

[15] Malhis N, Butterfield YSN, Ester M, et al. Slider–maximum use of probability information for alignment of short sequence reads and SNP detection. Bioinformatics 2009;25:6–13.

[16] Kim R. Rasmussen and Eugene W. Myers. Retrieved from http://www.cs.toronto.edu/~brudno/csc2431/qgramfilter.html

[17] Ferragina P, Manzini G. Opportunistic data structures with applications. In: Proceedings of the 41st Symposium on Foundations of Computer Science (FOCS 2000), Redondo Beach, CA, USA. 2000;390–8.

[18] Abouelhoda MI, Kurtz S, Ohlebusch E. Replacing suffix trees with enhanced suffix arrays. J Discrete Algorithms 2004;2:53–86.

[19] Kurtz S, Phillippy A, Delcher AL, et al. Versatile and open software for comparing large genomes. Genome Biol 2004;5:R12.

[20] Munro JI, Raman V, Rao SS. Space efficient suffix trees. J Algorithms 2001;39(2):205–22.