DOI 10.29042/2018-3686-3691

Helix Vol. 8(5): 3686- 3691

Exact Tandem Repeats using Suffix Array and Longest Common Prefix

Dr Raju Bhukya, Naveen.I, Rohan Gupta, Anurag.K, Achyuth.A, Taruni.
National Institute of Technology, Warangal.
Email: drrajunitw@gmail.com

Received: 22nd June 2018, Accepted: 01st August 2018, Published: 31st August 2018

Abstract
Tandem Repeats (TRs) are the repeats which occur in a chromosome, while a pattern of 1 or more nucleotides (A, C, G, T) is repeated more than one time and the repetitions are continuous. Repeating patterns also referred to as motifs or base patterns, can be of various lengths. There is a complete of 25 linkage groups or chromosomes in the Zebrafish genome. The proposed algorithm reported exact tandem repeats with a motif length of at least three 3 with the help of an algorithm that uses a suffix array and longest common prefix. An overall of 119,909 repeats, with a base pattern of at least 3 nucleotides were detected. A massive number of repeats with a base period of 18, 24, 27 30 had been detected. The algorithm produces better results as compared to the existing results in terms of a number of repeats.

Keywords: Tandem Repeats, Chromosome, Nucleotide, Genome, Linkage Group, Motif.

Introduction

Genomes are comprised of nucleotides which are represented by A, C, G, T. The zebra fish genome is large, it is approximately 1.7 billion bases lengthy with a complete of 25 linkage groups [1], that is more or less half the dimensions of the human genome. Single nucleotide polymorphisms rate inside the zebra fish genome is approximately one in 2 hundred bases, while in human beings it is one in keeping with 1,000 bases[2]. Depending on the length of repeating motif DNA repeats are classified into subtypes. Patterns or motifs of length less than 10 are known as microsatellites or short tandem repeats (STR) [3] and repeats of longer lengths for example 1000 nucleotides are referred to as minisatellites [4]. Zebra fish are studied as a model vertebrate organism which is an animal of a large group distinguished by the possession of a backbone with an early embryonic development which is similar to the development of a human[5]. As compared to Zebra fish genome this is 1.7 billion bases lengthy, the human genome is 3.2 billion bases long and this equates to 3 million variations throughout the human genome and eight million variations across the zebrafish genome. These patterns are characteristic of protein-coding sequences. Duplication happens in large regions of a genome and this results in the repetition of a pattern. For example, if a pattern ACG is repeating 10times then the resulting sequence will beACGACGACGACGACGACGACGACGACG this can be written as(ACG)10 where ACG is called as the base pattern and 10 is the copy number means the pattern is repeating 10 times [6]. To detect and report these highly degenerate sequence motifs from genomic sequence collections we need more efficient tools. Thus, it is very important to identify and characterize repeating sequences in both the genome assembly and genomic sequences including weak or decentralized repeating sequences. For detecting repeats various computational approaches are developed. These approaches have some limitations which can be categorized into deterministic and probabilistic. Other than these, there are three main groups [7]. The first group detects and reports repeats indirectly and it is based on the alignment of matrices. The limitation of these algorithms is time complexity with the best algorithm having a time complexity of O(N2logN)[7]. The second group detects the repeats without any alignment. This kind of algorithms either assume some kind of similarity between possible repeats or they require some kind of input from the user. For example, the input may consist of predefined threshold on similarity, the number of nucleotides (A, C, G, T) between repeats, motif lengths[7].

Related Work

Many algorithms have been discovered till now to find tandem repeats in genome sequences. Based on their approach, there are two ways to categorize them – deterministic algorithms and probabilistic algorithms. The iterative method of Coward and Drablos uses the self-alignment property of sequence to find repeats [8], whereas Benson finds repeats probabilistically in the same problem [9]. Coward and Drablos algorithm finds approximate tandem repeats. This method is practical but it doesn't give all the repeats given a maximum number of mismatches Whereas Benson's probabilistic approach design and implementation are more complicated. Benson used his probabilistic model in the tool tandem repeat finder [9]. The idea is based on the percent identity that is the percent of matching probability and frequency of indels that is insertion and deletion in the sequence, between adjacent copies of the repeats. This algorithm is better

than many other algorithms. There are many algorithms to find exact tandem repeats in genome sequences. Algorithms were developed to find squares in a string [10-12] and these cannot be directly adapted to find tandem repeats in genome sequences. Some algorithms use hamming distance[13] and some use insertion and deletion as a measure of similarity [14]. These algorithms come in two phases, in the first phase that is scanning phase candidate tandem repeats are located and then in the second phase that is analysis phase candidate tandem repeats are checked to be the tandem repeat. Hauth and Joseph designed an algorithm to find NTRs with the constraint of length at most 6 nucleotides [14]. It uses the space-efficient data structure the enhanced suffix array. There are many algorithms like mreps, iMEx, inverter, Sputnik, TRF. All these algorithms work differently with different input parameters and constraints. Suffix array can replace the bottom up traversal of a suffix tree. Abouelhoda et al. proposed an enhanced suffix array algorithm. It is able to compute all branching tandem repeats in O (n log n) time. Time complexity plays a major role while comparing algorithms. Manber and Myers implementation creates a suffix array in optimal time [15]. This suffix array and longest common prefix array can help in finding all tandem repeats. Kasai, T.; Lee, G.; Arimura, H.; Arikawa, S.; Park, K gave an efficient implementation of constructing longest common prefix array[16]. First combinatorial approach [17,18] for finding tandem repeats was developed in sputnik tool and it was based on repeated motif length setting the resolution parameter to zero and in TRF we cansetsimilarityto100percentbysettingpercentindel to be zero. In other tools, similarity can be maximized by setting mismatch, insertion, deletion accordingly. To improve the various algorithms time complexity and space complexity researches have spent many years. In TRF, we can set this by setting the alignment weights to the extreme values of 2, 50 and 50 for matching score and mismatch penalty and indelspenalty respectively and the minimal score may be driven down to 0, permitting reporting of all repeats detected indiscriminately. Similarly, mismatch penalty, match score, minimum score and indelpenalty for ATR Hunter were set to 50, 1, 50, and -50 respectively [19].

Proposed Work

The proposed method works on two techniques.

- Suffix Array.
- Longest Common Prefix Array.

The algorithm first Pre-processes the DNA sequence then it calculates the suffix array, next it calculates the longest common prefix array using the suffix array and at last, it calculates the three factors(motif length, frequency, starting position).In the pre-processing step of the algorithm, the header and symbol 'N' from the fasta file are removed so that the modified input file contains only A, C, G, T symbols. The algorithm first creates a suffix array (SA) for the pre-processed DNA sequence. After creating the suffix array, it creates a Longest Common Prefix Array(LCPA) by using the suffix array of the pre-processed DNA sequence. Suffix array can always be constructed in linear time O(n) using Manber and Myers approach[15] for any given sequence, where n represents the sequence length. The suffix array is an integer array that provides the starting position of the suffix if arranged in lexicographical order. Manber and Myers [15] introduced the suffix array first time, to improve over the suffix trees: Suffix array contains M integers for M length string or text. If integer type size is 4 bytes then a suffix array need 4*M bytes memory. LCPA Integer array consists of a number of common prefix between two consecutive suffixes according to the suffix array.

Algorithm:

Input: DNA sequences, minimum length, maximum length, minimum repeat count.

Output: Starting Position of repeat, Repeat count, Repeat length.

Step1: Create Integer arrays SP[1..n],RL[1..n],RC[1..n],SA[1..n],LCPA[1..n].

n = length of sequence.

Step 2: Remove 'N' characters and header part from fasta files.

Step 3: Create Suffix Array SA[1..n] of preprocessed DNA sequence.

Step 4: Create Longest Common Prefix Array, LCPA[1..n] of pre-processed DNA sequence.

Step 5: FOR i =1..n do

```
SP[i] = Minimum(SA[i],SA[i-1])
RL[i] = SA[i]–SA[i-1]
RC[i] = Floor(LCPA[i]/RL[i])
// Floor isafunction of C
END FOR
```

Step 6: Store repeat starting position, length, repeat count in output file.

END

Implementation

First symbol 'N' and header of fasta file were removed from the input DNA sequence. After that suffix array was created. Suffix array is an integer array that provides the starting position of suffix ifarranged in lexicographical order. Suffix array can always be constructed in linear time O(n) using Manber and Myers approach [15] for any given sequence, where nrepresentsthesequencelength.After this longest common prefix array was created. Longest common

prefix array contains a number of commonprefixes between two consecutive suffixes according tosuffix array. SP array is used to store starting position of repeat. RL array is used to store repeat length. RC array is used to store repeat count. An efficient implementation can be done by using Kasai et al approach [16]. Construction of LCPA from Suffix array also takes linear time O(n). Dynamic arrays were used in the construction of suffix array and longest common prefix array. Thealgorithm is coded or implemented in c language. Three arrays, of length equal to sequence lengths, are created to store Motif Length, Number of Repeats and Starting Position of the repeats. All these arrays are created in linear time O(n) and overall time complexity of this algorithm is O(n).

In the proposed technique we are calculating differenceof the adjacent value of SA which gives us a unit motif length. Repeating frequency is calculated by taking the floor of LCPA[i] to ML[i]. Starting position is determined by taking the minimum of two adjacent SA values. After getting these parameters motif can be found by starting from starting position and adding motif length to it. In case of memory constraints, file has to be processed in small chunks while the output file is created for storing repeat starting position, repeat length and repeat count or copy number.

Example

Let us consider finding number of exact tandem repeats in string "banana". For this string suffix array and longest common prefix array can be shown as in the Table.1. Suffixes of "banana" are – banana, anana, nana, ana, na, a. In the below Table.1, suffix array row, represents the indexes of the suffixes in the string. So suffix array in terms of suffixes can be shown asa, ana, anana, banana, na, nana.

Index	0	1	2	3	4	5
String	b	a	n	a	n	a
Suffix Array	5	3	1	0	4	2
LCP Array	1	3	0	0	2	0

Table 1: Suffix Array and LCP array For "banana"

Now suppose if we take two suffixes "ana" and "anana" whose indexes are 3 and 1 respectively. Longest common prefix between suffixes starting at 3 and 1 is 3 as can be seen in Table 1.

Step 1: The starting position of repeat is minimum of 1 and 3 which are starting position of "anana" and "ana". So starting position of repeat is 1.

Step 2: Repeat length is difference between the lengths of these two suffixes. So repeat length is = 3-1 =2.

Step 3: Repeat count is longest common prefix divided by repeatlengthadd1. So repeat count=floor (3/2)+1= 2.

Step 4: So we get a repeat starting at position 1 whose length is 2 is repeating 2 times. This repeat is"an". Similarly, other repeats can be found. Other repeats are "na" starting from2 and repeating 2 times that can be found easily with this algorithm.

Discussion and Results

In this section we present several experiments comparing our algorithm to the existing algorithms and evaluating with the number of repeats. Experiments were performed on various sequence sizes. The implementation was tested on Intel Core i5-5250U CPU at 1.60GHz4 processor with 4GB RAM. The Algorithm was run for all chromosomes of Zebrafish(Chromosome 1 to Chromosome 25). The results obtained with the number of repeats and the corresponding runtime is shown in Table2. From the below table.2 it can be seen that the wide variety of repeats will increase with the growth inside the DNA size. In case of memory constraints, the sequence has to be divided into chunks and processed. The proposed algorithm (ESL) is compared with the existing algorithm (SWA) in terms of number of repeats. The proposed technique shows better performance when compared with the existing technique. In addition to the existing technique we have calculated runtime for the proposed technique. Table.2 shows the repeats found by SWA and ESL.

Chr	Seq Size (MB)	Repeat ESL	Repeat SWA	Runtime ESL	Diff	Chr	Seq Size (MB)	Repeat ESL	Repeat SWA	Runtime ESL	Diff
1	60.3	5263	5176	69.46	87	13	51.6	3852	3743	58.86	109
2	59	3880	3852	67.87	28	14	53.8	4558	4466	61.54	92
3	61.9	4209	4205	71.41	4	15	48	3499	3496	54.47	3
4	72.9	3544	3411	84.81	133	16	52.8	3891	3803	60.32	88
5	75.7	4883	4693	88.22	190	17	50.3	3617	3617	57.27	0
6	62.7	4819	4687	72.38	132	18	50.1	4161	4154	57.03	7
7	78.2	6121	5884	91.26	237	19	49.5	4131	3895	56.30	236
8	56.5	3966	3911	64.83	55	20	52.7	3509	3507	60.20	2
9	55.6	4118	4109	63.73	9	21	48.4	3430	3345	55.26	85
10	44.2	3475	3382	49.84	93	22	42.1	2473	2355	47.55	118

11	44.9	3692	3625	50.70	67	23	45.5	3011	2934	51.43	77
12	47.6	3994	3925	53.98	69	24	41.1	3731	3384	46.3	347

Table 2: Runtime of the Algorithm for all Chromosomes (Chr) and Difference (diff).

In this technique we had found the repeats length from 3 to 50 in Zebrafish chromosomes. The following Table.3 shows the frequency of tandem repeats by the base length Zebrafish genome Zv8 assembly. Table 3 shows that repeats with base length 18, 24, 27, 30 have significant number of repeats as compared to other base length. Repeats with base length 3, 4 and 5 are abundant in zebra fish genome.

Motif Length	Repeats SWA	Repeats ESL	Motif Length	Repeats SWA	Repeats ESL	Motif Length	Repeats SWA	Repeats ESL
3	37383	38028	19	5	5	35	2	2
4	67313	69111	20	6	6	36	9	9
5	11767	12355	21	7	7	37	3	3
6	93	102	22	9	9	38	1	1
7	1	1	23	7	7	39	2	2
8	10	10	24	18	18	40	0	0
9	1	1	25	4	4	41	2	2
10	5	5	26	7	8	42	0	0
11	5	5	27	40	42	43	1	1
12	6	6	28	5	5	44	2	2
13	3	3	29	4	4	45	2	2
14	16	18	30	117	120	46	5	5
15	6	6	31	10	10	47	0	0
16	16	18	32	5	5	48	1	1
17	5	6	33	1	1	49	1	1
18	21	24	34	2	2	50	1	1

Table.3: Frequency Comparison of Proposed and Existing Output by Base Length Zebrafish Genome (Zv8assembly)

Table.4 shows the difference between the repeats already found with the repeats found with the help of our proposed algorithm for linkage group 22 of zebra fish. We compared our algorithm output with sliding window approach algorithm. Four fields are represented in the results, starting position of the motif, motif length, base pattern and copy number of the base pattern. The one represented in bold symbols are new repeats found other than the existing. A web interface has been developed to get repeats of a user-specified length for all linkage groups 1 to 25.

Starting Position	Repeat Length	Base Pattern	Copy Number	Starting Position	Repeat Length	Base Pattern	Copy Number
14772221	3	TAG	14	20841905	3	TAT	15
@14772335	**3**	**TAA**	**10**	**@20842215**	**3**	**ATT**	**10**
14772484	3	TAA	10	20842243	3	TTG	12
15222502	3	CAT	16	15050352	5	TATAT	11
@15364592	**3**	**TAA**	**11**	**@15535739**	**5**	**TTATA**	**20**
@15373855	**3**	**TAT**	**16**	**@15548099**	**5**	**TTATA**	**14**
@15375793	**3**	**AAT**	**24**	**@15661029**	**5**	**AATAA**	**16**
@15375942	**3**	**TAA**	**17**	**@15664369**	**5**	**AATAT**	**13**
@15391038	**3**	**TTC**	**11**	16252212	5	ATAAT	19
@15393514	**3**	**TTA**	**10**	16488891	5	ATTAT	16
@15394889	**3**	**TTC**	**11**	39505117	5	AATAT	15
@15397180	**3**	**TTC**	**15**	39527858	5	TATAT	17

(Note:@ denotes the new repeats found using the proposed algorithm).

Table 4: Result for Chromosome 22 of Z8 Assembly

The Figure.1 shows the comparisons with existing algorithm. Red bar is for ESL and blue bar is for SWA which is the existing algorithm. Figure.2 shows the comparison of proposed algorithm with TRF and mreps. Green bar shows the repeats found from ESL (Proposed Algorithm), purple bar shows the repeats found from mreps and orange bar shows the repeats found from TRF algorithms.

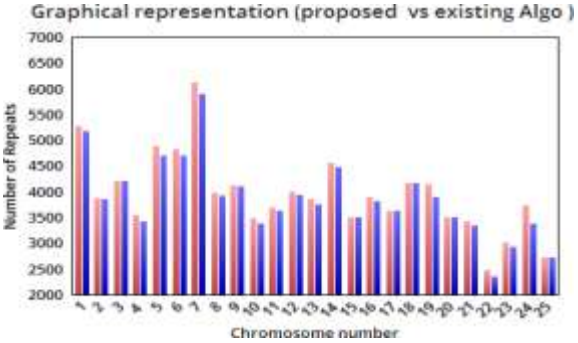

Figure 1: Results comparison of SWA and ESL

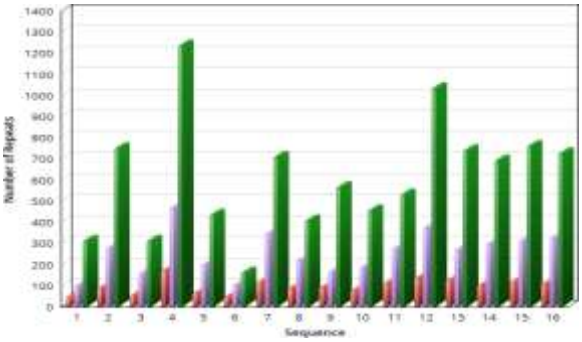

Figure 2: Results Comparisons

Sequence	Seq. Size(bp)	TRF	Mreps	ESL	Sequence	Seq. Size(bp)	TRF	Mreps	ESL
NC 001133.8	230218	46	96	291	NC 001140.5	562643	85	214	396
NC 001134.8	813184	86	275	742	NC 001141.1	439888	90	161	561
NC 001135.4	316620	52	153	311	NC 001142.8	745751	77	180	447
NC 001136.9	1531933	168	464	1234	NC 001143.8	666816	110	270	507
NC 001137.2	576874	62	194	431	NC 001144.4	1078177	132	368	1020
NC 001138.4	270161	46	100	156	NC 001145.2	924431	123	268	730
NC 001139.8	1090940	112	343	698	NC 001146.7	784333	98	293	687

Table 5: Comparison of repeats when standard input is given (ESL = Proposed Algorithm)

The above graph shows that ESL has compared with all the other existing tools. We compared our algorithm with other existing tools to find exact tandem repeats and results are indexed in Table.5. S.cerevisiae S288c genome dataset was obtained from the NCBI website for Chromosome 1 to 14 respectively. As a reference dataset Saccharomyces cerevisiae S288c genome was used and results were indexed. The Chromosome's accession numbers are given in the below table.5. The algorithm finds all the repeats with base pattern length 3and copy number at-least 2 in the sequence and compares with other existing tools for exact tandem repeats. For TRF the parameters were set to, match score = 2, mismatch penalty = -7, indel penalty = -7, minimum alignment score = 50, maximum size = 500, and results were filtered. For mreps period size was set to 3 and exponent was set to 2. Experimental results are shown in the below table.5 which are existing algorithms like TRF, Mreps with the proposed technique ESL. In some cases when copy number or base size is more mreps shows better performance as compared to ESL and other tools.

Conclusion and Future Work

We proposed a method based on Suffix Array and Longest common prefix array techniques to find motif length, frequency and stating position of that motif. Suffix array can always be constructed in linear time O(n) for any given sequence, where n represents the sequence length. Construction of LCPA from Suffix array also takes linear time O(n) overall time complexity of the proposed algorithm is O(n). The proposed algorithm (ESL) is compared with the existing algorithm (SWA) in terms of number of repeats. The proposed technique shows better performance when compared with the existing techniques. In this technique we had found the repeats length from 3 to 50 in Zebrafish chromosomes. Our future work is to apply the algorithm for all the species with better complexity.

References

[1] Zebrafish Sequencing Project [http://www. sanger. ac.uk\\/Projects/Drerio/].
[2] Stickney HL, Schmutz J, Woods IG, Holtzer CC,

Dickson MC, Kelly PD, Myers RM. Talbot WS: Rapid mapping of zebrafish mutations with SNPs and oligonucleotide microarrays. Genome Res 2002, pp.1929-1934.
[3] H. Ellegren Microsatellites: simple sequences with complex evolution Nat. Rev. Genet., 2004, pp. 435-445.
[4] F.Denoeud, G. Vergnaud, G. Benson. Predicting human mini-satellite polymorphism Genome Res., 2003, pp. 856-867.
[5] Tandem Repeat at the US National Library of Medicine Medical Subject Headings (MeSH).
[6] Eric C Rouchka, Database of exact tandem repeats in a Zebrafish genome BMC Genomics. 2010,11: 347.
[7] Ilya Shmulevich, John D. Aitchison Methods Enzymol. Deterministic and stochastic models of genetic regulatory networks. Author manuscript;PMC 2011.
[8] Coward E, Drablos F. Detecting periodic patterns in biological sequences. Bioinformatics. 1998, pp. 498-507.
[9] Benson G. Tandem repeats finder: a program to analyze DNA sequences. Nucleic Acids Res. 1999. pp. 573-80.
[10] Apostolico A, Preparata FP. Optimal offline detection of repetitions in a string, Theor. Comput. Sci. , 1983, pp. 297-315.
[11] Crochemore M. An optimal algorithm for computing the repetitions in a word, Inf. Process. Lett. ,1981, pp. 244-250.
[12] Kolpakov R, Kucherov G, Logiciel TG. Finding approximate repetitions under hamming distance, Theoretical computer science, Springer-2001.pp. 170-181.
[13] Landau GM, Schmidt JP, Sokol D. An algorithm for approximate tandem repeats, J.Comput.. Biol. ,2001, pp. 1-18.
[14] Hauth AM, Joseph D. Beyond tandem repeats: complex pattern structures and distant regions of similarity, ISMB, 2002, pp. 31-37.
[15] Manber U., Myers G. Suffix arrays: a new method for on-line string searches. SIAM J. Comput., 1993, pp. 935– 948.
[16] Kasai, T.; Lee, G.; Arimura, H.; Arikawa, S.; Park, K. Linear-Time Longest-Common-Prefix Computation in Suffix Arrays and Its Applications.2001. pp.181-192.
[17] Kolpakov R, Bana B, Kucherov G. Mreps: efficient and flexible detection of tandem repeats in DNA. Nucleic Acids Res 2003;3672–8.
[18] Mudunuri SB, Nagarajaram HA. IMEx: imperfect microsatellite extractor. Bioinformatics 2007, 1181–7.
[19] Kian Guan Lim, Chee Keong Kwoh, Li Yang Hsu, Adrianto Wirawan; Review of tandem repeat search tools: a systematic approach to evaluating algorithmic performance, Briefings in Bioinformatics,2013, pp.67–81.
[20] Ensembl public FTP site [ftp://ftp.ensembl.org/pub\\/assembly/zebrafish/Zv8release/].

  ISSN 2319 – 5592 (Online)