AUT Journal of Mathematics and Computing

AUT J. Math. Comput., 1(2) (2020) 165-170
DOI: 10.22060/ajmc.2019.15991.1020

A new hash function and its use in read mapping on genome

Farzaneh Salari[a], Fatemeh Zare-Mirakabad*[b], Mehdi Sadeghi[c]

[a] *Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran*
[b] *Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran*
[c] *National Institute of Genetic Engineering and Biotechnology, Tehran, Iran*

ABSTRACT:

Mapping reads onto genomes is an indispensable step in sequencing data analysis. A widely used method to speed up mapping is to index a genome by a hash table, in which genomic positions of k-mers are stored in the table. The hash table size increases exponentially with the k-mer length and thus the traditional hash function is not appropriate for a k-mer as long as a read. We present a hashing mechanism by two functions named *score*1 and *score*2 which can hash sequences with the length of reads. The size of hash table is directly proportional to the genome size, which is absolutely lower than that of hash table built by the conventional hash function. We evaluate our hashing system by developing a read mapper and running the mapper on *E. coli* genome with some simulated data sets. The results show that the high percentage of simulated reads can be mapped to correct locations on the genome.

Review History:

Received:13 March 2019
Revised:13 July 2019
Accepted:16 July 2019
Available Online:01 September 2020

Keywords:

Read mapping
Reference-based assembly
Hash function

AMS Subject classifications:

35Q68; 35Q93

1. Introduction

The information directing the behavior of cells is encoded in genomes. Accordingly, genomic information from bacteria to humans is an important component of biological research. The genome sequence has to be determined to analyze the genome of an organism. There are different sequencing technologies in which the DNA of an organism is broken up into millions of small fragments that are then sequenced as reads. In the past decade, with the introduction of next-generation sequencing (NGS) technologies, the volume of generated short reads has increased sharply while the cost has decreased. To obtain a genome sequence, NGS reads have to be assembled by a computer program. There are two approaches for genome assembly: de novo and reference-base assembly. In the first method, the reads are merged together without any knowledge of the original genome. In the second one, the reads are put together by aligning to a reference genome.

Several algorithms have been developed for reference-based assembly which can be divided into two main categories: FM-index and hash-based. At FM-index mappers including Bowtie [4], Bowtie2 [3] and BWA [6] the construction of index is memory-efficient but time-consuming. On the other hands, hash-based mappers such as RMAP [8], SHRiMP [7], BFAST [2], Hobbes [1], Mozaik [5] and FEM [10] make hash tables quickly but storing the tables are memory-demanding. Hash-based mappers create a hash table with substrings of length k called k-mers extracted from the reference genome as keys and their positions on the reference as values. Then, some k-mers of each read are looked up in the hash table to find candidate locations of the read on the reference (hits). Afterward, each read is aligned to its hits by an alignment algorithm such as Smith-Waterman [9] in order to obtain its origins.

**Corresponding author.*
E-mail addresses: farzsalari@gmail.com, f.zare@aut.ac.ir, sadeghi@nigeb.ac.ir

The length of k-mer is very critical, because the larger k implies the smaller number of hits for each read and thus decreases the number of required alignments. In other words, if k equals the length of reads then there is no need to align reads to the reference genome. Most of the hash-based mappers are not able to consider k equals to the read size since these mappers usually use a simple hash function transforming a DNA sequence of length k to a binary sequence of length $2k$ bits, where each nucleotide is represented by two bits. Thus for $k > 16$, the hash table becomes extremely large.

In this paper, we would like to design a hash function on L-mers, in which L is the length of reads to find genomic positions of a read in one step. Therefore we present a double hashing system which encodes a DNA sequence to a small integer value with two hash functions paradigm collisions resolution such that storing the hash table on RAM is memory-efficient.

2. Methods

Each fragment of DNA such as r is denoted by $r = r_1 \dots r_k$ where $r_i \in \{A, C, G, T\}$. The length of r is denoted by $|r|$. Typically, the sequence r can be mapped to a number based on function h:

$$h(r) = \sum_{i=0}^{|r|-1} f(r_i) \cdot 4^i, \tag{1}$$

where $f(A) = 0, f(C) = 1, f(G) = 2$ and $f(T) = 3$. Note that for each two sequences r and r' if $r \neq r'$, then $h(r) \neq h(r')$. Since h is a one-to-one function, it can be used as a hash function with no collision. To make an index from a genome sequence by the function h, a hash table with the size of $4^{|r|}$ is required. Thus, making index by the function h is not appropriate for hashing genome sequences with read lengths of ranging from 21 to 150 (base-pair) bp because the hash table grows exponentially with $|r|$ and consequently becomes very large for $|r| > 16$. To make a memory-efficient hash table, we define the following function:

$$g(r) = \sum_{i=1}^{|r|} f'(r_i) \cdot i^p, \tag{2}$$

where f' is a function that encodes each nucleotide to a natural number and p is a constant greater than zero. By identifying appropriate function f' and value of p, the function g can be employed as a hash function. To do so, we consider a sequence s containing identical nucleotides in which $|s| = |r|$. Let the value of f' on the mentioned nucleotide be the largest one denoted as z (i. e. $g(s) = \sum_{i=1}^{|s|} z \cdot i^p$). The value of $g(s)$ is the maximum value of function g. We find the largest values of z and p which satisfy the following inequality:

$$z \cdot \sum_{i=1}^{|r|} i^p < m, \tag{3}$$

where m is an arbitrary value. Function f' is defined by identifying values of $f'(x)$ for each $x \in \{A, C, G, T\}$. In this regard, value of z is assigned to $f'(T)$ and three different numbers less than $f'(T)$ value are assigned to the remaining domain such that $f'(A) < f'(C) < f'(G)$. Since we limit the maximum value of function g to m, it is possible to find two sequences r and r' such that $r \neq r'$ and $g(r) = g(r')$. Thus, g as a hash function has some collisions. To solve this, we generate two numbers for each sequence r. In other words, we consider two hash functions called $score1$ and $score2$ based on the function g, with two different functions as f'. We define function f' for $score1$ as follows:

$$f'(A) = n1, f'(C) = n2, f'(G) = n3 \ \ and \ \ f'(T) = n4,$$

where

$$n_{i1 \leq i \leq 4} = \begin{cases} \lfloor \frac{i}{4} \times z \rfloor & \lfloor \frac{i}{4} \times z \rfloor \ mod \ 2 \neq 0, \\ \lfloor \frac{i}{4} \times z \rfloor - 1 & o.w. \end{cases} \tag{4}$$

To define function $score2$, we change the definition of f' such that current values of f' are replaced by a permutation of them as follows:

$$f'(A) = n3, f'(C) = n1, f'(G) = n4 \ \ and \ \ f'(T) = n2.$$

Lemma 2.1. *Assume that two numbers between 1 and n are selected uniformly at random. The probability of equality between the two numbers is* $\frac{1}{n^2}$.

According to Lemma 2.1, applying the functions $score1$ and $score2$ can dramatically reduce the collision rate in the hash table.

In order to test the hash functions, we design a prototype read mapper, SBA (Score Based Alignment) which maps a set of reads on a genome sequence. Firstly, the reference genome is indexed by SBA. For this purpose, the reference is scanned by an overlapping window with the length of L (read length). For each L-mer, $S1$ and $S2$ are computed using functions $score1$ and $score2$, respectively. Then, $S2$ value and position of L-mer on the genome are stored in the bucket with the index number of $S1$ value. Secondly, reads are mapped on the reference by searching the hash table. In this way, $S1$ and $S2$ are calculated and looked up at the hash table for each read. Consequently, the locations of reads aligned exactly on the reference are determined.

3. Datasets

To evaluate the hash function, we use*Escherichia coli str. K12 substr. MG1655* [GenBank:NC_000913] genome sequence with the length of 4,639,675 bp. We simulate some read sets for *E. coli* genome of lengths 21bp, 45bp and 101bp with coverage depth 20x as follows. The datasets ReadSet1_21, ReadSet1_45 and ReadSet1_101 are generated with no errors. To simulate datasets ReadSet2_21, ReadSet2_45 and ReadSet2_101, a genome sequence is derived from *E. coli* with the rate of 0.1% single nucleotide variants, then read sets are simulated for the mutated genome. The datasets ReadSet3_21, ReadSet3_45 and ReadSet3_101 are simulated with sequencing errors. All datasets are generated by DWGSIM simulator using the Illumina platform.

4. Result

Values of p and z in Eq. (3) are calculated by setting m and $|r|$ with an index size and a read length, respectively. For the genome size of *E. coli* as value of m and three different read lengths, p and z values are shown in Table 1. For each case of Table 1, we identify function f' as mentioned in section 2 to obtain the hash function $score1$. Then, the genome sequence of *E. coli* is indexed by $score1$. Fig. 1 illustrates the collision distributions of the function $score1$ applied to the index of *E. coli* genome for three different read lengths. The result indicates that the function $score1$ on a genome sequence with any read length has a collision problem. The total numbers of t-way collisions (i.e. t fragments that hash to the same value) for read lengths 21 bp, 45 bp and 101 bp are 1805261, 1255376 and 636958 respectively. Thus the shortest read length (21 bp) has the most total collisions and 2-way collisions. In addition, the number of t-way collisions for it declines sharply. Consequently the shorter read length results the more collision number.

Table 1: The values of p and z in Eq. (3) for three read lengths

$\|r\|$	p	z
21	4	5
45	3	4
101	2	13

Table 2: Collision rates of hashing mechanism with functions $score1$ and $score2$ are computed for read sets of various lengths (rows) with different index size (m) values (columns).

$\|r\|$	$m = \|G\|$	$m = 2\|G\|$
21	2×10^{-6}	1×10^{-6}
45	1×10^{-5}	2×10^{-7}
101	5×10^{-5}	1×10^{-6}

To resolve the collision problem of the hash function $score1$, we employ functions $score1$ and $score2$ for indexing a genome as noted in section 2. As shown in Table 2, the collision rates drop sharply by considering two hash functions $score1$ and $score2$ for indexing *E. coli* genome. Furthermore, by considering the index size of $2|G|$, the number of collisions becomes negligible (Fig. 2). It's worth mentioning that hashing system with functions $score1$ and $score2$ constructs hash tables with the size of $O(|G|)$, where $|G|$ is the genome size (2^{22}) while conventional hash function (Eq. (1)) makes hash tables with sizes 4^{21}, 4^{45} and 4^{101} for read sets with read length 21 bp, 45 bp and 101 bp, respectively.

To assess our hashing system in read mapping, we index *E. coli* genome by hash functions $score1$ and $score2$ with two different index sizes: the genome length ($|G|$) and twice the genome length ($2|G|$). Then for each simulated dataset in section 3, SBA mapper is performed on each index separately. As shown in Table 3, for each dataset, the percentage of mapped reads is the same for the indices with sizes $|G|$ and $2|G|$ while the number of reads mapped incorrectly on the genome varies slightly between the two cases. In fact, for the datasets ReadSet1_21, ReadSet1_45 and ReadSet1_101 which have no errors, all reads are mapped on correct locations whereas for the datasets with errors, more than 80% of reads are mapped to the genome of which very few reads ($< 0.001\%$)are mapped to incorrect locations. In addition, the number of incorrect mapped reads for the indices with the size of $2|G|$ is less than that of the indices with the size of $|G|$ except for the read sets with read length of 21 bp. It seems that because the collision rate of the hash table with the size of $|G|$ with read length of 21 bp is very low, increasing the size of index doesn't have any effects on improvement of incorrect mapping rate.

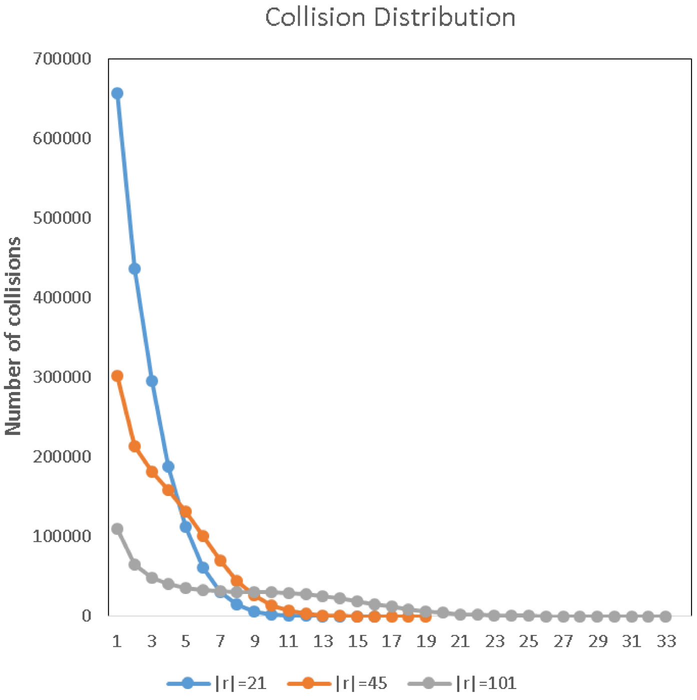

Figure 1: The genome sequence of *E. coli* is indexed by $score1$ with read lengths of 21 bp, 45 bp and 101 bp. Then the number of collisions are computed. Horizontal axis indicates the number of fragments with the same hash value of $score1$.

Our assessment demonstrates that employing hash functions $score1$ and $score2$ for indexing a genome with the index size of $|G|$ is appropriate for mapping reads on the genome because nearly all reads are mapped to the correct locations.

5. Conclusion

Our goal is to index a genome by applying k-mers with the length of a read. To achieve this purpose, we propose a hashing system by two functions which makes a hash table with a low collision rate. To assess our hash functions on the simulated data, we design SBA mapper. The result suggests that for datasets with varying read length, SBA mapper can map approximately all reads on their correct genomic locations. In fact, an application of proposed hashing system and specially SBA mapper is exactly mapping Illumina short reads on a genome, where it is required reads to be mapped without any mismatches or gaps. In this case, SBA mapper can reduce number of candidate locations for mapping and yield decreasing in searching time.

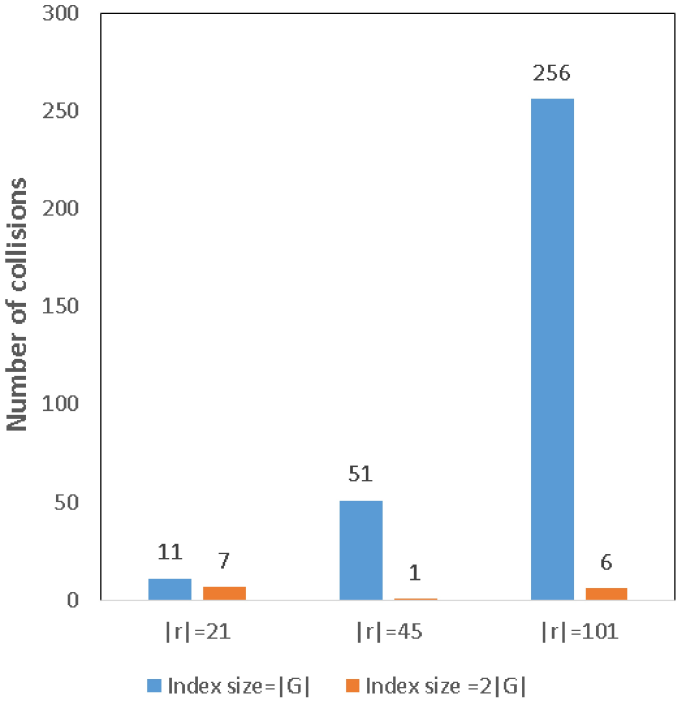

Figure 2: The genome sequence of *E. coli* is indexed by functions $score1$ and $score2$ for two index sizes with read lengths of 21 bp, 45 bp and 101 bp. $|G|$ is the genome length of *E. coli*. Then the number of collisions are computed.

Table 3: Simulated dataset analysis

dataset	mapped reads (%)	mapped reads	incorrect mapped reads	mapped reads	incorrect mapped reads
		index size = $\lvert G\rvert$		index size = $2\lvert G\rvert$	
Readset1_21	100	4418738	0	4418738	0
Readset2_21	97.92	4326950	5	4237759	4
Readset3_21	95.9	4237761	49	4237760	48
Readset1_45	100	2062076	0	2062076	0
Readset2_45	95.61	1971561	4	1971557	0
Readset3_45	91.34	1883565	16	1883558	9
Readset1_101	100	918746	0	918746	0
Readset2_101	90.41	830725	14	830711	0
Readset3_101	81.64	750121	38	750088	5

References

[1] A. Ahmadi, A. Behm, N. Honnalli, C. Li, L. Weng, X. Xie, Hobbes: optimized gram-based methods for efficient read alignment, Nucleic Acids Research, 40(6) (2011) e41-e41.

[2] N. Homer, B. Merriman, S. F. Nelson, BFAST: An alignment tool for lLarge scale genome resequencing. PLoS ONE, 14(11) (2009) 1-12.

[3] B. Langmead, S. L. Salzberg, Fast gapped-read alignment with Bowtie 2. Nature Methods, 9(4) (2012) 357-359.

[4] B. Langmead, C. Trapnell, M. Pop, S. L. Salzberg, Ultrafast and memory-efficient alignment of short DNA sequences to the human genome, Genome Biology, 10(3) (2009) R25.

[5] W-P. Lee, M. P. Stromberg, A. Ward, C. Stewart, E. P. Garrison, G. T. Marth. Mosaik: A hash-based algorithm for accurate next-generation sequencing short-read mapping, PLOS ONE, 9(3) (2014) 1-11.

[6] H. Li and R. Durbin. Fast and accurate short read alignment with burrows-wheeler transform, Bioinformatics, 25(14) (2009) 1754-1760.

[7] S. M. Rumble, P. Lacroute, A. V. Dalca, M. Fiume, A. Sidow, M. Brudno, Shrimp: Accurate mapping of short color-space reads, PLOS Computational Biology, 5(5) (2009) 1-11.

[8] A. D. Smith, W-Y. Chung, E. Hodges, J. Kendall, G. Hannon, J. Hicks, Z. Xuan, M. Q. Zhang, Updates to the RMAP short-read mapping software, Bioinformatics, 25(21):2841–2842, 2009.

[9] T. F. Smith, M. S. Waterman, Identification of common molecular subsequences, Journal of Molecular Biology, 147(1) (1981) 195-197.

[10] H. Zhang, Y. Chan, K. Fan, B. Schmidt, W. Liu, Fast and efficient short read mapping based on a succinct hash index, BMC Bioinformatics, 19(1):92, 2018.

Please cite this article using:

Farzaneh Salari, Fatemeh Zare-Mirakabad, Mehdi Sadeghi, A new hash function and its use in read mapping on genome, AUT J. Math. Comput., 1(2) (2020) 165-170
DOI: 10.22060/ajmc.2019.15991.1020