A Fast Multiple Hash Function to DNA Nucleotide Sequence using K-Mer Index

S. Jawahar [1], Dr. P. Sumathi [2], Divya Bharathi. S[3], Kavitha. N[4], Sharmada. R[5]

[1]Assistant Professor, Department of Computer Science and Application, Sri Krishna Arts and Science College, Coimbatore-641008. Tamil Nadu. India. [1]shivamjawahar@gmail.com

[2] Assistant Professor, PG & Research Department of Computer Science, Government Arts College, Coimbatore-18. Tamilnadu. India. [2]sumathirajes@hotmail.com

[3,4,5] Student, Department of Computer Science and Application, Sri Krishna Arts and Science College, Coimbatore-641008. Tamil Nadu. India.

Abstract:

Hashing method has different functions which can be used for querying, indexing, similarity search, genome assembly, sequence alignment, K-mer counting. The hash operation in bioinformatics increases the speed and efficiency for different applications. The Biological K-mer sequences are decomposed into K-contiguous values and each value is stored in hash table using the hash function. This paper DNA sequence is decomposed using K-mer and multiple hash function is used to limit memory space and time. The hash function produces short hash value to establish index base paring and avoiding search of all the DNA sequence in the hash table. The proposes FMSHK methods is compared with MD5, SHA1 hash functions, which converts large size DNA sequence into unique small value. The experiments were conducted on artificially generated sequence which is more efficient in memory space and generating hash value.

Keywords:

K-mer, DNA, MD5, SHA1, indexing, sequence alignment.

1. Introduction

Hashing is function which can be used uniquely identify a specific object from a group of similar objects. It is a common function which is used across many applications. The values provided by the hash function represent the hash value. This hash value is also termed as digests, hash codes and/or hashes. Hashing refers to map an input key value of varied size to fixed size memory of predetermined tables. The hash table contains values which are used to index a varied-size table. The hashing or scatter memory addressing is the use of a hash function for indexing a hash table according to user requirement.

Figure 1: Hashing function

A hash function performs three functions: 1) The variable length key is converted to fixed length key using ADD or XOR functions, 2) Mapping key values to the hash table and 3) Distributing values uniformly over key space. The hash function and hash tables are used for data storage and retrieving data which is small in size with constant time interval. The hashing method is efficient in data storage space and computing time of data which avoids ordered and unordered list of non-linear access time with variable length key values.

A hash function converts larger keys into smaller key values in hash table for fast access of the different values. The hashing is used to distribute the key or values across the hash table in the form of arrays. The stored value can be quickly retrieved by using the hashed key.

$$hash = hashfunc(key)$$

$$index = hash \% array\_size$$

Equation 1

In the field of bioinformatics there are many applications which include genome and transcriptome, k-mer counting, sequence alignment, RNA-seq expression quantification and error correction. The k-mer decomposition in DNA sequencing includes similarity searching, indexing and querying values from hash tables. These operations can be implemented efficiently through hash based data structures namely hash tables or Bloom Filters respectively. The hashing algorithm plays a vital role in many bioinformatics tools for performance and accuracy in hash values.

2. Related Work

In DNA sequences there are four nucleotides which is represented as {a,c,g,t} where a is adenine, c is cytosine, g is guanine and t is thymine. The ASCII codes for these nucleotide values are **01000001, 01000011, 01000111, 01010100}** respectively**.** There are three factors which affects the performance of hash table: 1) non-frequent access of hash table bucket, 2) hash function selection and 3) accessing the elements stored in the hash table bucket. Generally hash table is one-dimensional array in indexed method value is computed by a function called hash function. There are four basic operations in hash table:

1. Initialization
2. Insertion
3. Retrieval
4. Deletion

Table 1 represents the character code for different DNA sequences. The DNA character A has an index code 00, C has index code 01, G has an index code 10 and T has an index code 11. Figure 2 represents the example for DNA sequence which has input value ‘ACGTACGTACGTACGT’ and corresponding index code is given in the figure.

Table 1: Character code for DNA sequence

Nucleotide	Bases	Index Code
Adenine	A	00
Cytosine	C	01
Guanine	G	10
Thymine	T	11

A	C	G	T	A	C	G	T	A	C	G	T	A	C	G	T
00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11

Figure 2: Encoding Characters of DNA Sequence

The encoding of DNA sequence characters is the process of changing each DNA character using set of symbols in encoding systems and various types of data. This encoding method is also termed as character map or character set. These DNA characters use 2 bits of total memory space for storing individual characters. It reduces the memory usage by 2/3rd of the memory which is lesser than a byte. The applications of hashing include Symbol tables, Database Indexing, Network Processing Algorithm, Browser Cache and Database Indexing.

3. Multiple hash function for biological data

An n-gram or k-mer is a sequence of 'k' consecutive values which is denoted as $\sum$. In our approach k-mer value for biological data (DNA) is stored in hash table with auxiliary data structure. The k-mer correct neighbour value is calculated by examining the k-mer neighbour bit in DNA sequence. The other k-mer neighbour values can be generated by using the same process. The hash algorithm must satisfy statistical property for mapping all k-mer to single integer using one or more hash functions. The single hash function will map a k-mer to single hash value and multiple hash function will map two or more hash values.

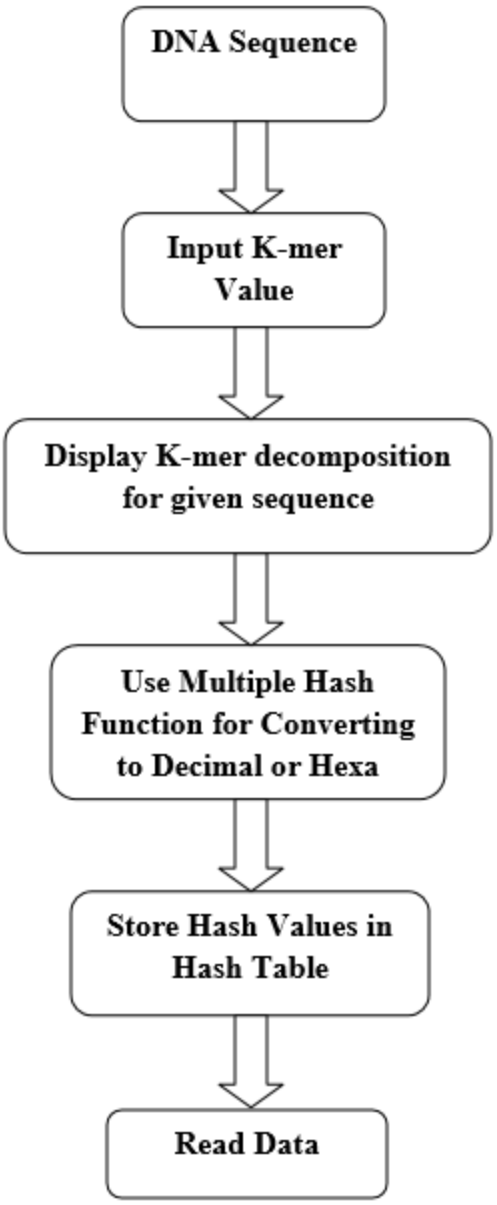


Figure 3. Flow Chart for K-mer decomposition using multiple hash function

In figure 3 represents the overall function of the system which contains DNA sequence as input value, multiple hash function to avoid collision, user desired K-mer value, decomposing the input DNA sequence according to 'K' value and storing the hash value in the hash table with corresponding index value.

Proposed Fast Multiple Short Hashing using K-mer (FMSHK) Algorithm:

Step 1: Select a user desired 'K' value for decomposing the given sequence or pattern into K-mer of size by sliding window method.

Step 2: By using the decomposed sequence pattern in k-mer select the least occurrences pattern and detect its location in the sequence.

Step 3: The multiple hash function is used to convert the input DNA sequence into decimal or hexa value.

Step 4: The key and value for pattern or sequence is checked and verified for each value in the hash table.

Step 5: The process continues for searching remaining K-mer patterns in the given sequence using hash function and Counting patterns.

In our approach all k-mers of a DNA sequence are hashed on the basis of canonical hash value for k-mer in given DNA sequences effectively and efficiently. Given a DNA string S, let $\bar{S}$ be its complement reverse using Watson-Crick method. Suppose h: $\Sigma^* \rightarrow Z$ h: $\Sigma^* \rightarrow Z$ is a DNA string function which is arbitrary in function, it can be defined as

$$\tilde{h}(s) \triangleq \min\{h(s), h(\bar{s})\} \quad \textbf{Equation 1}$$

Then for any DNA string S,

$$\tilde{h}(s) = \tilde{h}(\bar{s}) = \tilde{h}(\text{canonical}(s|h)) \quad \textbf{Equation 2}$$

The canonical hash function can be defined in other way which contains 2-bit encoding hash function. The input has any DNA sequence with any integer hash function which can be defined as,

$$h'(s) \triangleq g\big[\min\{h_0(s), h_0(\bar{s})\}\big] \quad \textbf{Equation 3}$$

In this paper DNA sequence is represented in numerical form which is one to one relationship between the nucleotides. A particular numerical value is assigned to nucleotides in the sequence as, A is assigned as 0, C is assigned as 1, G is assigned as 2 and T is assigned as 3 respectively in decimal form.

4. Experimental Results

The complete hashing process is illustrated in the figure 4 with an example. The input for the process read may be FASTA or FASTQ format. The k-mer value is user defined according to the input value for the process.

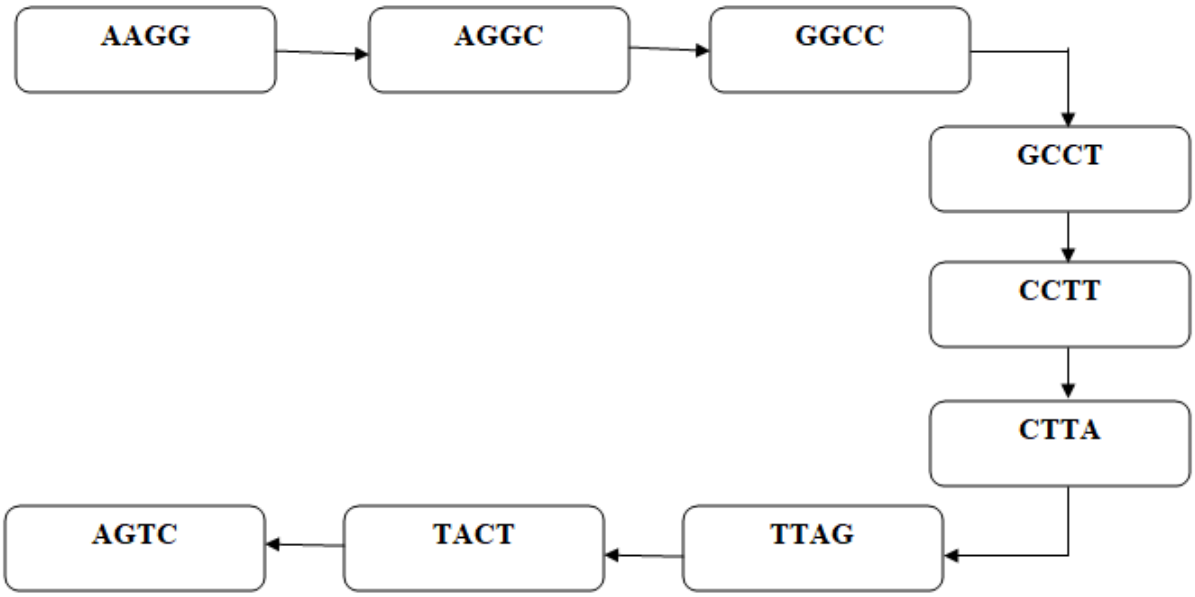


Figure 4: K-mer nodes decomposition

By using this method the k-mer is generated for all possible k-mer values. The input read for this read sequence is **AAGGCCTTAGTC.** The value for k= 4 and h=2, where 'h' represents the total number of hash functions used in the read operation. The all possible k-mer for the above input is {AAGG, AGGC, GGCC, GCCT, CCTT, CTTA, TTAG, TACT, and AGTC}. There is totally 9 k-mer decomposition values for the input value.

Table 2: K-mer short hash value for broken reads

Index	4-Mer	Hash Value	3-Mer	Hash Value	2-Mer	Hash Value
0	AAGG	8NML	AAG	hld	AA	y6
1	AGGC	8Ple	AGG	hog	AG	yc
2	GGCC	9Cls	GGC	iVI	GG	Bf
3	GCCT	9BjI	GCC	iTG	GC	Bb
4	CCTT	95qQ	CCT	hRW	CC	z9
5	CTTA	99Om	CTT	i1A	CT	zq
6	TTAG	bnLd	TTA	bnKc	TT	I4
7	TAGT	biU9	TAG	mfx	TA	HK
8	AGTC	8PrP	AGT	hot	AG	yc

In the table 2 the k-mer decomposition for the input sequence is displayed with corresponding hash value. The k-mer value used in the table is k = 4 and ht = 2, where 'ht' represents the total number of hash function used in the process. The first k-mer is considered, AAGG, the hash value for the AAGG is 57. Since the hash table contains 9 values the index of the table is calculated as, (57%9). The index value is 3 and the quotient is 6 which is stored in the hash table. The same method is repeated for remaining k-mers for storing the values in the hash table. The hash tables uses dynamic memory allocation to perform the program to read FASTA files with different size. The time complexity of the hash tables of number of nucleotides or amino acids is approximately a linear function which identifies the existing sequence in the hash table.

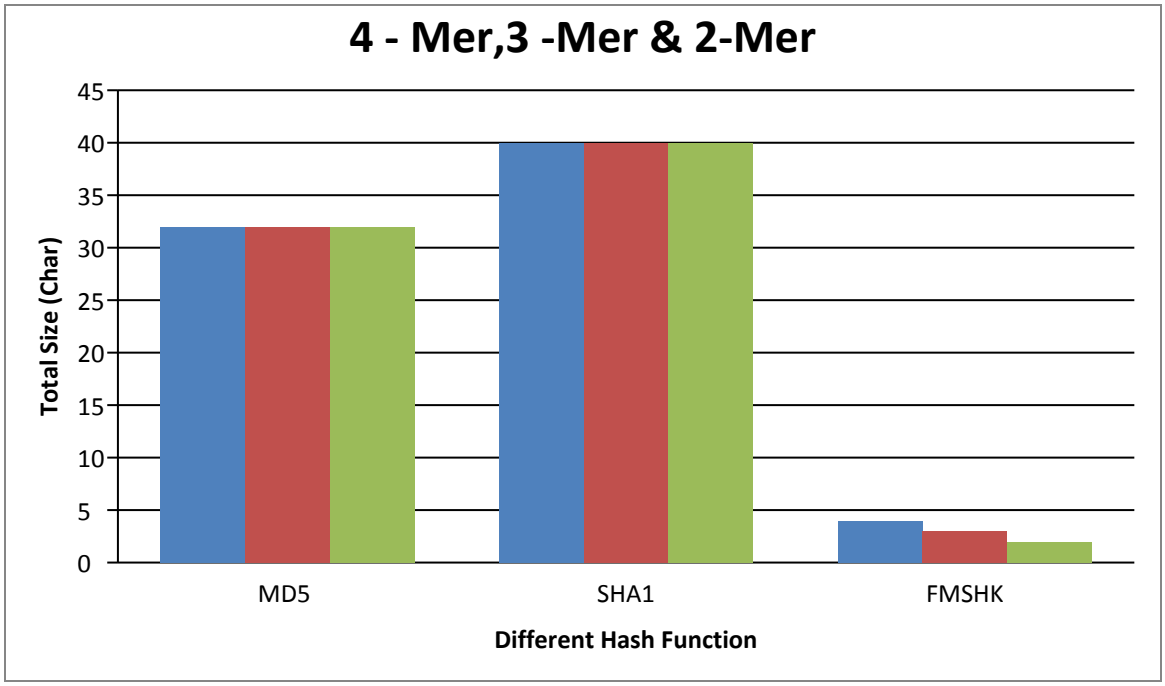


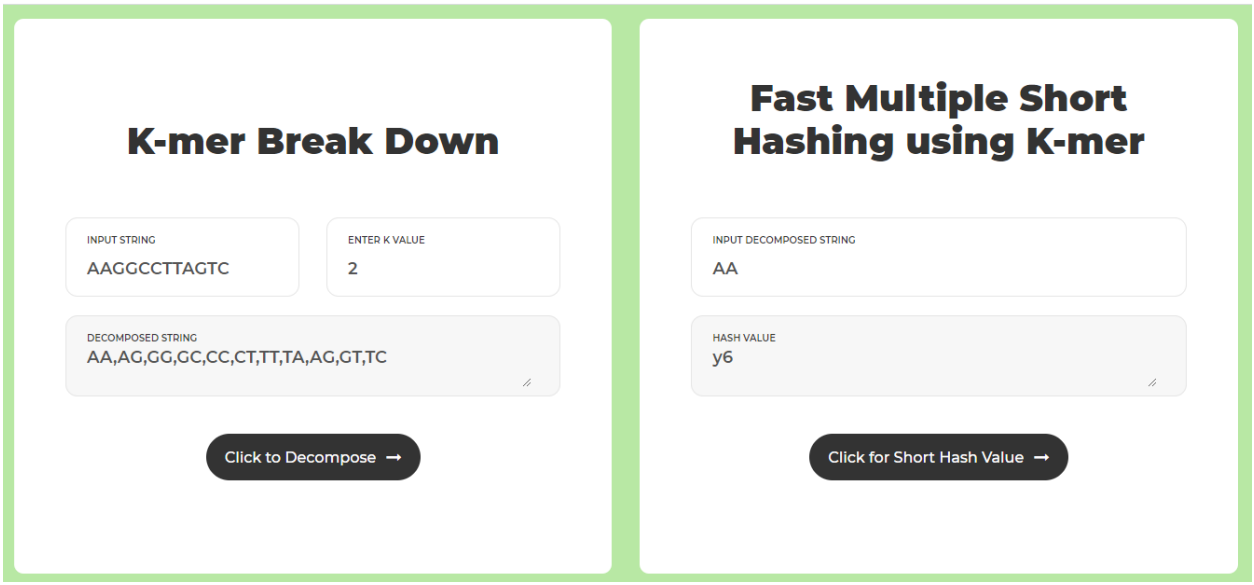
Figure 5: Total Hash value size (char) for MD5, SHA1 and FMSHK methods.

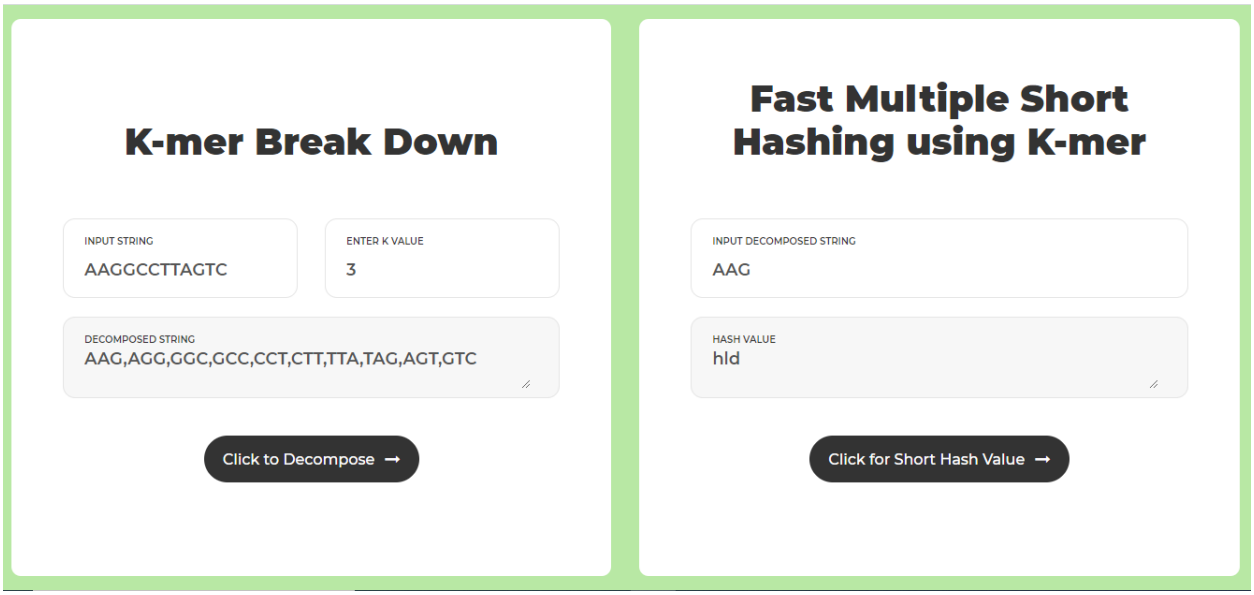
In the figure 5 the total hash value size is calculated in terms of characters in the output value. The proposed FMSHK method is more efficient when compared to existing MD5 and SHA1 function. The K-mer value used in the function is K = (4, 3, 2) for calculating hash value size. For K = 4 the total hash value size for MD5 is 32 length in character, SHA1 size is 40 length in character and proposed FMSHK method has 4 length in character. For K = 3 the total hash value size for MD5 is 32 length in character, SHA1 size is 40 length in character and proposed FMSHK method has 3 length in character. For K = 2 the total hash value size for MD5 is 32 length in character, SHA1 size is 40 length in character and proposed FMSHK method has 2 length in character. In our proposed approach hash value size in terms of character is filtered and minimized.

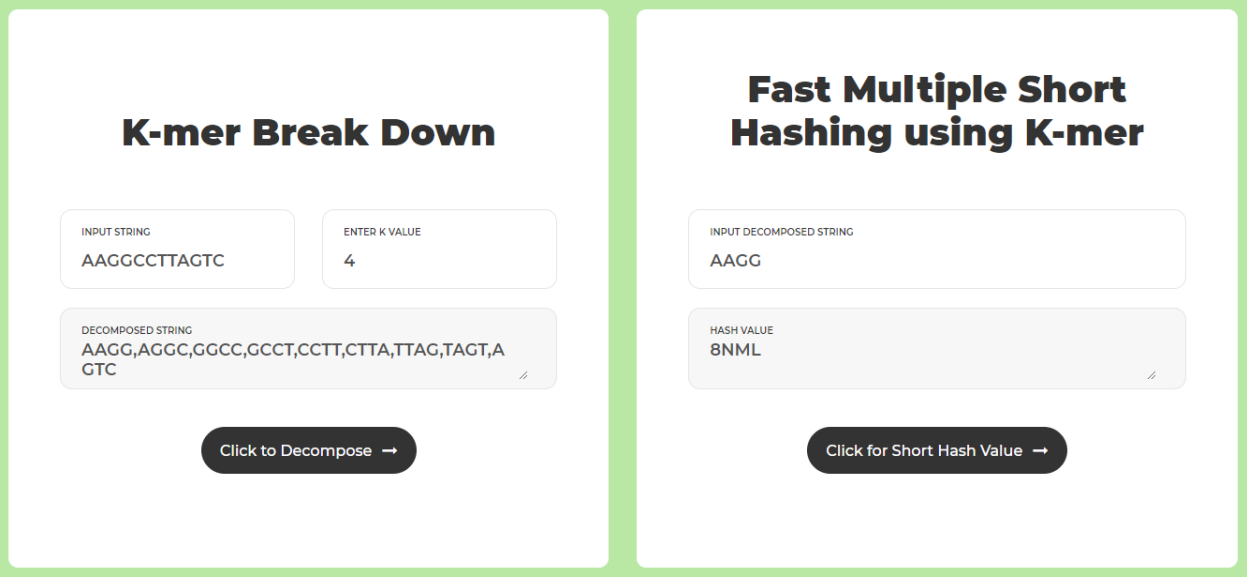
5. Conclusion

The proposed FMSHK method uses multiple hash function for DNA sequences for solving K-mer nucleotide with memory consumption, multiple hash index and hash table. The proposed FMSHK is compared with MD5 and SHA1 functions with different K-mer values. The FMSHK method is more efficient in calculating the hash value when compared with MD5 and SHA1

functions. The retrieval time of values from the hash table is also minimized by decreasing the hash value size.







REFERENCES

[1] Introduction to Algorithms,2nd edition by Building a Better Bloom Filter Thomas H. Coremen, Charles E. Leiserson Adam Kirsch_ and Michael Mitzenmacher_ ,Ronald L. Stein,.PHI,Chapter 11. Division ofEngineering and Applied Sciences Harvard University, Cambridge, MA 02138.

[2] Algorithms in C, 2nd edition,3rd edition by Robert Sedgewick,. Addison- Wesley, 1998.Chapter 14.

[3] J. Zobel, S. Heinz, and H.E. Williams. Inmemory Hash Tables for Accumulating Text Vocabularies.

[4]. J. Reneker and C.-R. Shyu, “Refined repetitive sequence searches utilizing a fast hash function and cross species informationretrievals,”BMCBioinformatics,vol.6,article111,2005.

[5]. Yang Xu; Lei Ma; Zhaobo Liu; Chao, H.J., "A Multi-dimensional Progressive Perfect Hashing for High-Speed String Matching," Architectures for Networking and Communications Systems (ANCS), 2011 Seventh ACM/IEEE Symposium on , vol., no., pp.167,177, 3-4 Oct. 2011

[6] Yasuda, K.; Miura, T.; Shioya, I., "Distributed Processes on Tree Hash," Computer Software and Applications Conference, 2006. COMPSAC '06. 30th Annual International, vol.2, no., pp.10,13, 1721 Sept. 2006.

[7] Le Cai and Yung-Hsiang Lu, Energy Management Using Buffer Memory for Streaming Data, IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 24, NO. 2, pp. 141-152, FEBRUARY 2005.