“Reimplementation of Burrows-Wheeler Transformation for genome wide sequence data compression “

By

Palash Sethi

Under the Guidance of

Prof. Shandar Ahmad

School of **C**omputational and **I**ntegrative **S**ciences

Jawaharlal Nehru University

New Delhi – 110067

Introduction:

Compression techniques are generally used to handle a large amount of data. The compression methods reduce the storage space and increase the data circulation(e.g. among peers or research institutes). The compression technique owes its origin to Claude E Shannon's 1948 paper '*A Mathematical Theory of Communication'* in which he formulated the theory of data compression. Nowadays data compression is used almost everywhere as the amount of data stored and transmitted is huge.

With the advancement in sequencing technology the cost of sequencing has seen a huge fall in terms of cost. Companies like Illumina offer whole genome sequencing for as less as five thousand US dollars. The sequence dataset of large scale projects like 1000 genome projects, which aim at sequencing the genomes of several thousand humans and determining the genetic variants with at least 1% frequency, contains around 6 trillion base pairs. Along with humans, other species' genomes are also being sequenced. The large amount of data produced by sequencing methods makes it difficult to store and transfer the data. In 2013, Beijing Genomics Institute used 188 sequencers to produce ~3 PB of raw sequencing read files. The additional output space for mapping to the reference genomes take additional 7 PB of space. While the sequencing techniques have seen a rapid growth, the space required for storing has not been able to keep the pace. As seen in figure 1, the cost of sequencing a single base has been halving every 8 months in 2008-2013 while the cost of hard disk has been halving every 25 months. The cost of storing data in cloud storage, though more reliable, is more than the cost of storing in HDD. The trends show that the cost of storage and transfer is becoming comparable to the cost of sequencing and IT costs will be a significant obstacle in near future.

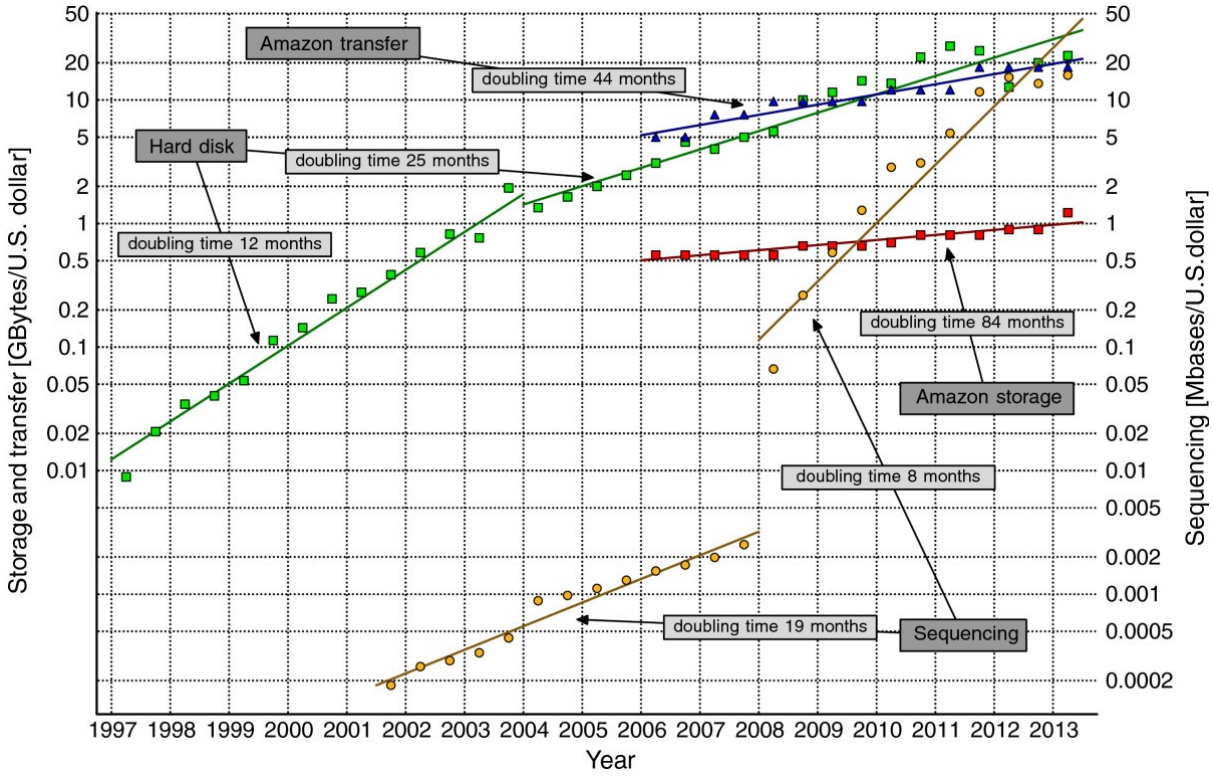


Fig 1:*Image from *Data Compression for Sequencing data, Sebastian Deorowicz and Szymon Grabowski,Algorithms for Molecular Biology,2013

A code is an assignment of bit strings to symbols present in the data such that the strings can be decoded without any ambiguity to recover the original data. Compression involves encoding information using codes that have fewer bits than the original representation. It is the art of reducing the number of bits needed to store or transmit data.

An example of compression lies in Morse Code. Each letter of the alphabet of English language is assigned a code made up of dots and dashes. The most frequent letters like E and T receive the shortest code while the least common like Q, Z are assigned the longest codes. This ensures that the information is transferred using least number of dots and dashes.

Compression:

1. Compression involves encoding(changing data representation) information using codes that have fewer bits than the original representation.

2. It is the art of reducing the number of bits needed to store or transmit data.

Compression Algorithms:
Compression algorithms are techniques that exploits data redundancy (repetion in data) to reduce the size of the data representation. To encode data, we need the encoding representation of each character in data.

Types of Data Compression:

- *Lossless:*
  Lossless data compression algorithms usually exploit statistical redundancy to represent data without losing any information, so that the process is reversible. Lossless compression is possible because most real-world data exhibits statistical redundancy.
  Example: Example of Red, Red,Red, Blue can be represented as 4R1B (no loss in data)

- *Lossy:*
  In these schemes, some loss of information is acceptable. Dropping nonessential detail from the data source can save storage space.

Entropy Encoding:

- Entropy coding is a type of lossless coding to compress digital data by representing frequently occurring patterns with few bits and rarely occurring patterns with many bits. In simple terms, Entropy is a measure of information per byte.

- Two of the most common entropy encoding techniques are:
  1)Huffman coding
  2)Arithmetic coding.

Burrows-Wheeler Transform:

The Burrows–Wheeler transform (BWT) rearranges a character string into runs of similar characters. This is useful for compression, since it tends to be easy to compress a string that has runs of repeated characters by techniques such as move-to-front encoding

It is an algorithm that permutes the given input in such a way that Compression Algorithms can compress the string better. It is not specific for next gen sequencing but can be used with any type of data.

BWT rearranges the characters in the input so that there are lots of blocks with repeated characters. Its goal is to ‘apply a reversible transformation to a block of text to form a new block that contains the same characters but is easier to compress by simple compression algorithms. This transformation group characters together so that the probability of finding a character close to another instance of the same character is increased substantially .

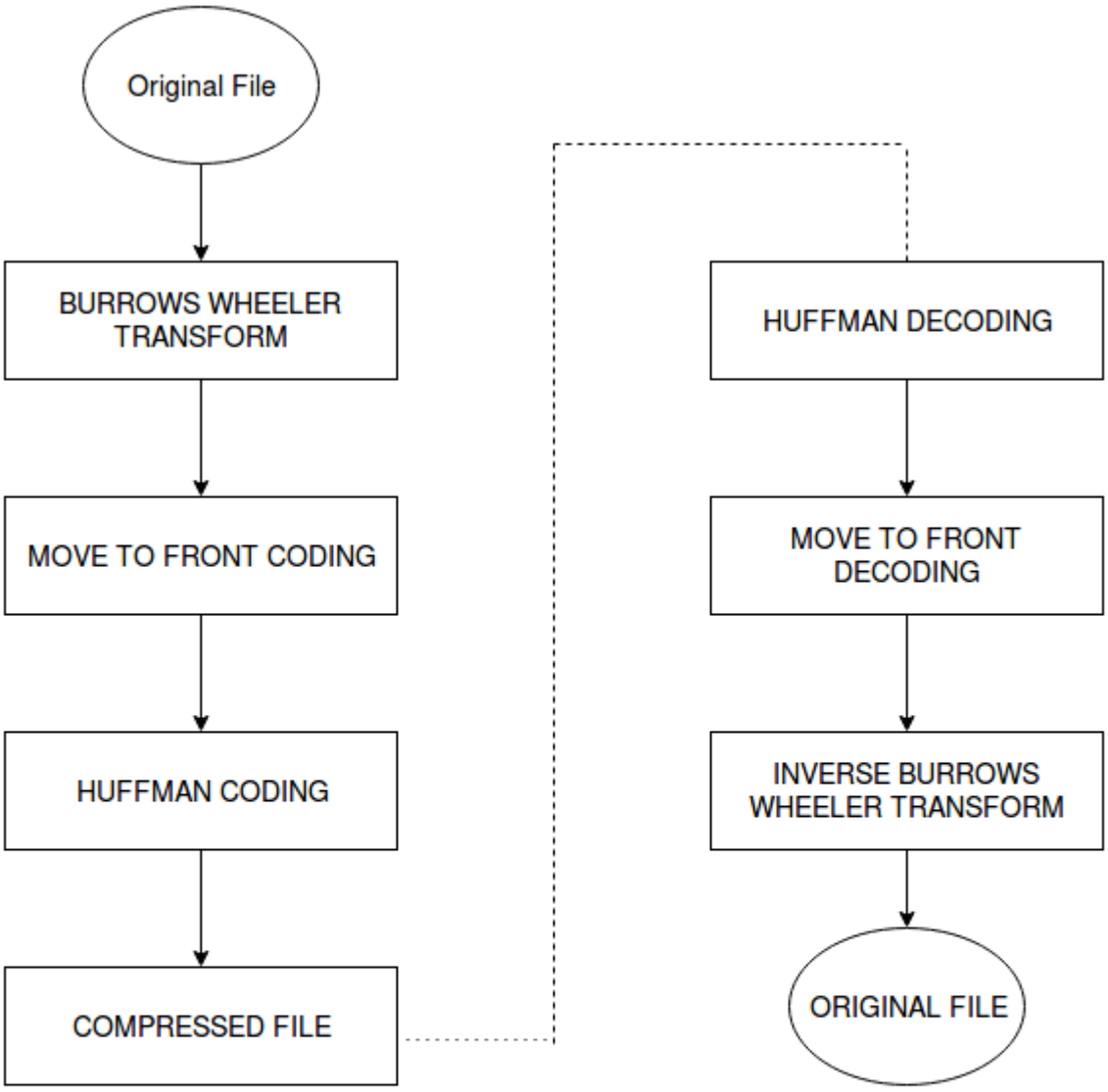
Move-to-Front Encoding:

The move-to-front (MTF) transform is an encoding of data designed to improve the performance of entropy encoding techniques of compression. When efficiently implemented, it is fast enough that its benefits usually justify including it as an extra step in data compression algorithms.

The MTF transform takes advantage of local correlation of frequencies to reduce the entropy of a message. Indeed, recently used letters stay towards the front of the list; if use of letters exhibits local correlations, this will result in a large number of small numbers such as "0"'s and "1"'s in the output.

Huffman Encoding:

Huffman coding is a lossless data compression algorithm. The idea is to assign variable-legth codes to input characters, lengths of the assigned codes are based on the frequencies of corresponding characters. The most frequent character gets the smallest code and the least frequent character gets the largest code.



AN OVERVIEW OF BWT BASED COMPRESSION

Example of Compression:

Let the input file contain the string “compression”.

Below we dicuss how the compression algorithms transforms the given input to produce final output.

Burrows-Wheeler Transorm:

Input: compression **Output:** npsoocimpse
Steps Involved:
1) Cyclic permutations
2) Sort
3) Last column of BWT matrix is output.

0	compressio**n**
1	essioncom**p**
2	ioncompres**s**
3	mpressionc**o**
4	ncompressi**o**
5	ompression**c**
6	oncompress**i**
7	pressionco**m**
8	ressioncom**p**
9	sioncompre**s**
10	ssioncompr**e**

Move-to-Front Transform:

Input: npsoocimpse (o/p of BWT)
Output: [4,7,8,7,0,4,6,7,8,5,8]

index	L[index]	Y (Lexically sorted set of all characters of input string) [0 1 2 3 4 5 6 7 8]	R
0	n	[c e i m **n** o p r s]	[4]
1	r	[n c e i m o p **r** s]	[4,7]
2	s	[r n c e i m o p **s**]	[4,7,8]
3	o	[s r n c e i m **o** p]	[4,7,8,7]
4	o	[**o** s r n c e i m p]	[4,7,8,7,0]
5	c	[o s r n **c** e i m p]	[4,7,8,7,0,4]
6	i	[c o s r n e **i** m p]	[4,7,8,7,0,4,6]
7	m	[i c o s r n e **m** p]	[4,7,8,7,0,4,6,7]
8	p	[m i c o s r n e **p**]	[4,7,8,7,0,4,6,7,8]
9	s	[p m i c o **s** r n e]	[4,7,8,7,0,4,6,7,8,5]
10	e	[s p m i c o r n **e**]	[4,7,8,7,0,4,6,7,8,5,8]

Huffman Encoding:

Input: [4,7,8,7,0,4,6,7,8,5,8] (O/P of MTF transform)
Output:

R	Huffman Code
0	0110
4	00
5	0111
6	010
7	10
8	11

Output File:

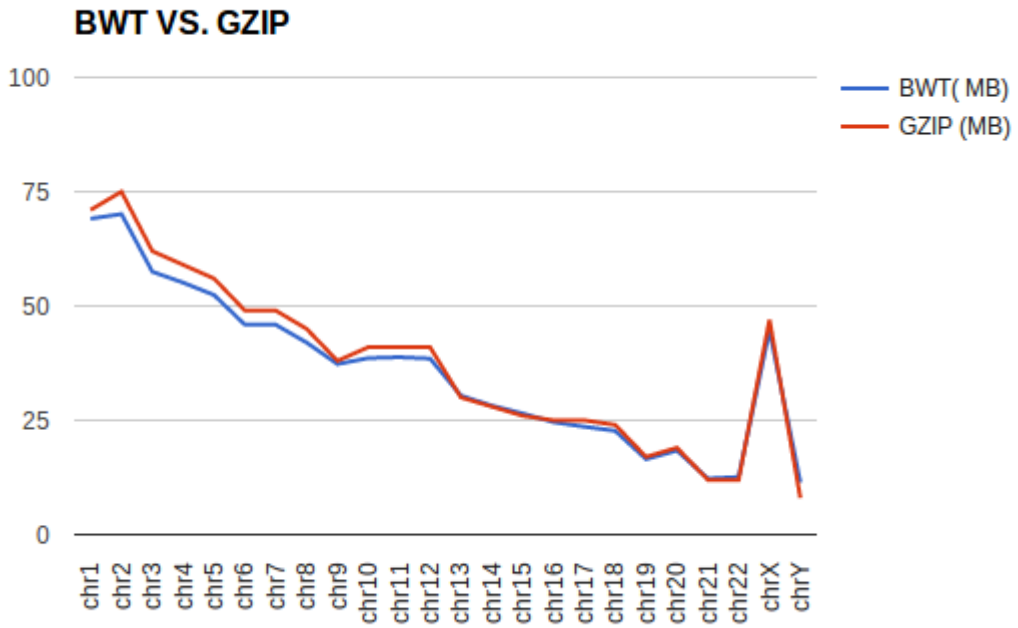
```
27
['e', 's', 'p', 'm', 'i', 'c', 'o', 'r', 'n']
0
{0: '0110', 4: '00', 5: '0111', 6: '010', 7: '10', 8: '11'}
.a[à
```

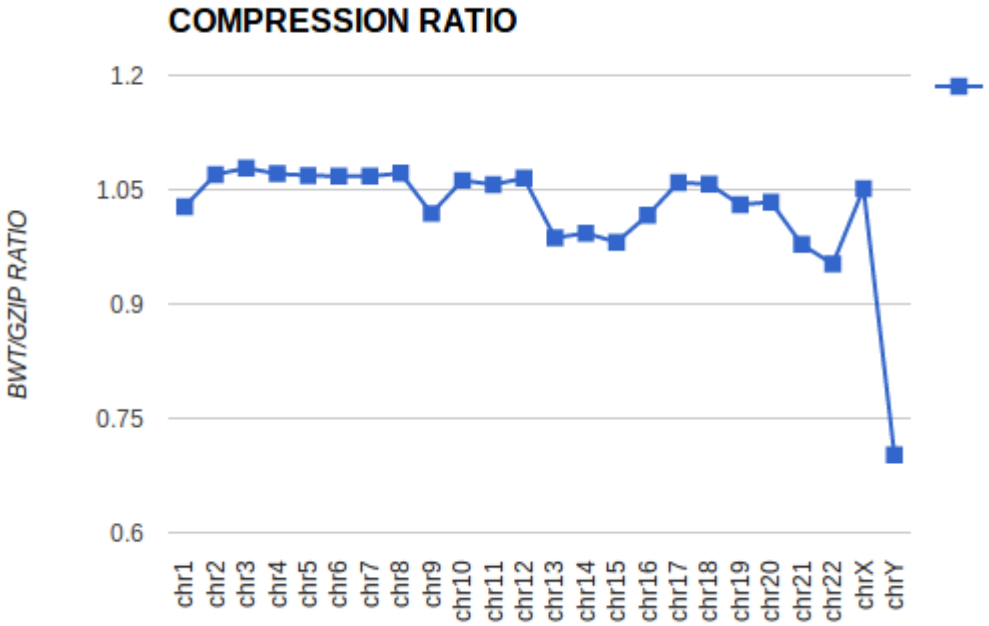
Results

The compression ratio achieved by Burrows-Wheeler Transform have been compared against the compression achieved by gzip. Gzip is the de-facto compression method used by Unix.

Input given to software is human genomic data.
The file sizes are in MB. The overhead incurred for the BWT is due to the dictionary of Huffman codes. Since the dictionary consists of just 4 nucleotide characters and their codes, it is negligible and can be ignored.

chromosome	Original file size	BWT(MB)	GZIP (MB)
chr1	253.9	69.1	71
chr2	247.0	70.1	75
chr3	202.3	57.5	62
chr4	194.0	55.1	59
chr5	185.2	52.4	56.0
chr6	174.2	45.9	49
chr7	162.5	45.9	49
chr8	148.0	42.0	45
chr9	141.2	37.3	38
chr10	136.5	38.6	41
chr11	137.8	38.8	41
chr12	135.9	38.5	41
chr13	116.7	30.4	30
chr14	109.2	28.2	28
chr15	104.0	26.5	26
chr16	92.1	24.6	25
chr17	84.9	23.6	25
chr18	82.0	22.7	24
chr19	59.8	16.5	17
chr20	65.7	18.38	19
chr21	47.6	12.27	12
chr22	51.8	12.6	12
chrX	159.2	44.7	47
chrY	58.4	11.4	8.0


BWT VS. GZIP
100
75
50
25
0
BWT(MB)
GZIP (MB)
chr1
chr2
chr3
chr4
chr5
chr6
chr7
chr8
chr9
chr10
chr11
chr12
chr13
chr14
chr15
chr16
chr17
chr18
chr19
chr20
chr21
chr22
chrX
chrY


COMPRESSION RATIO
BWT/GZIP RATIO
1.2
1.05
0.9
0.75
0.6
chr1
chr2
chr3
chr4
chr5
chr6
chr7
chr8
chr9
chr10
chr11
chr12
chr13
chr14
chr15
chr16
chr17
chr18
chr19
chr20
chr21
chr22
chrX
chrY