

Data Structures and Compression Algorithms for High-Throughput Sequencing Technologies

Kenneth M. Daily, Paul Rigor, Scott Christley, Xiaohui Xie, Pierre Baldi

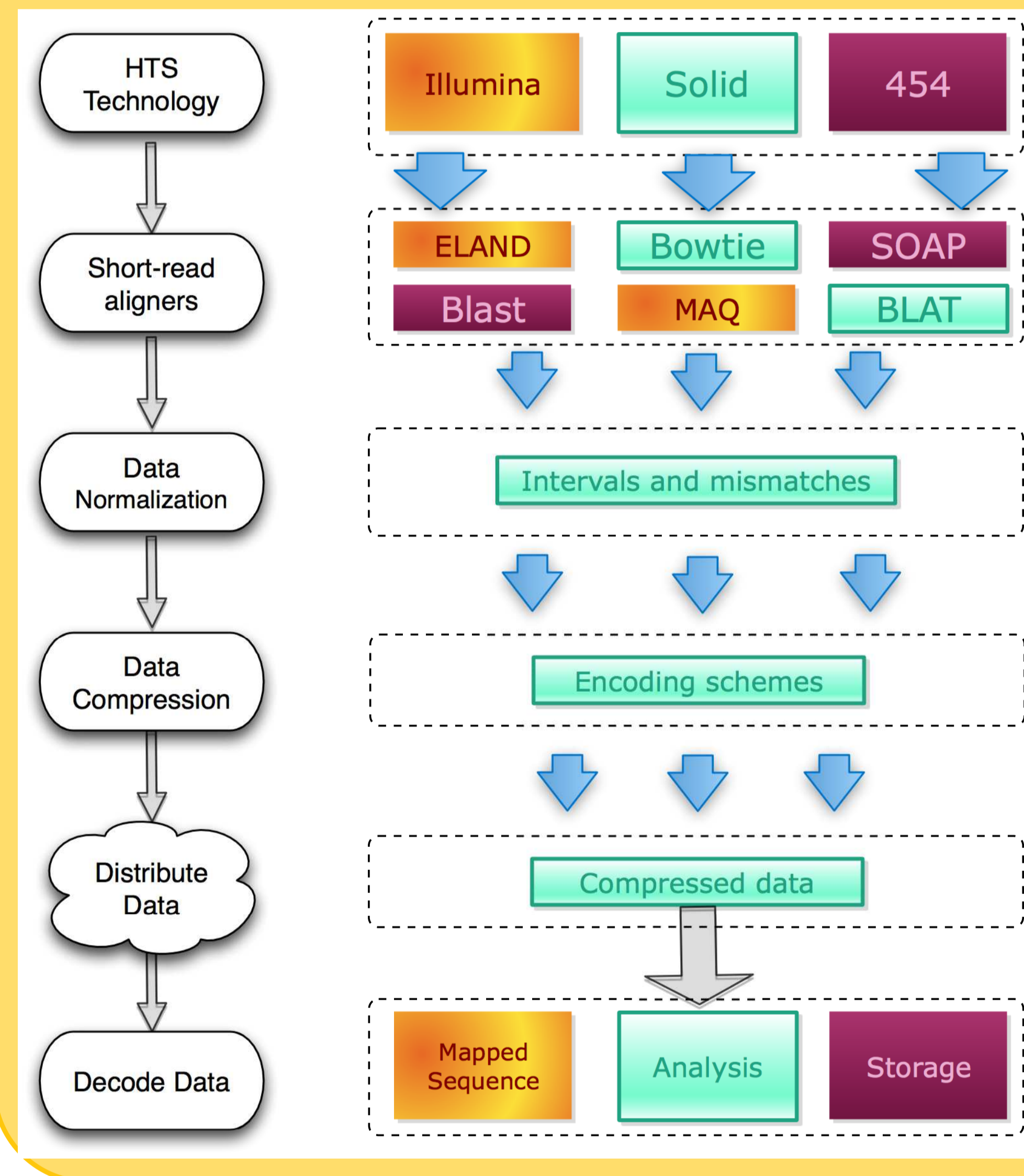
Dept. of Computer Science, Institute for Genomics and Bioinformatics, Dept. of Mathematics, Dept. of Biological Chemistry

Introduction

Sequencing technologies have been one of the major driving forces in the life sciences producing the full genome sequences of thousands of prokaryotic and dozens of eukaryotic organisms, from yeast to man. This trend is being accentuated by modern high-throughput sequencing (HTS) technologies: several human genomes were recently produced [1, 2, 3] and a project to sequence 1,000 human genomes in the next few years is under way [4]. HTS technologies are useful for sequencing and resequencing genomes, digital expression microarrays, ChIP-Seq and SNP genotyping. **In all cases, the amount of data produced by HTS technologies as experiments scale up creates significant bioinformatics challenges to understand, store and share data.**

We are developing data structures and compression algorithms for efficient management and storage of HTS data. Because of the variations that may exist between different technologies and constraints associated with deployment scenarios, our goal is not to provide a single solution, but to describe general methods by which customized solutions can be developed.

General Approach



References

References

- [1] Wheeler DA, Srinivasan M, Egholm M: **The complete genome of an individual by massively parallel DNA sequencing.** *Nature* 2008, **452**(7189):872–6.
- [2] Bentley DR, Balasubramanian S, Swerdlow HP: **Accurate whole human genome sequencing using reversible terminator chemistry.** *Nature* 2008, **456**(7218):53–59.
- [3] Wang J, Wang W, Li R, Li Y: **The diploid genome sequence of an Asian individual.** *Nature* 2008, **456**(7218):60–5.
- [4] Service RF: **The Race for the \$1000 Genome.** *Science* 2006, **311**:1544–1546.
- [5] Johnson DS, Mortazavi A, Myers RM, Wold B: **Genome-wide mapping of in vivo protein-DNA interactions.** *Science* 2007, **316**:1497–1502.

Supported by NIH Biomedical Informatics Training grant (LM-07443-01), NSF MRI grant (EIA-0321390), and NSF grant 0513376 to PB. SC is supported in part by NIH P50 GM76516.

Elias Gamma Encoding

An integer can be represented as a preamble $p(j)$ and a mantissa $m(j)$. $p(j)$ is a string of zeroes of length $\lfloor \log j \rfloor$, and $m(j)$ is the binary encoding of j . **The length of the encoding of j is $2\lfloor \log j \rfloor + 1$.** To decode, first read n 0-bits until the first 1-bit is encountered, then read n more bits to get the binary representation of j . Applying the relationship $-\log P(j) \approx 2\lfloor \log j \rfloor + 1$ to the integer probabilities, shows that **Elias Gamma encoding asymptotically approaches the Shannon limit** for $P(j) \approx Cj^{-2}$ (a power law relationship with exponent -2, C is a normalizing constant).

Datasets

Dataset	# Reads	Length	Coverage
Ty3	0.8M	19	very sparse
Wold [5]	1.7M	25	sparse
Yan22 [3]	31M	23-44	full

Datasets are from sequencing, TF binding, and insertion site mapping experiments with different combination of coverage, redundancy and specificity in the genome.

MOV Encoding

When the value of the integers being encoded increases monotonically, additional loss-less compression can be obtained by encoding only the scale increases and their location. More precisely, if a sequence of strictly increasing addresses is given by (j_1, j_2, \dots, j_K) :

- encode j_1 using Elias Gamma encoding;
- for $i \geq 2$, write $\lfloor \log(j_i) \rfloor - \lfloor \log j_{i-1} \rfloor$ 0-bits followed by the binary value of j_i beginning with its most significant 1-bit.

To decode each integer, set $k = 1$, then repeat:

- increment k by the number of 0-bits in the input stream before reaching the first 1-bit;
- counting this first 1-bit as the first digit of the integer, read the remaining $k - 1$ bits of the integer from the input stream.

Applying Encoding to Mismatches and Locations

A HTS read can be described by its mapped location to a reference sequence and the mismatches that occur between the original read and the reference. The **location** consists of a **chromosome**, a **integer position**, the **strand**, and the **length** of the read (if not fixed). A **mismatch** is given by the **position** in the read and the **nucleotide changed** versus the reference sequence. The position is encoded directly (either relative to the start or end of the read).

Location	EG Encoded Absolute (total)	EG Encoded Relative (total)	MOV (total)
1000	0000000001111101000 (20)	-	-
1024	000000000010000000000 (22)	000011000 (9)	010000000000 (12)
5000	000000000001001110001000 (26)	000000000011110100000 (23)	001001110001000 (15)

Mismatch	Ocurrence	Start (total)	End (total)	Combined (total)
25A	3	0000110011 (30)	11 (6)	0000111100 (33)
12G	100	0001100010 (300)	0001101010 (300)	1 (100)

Elias Gamma (EG) encoding was applied to both the absolute and relative start positions, while monotone value (MOV) encoding was applied only to the absolute start positions.

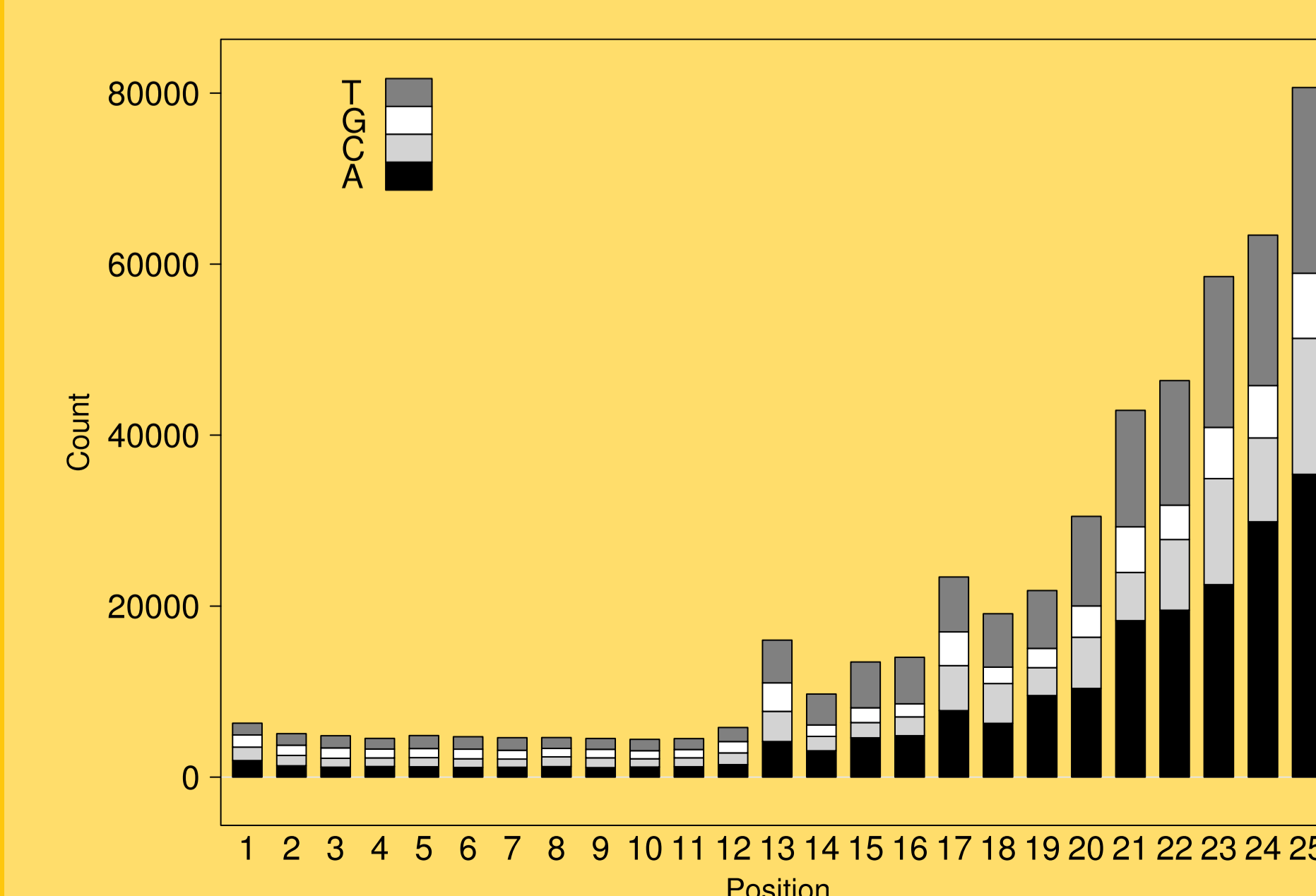
Location Results (in Mb)

	Method	Raw	Gzip	Ours
Ty3	UEG Rel	390	106 (4)	1 (390)
Wold	EG Rel	179	80 (2)	18 (10)
Yan22	EG Rel	3,693	703 (5)	76 (48)

Mismatch Results (in Mb)

	Method	Raw	Gzip	Ours
Ty3	Combined	65	18 (4)	15 (4)
Wold	End NT	20	5 (4)	4 (6)
Yan22	Combined	706	189 (4)	186 (4)

Mismatch distribution



Results and Conclusions

Depending on the properties of the dataset, we see compression rates similar to or exceeding gzip. The spread of the dataset has a large effect on the compression of the locations. Datasets with high coverage do not benefit from only storing unique positions. The mismatch encoding benefits from a non-uniform distribution of position and nucleotide substitution. Although we provide similar, and in some cases, improved compression performance over traditional tools, our goal is not to replace current tools but extend the compression tool chain. We recognize the need to establish best practice solutions for the storage and distribution of genomic information that will mitigate the costs associated with scaling both the storage and network infrastructure required to support the vast amount of data which extends from gigabytes to petabytes range. We provide a methodology to automate the selection of compression schemes that can adapt to datasets with characteristics intrinsic to the type of biological experiments that produce them. Direct implementations of the decoding algorithms process the compressed representations bit-by-bit; however, it is possible to implement faster decoders, which decode the compressed data byte-by-byte.

