SCALABLE GENOMIC COMPRESSION: TRANSFORMING DNA INTO BITS

Ryan Son, Gavin Saxer, Jared Arroyo Ruiz | Advised by Professor Layla Oesper
Department of Computer Science, Carleton College

Introduction

Genome **sequencing** has dropped from $30 million to just **$80 per genome** over the past two decades, removing financial barriers to large-scale research [1]. But with this affordability comes a **flood of data**—making **efficient compression algorithms essential** for cost-effective storage and analysis. We aimed to explore, implement, and test several genomic data algorithms: **DNAZip**, Biocompress, and Huffman Coding.

Compression Strategies

Type	Algorithm	Key Feature
Reference	*DNAZip* (2009)	Uses a **reference** genome to compare against a **target** genome and **compress** the **~0.1%** of **differences** [2]
Non-reference	*Biocompress* (1993)	Exploits intrinsic sequence properties like **repetitions** without a reference [3]
Entropy-based	*Huffman Coding* (1952)	Ordered assigning of the **most frequent** characters/symbols/words/*k*-mer with the **shortest bitstring** encodings [4]

Compressing DNA with DNAZip

DNA is composed of four **unique** nucleotides:

- Adenine (A)
- Guanine (G)
- Cytosine (C)
- Thymine (T)

Three types of possible **variations** may exist at any given position:

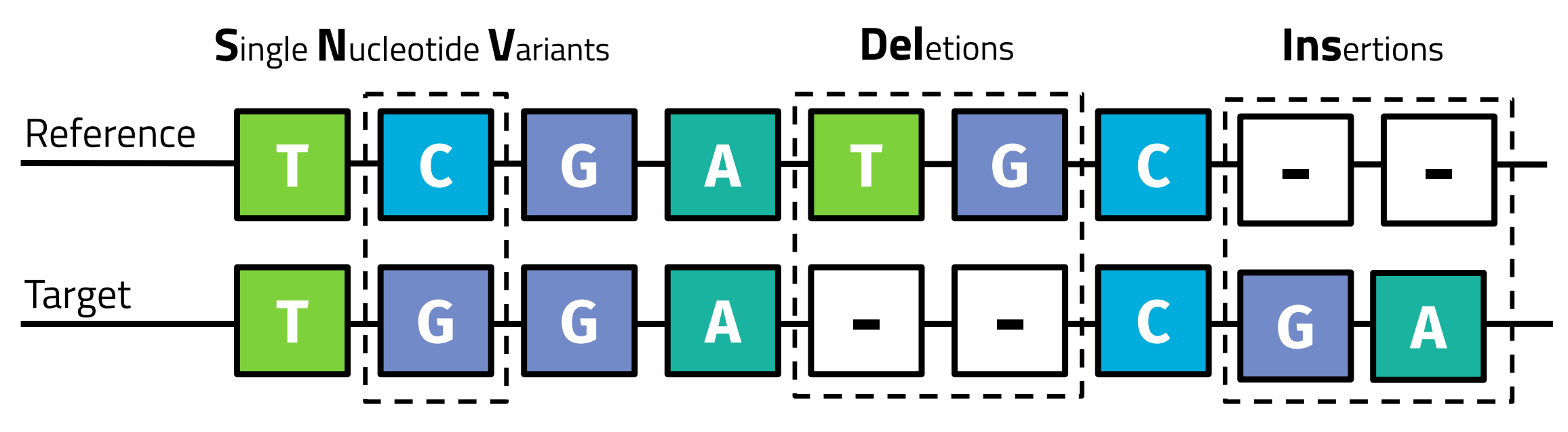


Figure 1. Target genome variations with respect to the reference genome: SNVs - single nucleotide variants; deletions - loss of 1+ nucleotides; and insertions - addition of 1+ nucleotides.

The **DNAZip** algorithm harnesses the fact that human genomes are **99.9% identical** [2]. This requires a **reference** genome, a **target** genome, and a **SNP database**. The target is **compared** against the reference to **create** a variant call format (VCF) file. This **VCF** file contains the **differences/variations** of the **target** genome.

Variation Header	*Variation Info*
SNV (0), CHROMOSOME, POSITION	C/G
DEL (1), CHROMOSOME, POSITION	TG/--
INS (2), CHROMOSOME, POSITION	--/GA

Table 1. Formatting of a VCF file for any target genome.

Nucleotide	*Encoding*
A	00
C	01
G	10
T	11

Table 2. Two-bit encoding for DNA.

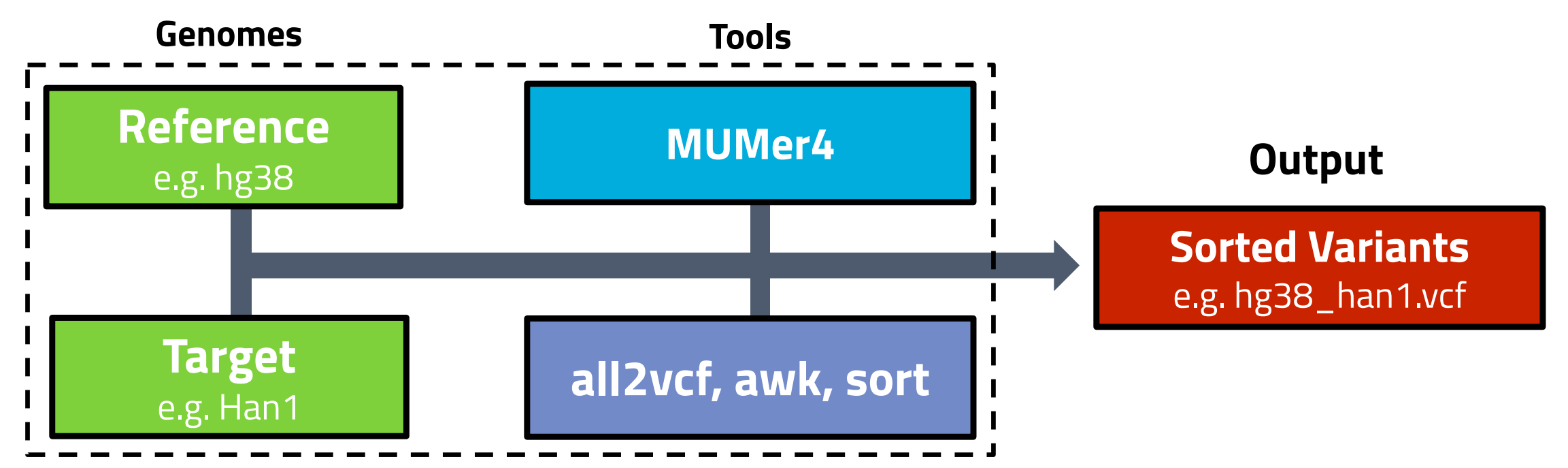
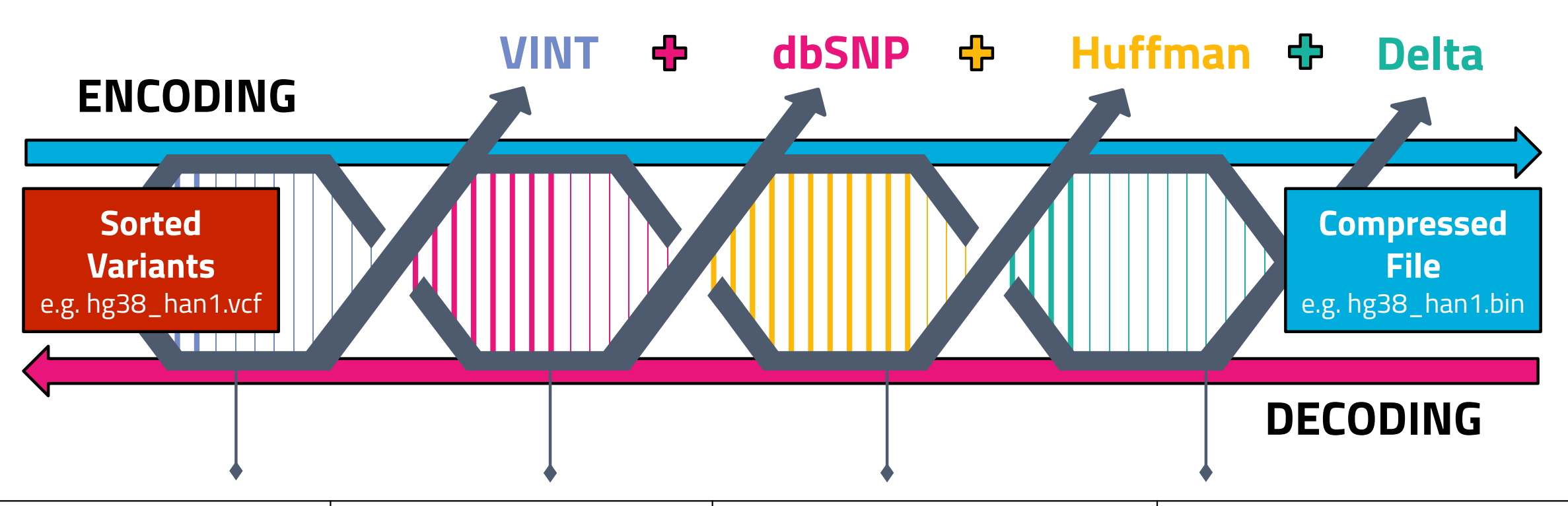


Figure 2. Creation of the VCF utilized **MUMmer4** [5] (genome alignment tool), a seed-extend algorithm that finds reference/target nucleotide matches and then dynamically identifies the DNA variations and **all2vcf** [6], a VCF conversion tool.

The processed VCF file is **compressed** by DNAZip through the combination of several key methods: **VINTs** (variable integer lengths), **SNP mapping**, **Huffman coding** of insertions, and **delta position encoding**.

Methodology



Variable Integers	dbSNP	Huffman Coding	Delta Positions
Store only the necessary number of bytes to represent any given integer.	SNVs found in both the VCF file and SNP database are encoded as true (1), otherwise as false (0) bits.	Entire insertion sequence is broken into *k*-mers. Used to create a Huffman frequency table, tree, and encoding map.	Positions are stored as the difference between the current position and the previous position.

Figure 3. DNAZip encoding of a VCF consists of four key parts: VINTs, dbSNP, Huffman Coding, and Delta Positions. Decoding reconstructs the original data by reversing each part.

8-mer	*Bitstring*	*Length*
CTACATAT	000000000000000	15
CCTAAGGT	11111111100001111	17
GCAATCGC	11111111111011110101	20
CGCGTACA	1111111011011100100000	22

Table 3. Sample *k*-mers (*k* = 8) from the Huffman encoding map. Each distinct *k*-mer has a unique bitstring. Each bitstring is encoded into the compressed file when the matching *k*-mer has been found. Ranked by frequency from top to bottom. The most frequent *k*-mer has the shortest bitstring.

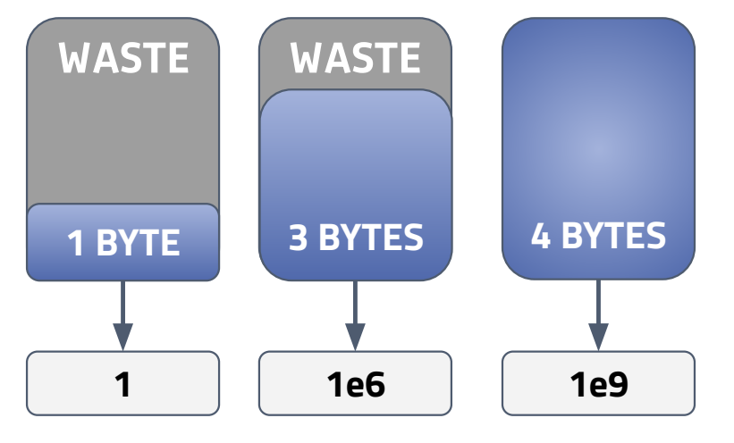


Figure 4. VINT representations only require the least number of bytes to store any integer.

DNAZip **encodes** all **integers** in a VCF—including variant positions, bitstring lengths, and deletion lengths—as **VINTs**. For each chromosome, mapped SNPs are stored in a bitstring, while unmapped SNVs are encoded using a two-bit nucleotide representation [2].

Insertions are compressed with **Huffman coding**: all insertion sequences of a chromosome are first concatenated, then partitioned into 8-mers, and finally frequency-analyzed to assign **shorter bitstrings** to the most **common** 8-mers. Bitstring lengths are recorded per variant type to support decoding.

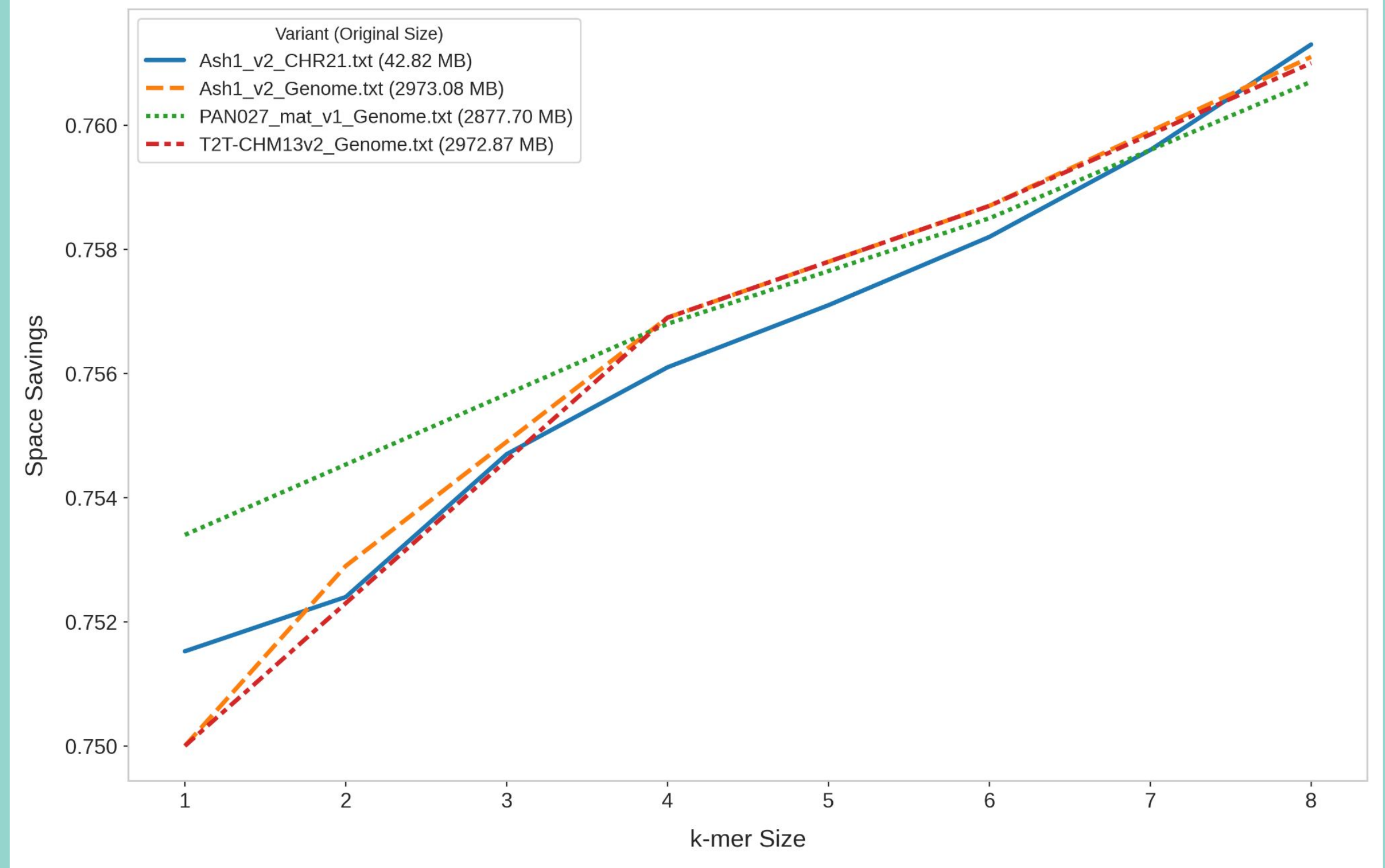


Figure 5. Increasing *k*-mer sizes when Huffman coding on several genomes resulted in higher space savings, or more effective compression. 8-mers provided the greatest space savings, so a k-mer of size 8 has been chosen for DNAZip.

After encoding each variation type of each chromosome, the final **bitstring** is then **stored** in the final compressed file.

Decoding reads the compressed file into a bitstring and uses the stored bitstring lengths to determine how far to advance through each variant type and chromosome.

Each compression feature can be individually enabled or disabled to quantify its contribution to overall VCF size reduction.

Results

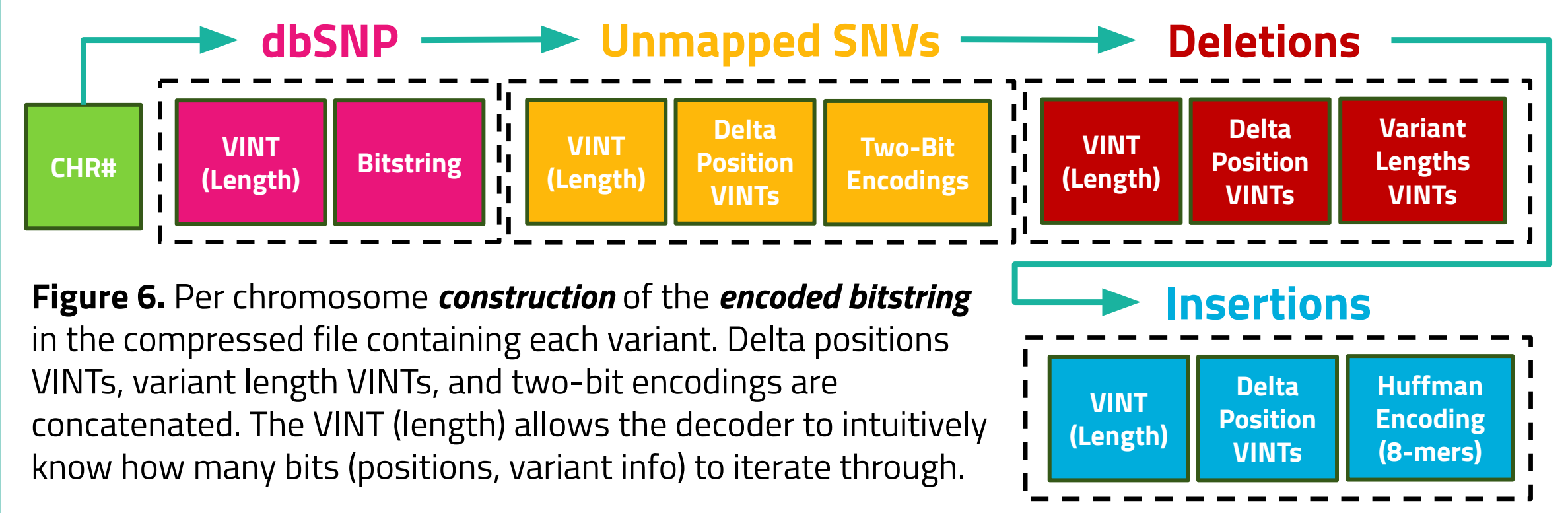


Figure 6. Per chromosome ***construction*** of the ***encoded bitstring*** in the compressed file containing each variant. Delta positions VINTs, variant length VINTs, and two-bit encodings are concatenated. The VINT (length) allows the decoder to intuitively know how many bits (positions, variant info) to iterate through.

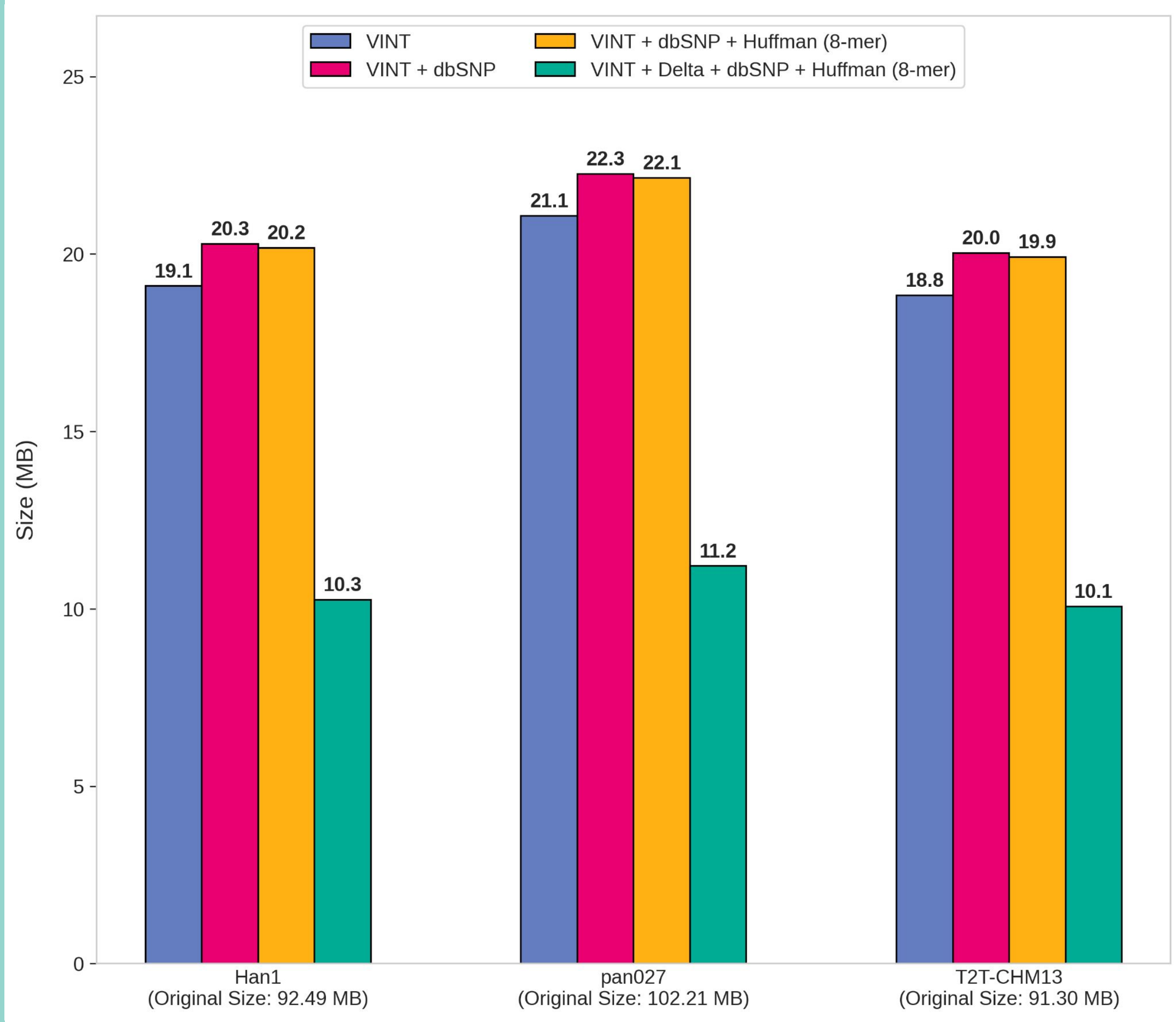


Figure 7. Compressed size depending on the activated compression features. dbSNP adversely contributed to compression by increasing the size of the end file. Huffman coding marginally improved compression. Delta positions provided the largest decrease in compressed file size.

Adding dbSNPs degraded compression performance. As this DNAZip implementation included only Common SNPs, very few SNVs in the target genomes actually mapped to entries in dbSNP. As a result, the bitstring grew without providing meaningful variation data, needlessly inflating the output.

In contrast, **delta-encoding** variant positions yielded the largest compression gains, **reducing** the encoded file **size** by nearly **50%.** This improvement stems from the large absolute genomic positions: encoding positional differences dramatically lowers the magnitude of these values, allowing the **VINT** representations to operate much more **efficiently** and occupy less space, greatly improving compression performance.

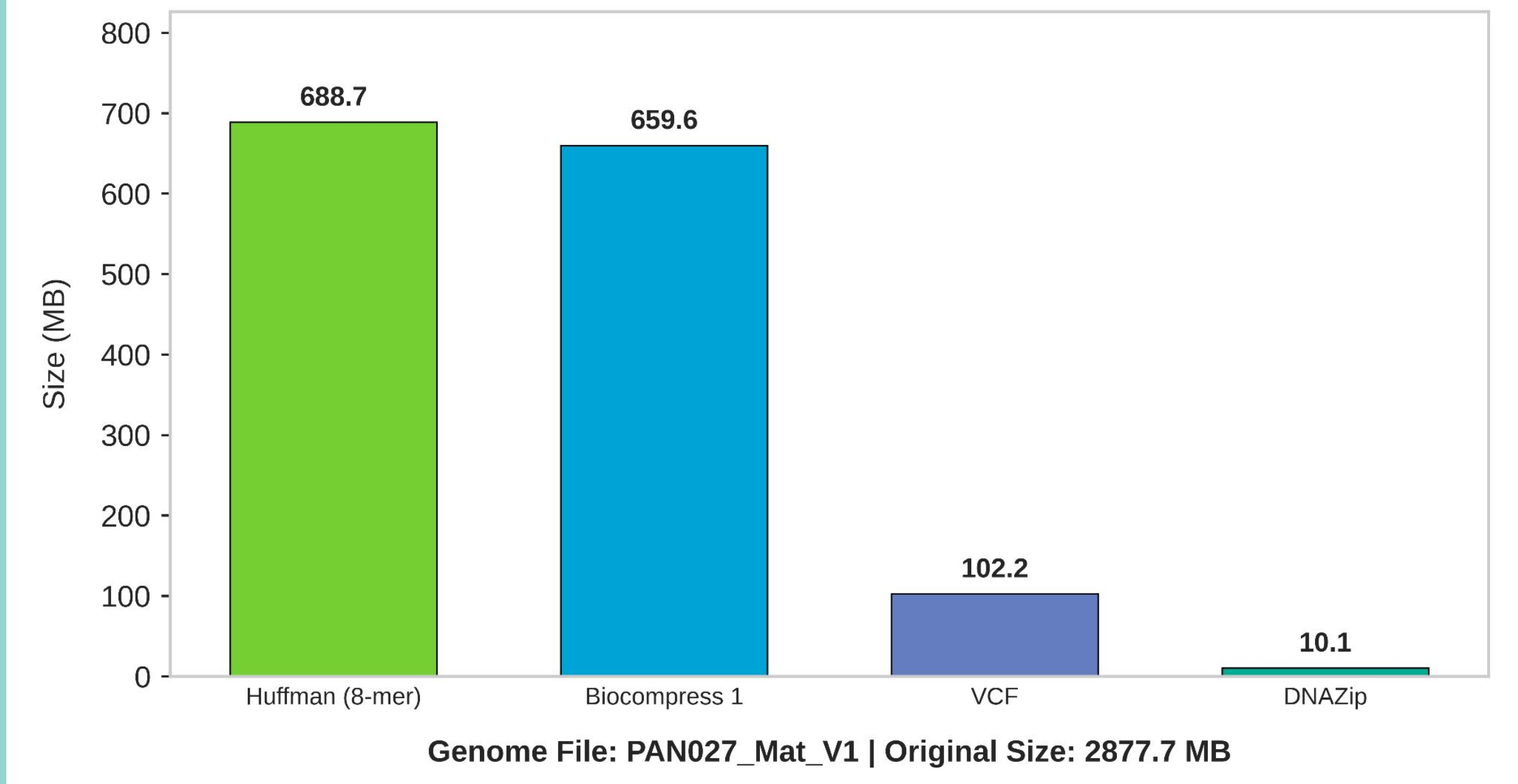


Figure 8. Overall compressed genome sizes across four types, Huffman coding (8-mer), Biocompress 1, uncompressed VCF, and DNAZip compression of the VCF. DNAZip has the smallest end file size, whereas Huffman has the largest.

Conclusion

Reference-based tools like DNAZip demonstrate exceptional scalability by efficiently representing genomic differences instead of entire sequences.

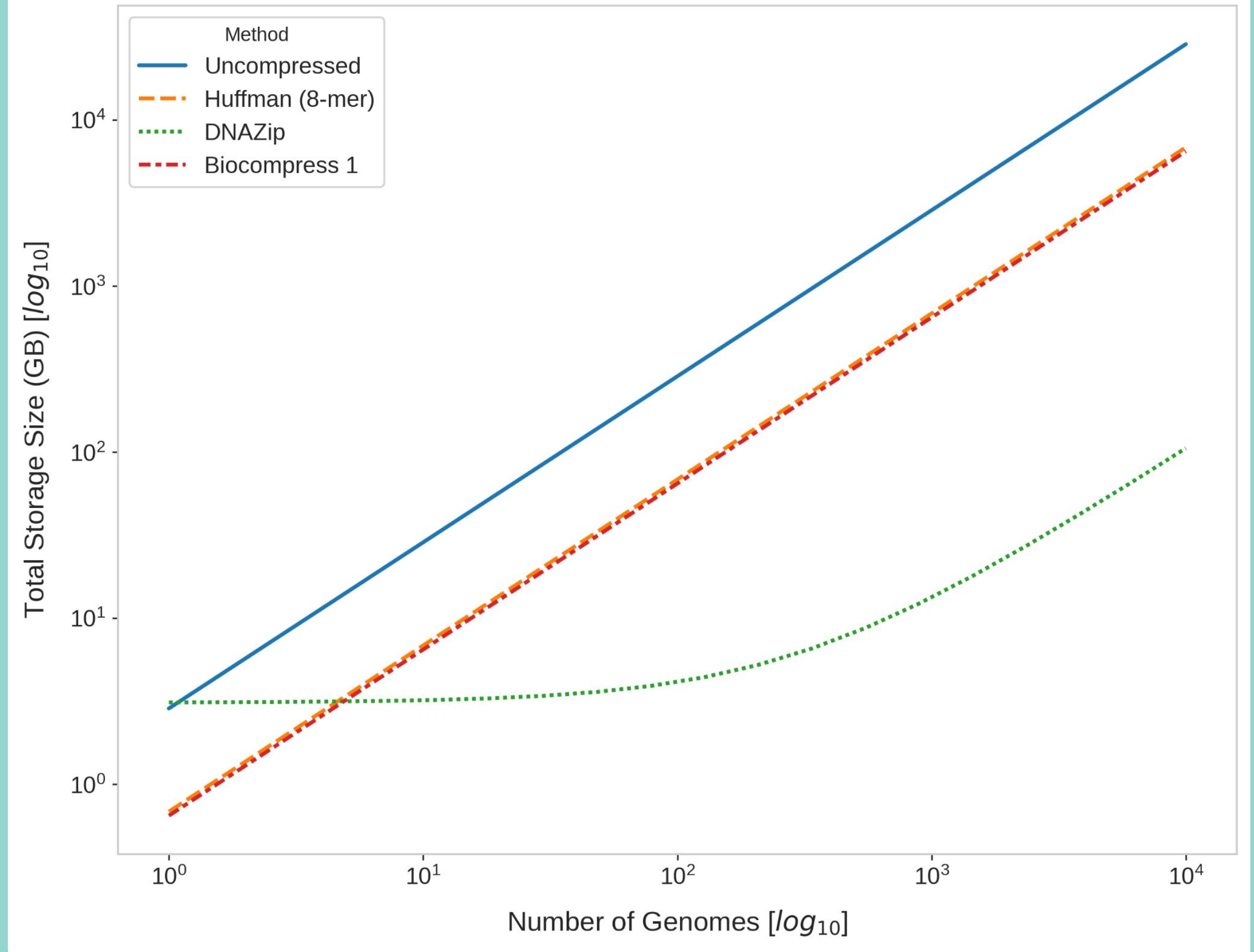


Figure 9. Linear growth of each algorithm. Starting storage sizes are equivalent to a single file encoded by each algorithm, with the exception of DNAZip including the uncompressed size of both its dbSNP and reference genome (~3.5 GB).

While DNAZip offers **superior** space **savings** compared to its non-reference-based counterparts like Biocompress. It also introduces additional computational complexity by **requiring** both **pre-processing** of the **target** genome (alignment for VCF creation) and storage of **reference** data.

Overall, referential compression systems like DNAZip represent a crucial advancement toward **scalable**, **cost-effective** genomic data management in the era of high-throughput **sequencing**.

Future Work

Test larger dbSNPs to determine whether higher mapping rates justify the storage cost of a substantially expanded SNP database and identify the number of human genomes where this tradeoff becomes beneficial. Additionally, more modern reference-based compressor would be explored and implemented to test their performance against DNAZip.

Acknowledgements

I would like to thank my amazing group members for their incredible contributions, Technical Director Mike Tie for his technical assistance, and Professor Layla Oesper for her invaluable guidance and support.

References

[1] F. L. Genomics and L. Fletcher, "The $100 Genome: Where's the Limit?," Front Line Genomics. Accessed: Oct. 05, 2025. [Online]. Available: https://frontlinegenomics.com/the-100-genome-wheres-the-limit/
[2] S. Christley, Y. Lu, C. Li, and X. Xie, "Human genomes as email attachments," Bioinformatics, vol. 25, no. 2, pp. 274–275, Jan. 2009, doi: 10.1093/bioinformatics/btn582.
[3] S. Grumbach and F. Tahi, "Compression of DNA sequences," in Proc. Data Compression Conf., 1993, pp. 340–350, doi: 10.1109/dcc.1993.253115.
[4] D. A. Huffman, "A method for the construction of minimum-redundancy codes," Reson, vol. 11, no. 2, pp. 91–99, Feb. 2006, doi: 10.1007/BF02837279.
[5] G. Marçais, A. L. Delcher, A. M. Phillippy, R. Coston, S. L. Salzberg, and A. Zimin, "MUMmer4: A fast and versatile genome alignment system," PLoS Comput Biol, vol. 14, no. 1, p. e1005944, Jan. 2018, doi: 10.1371/journal.pcbi.1005944.
[6] M. Schiavinato, "Matteoschiavinato/all2vcf: Toolkit to convert the output of common variant calling programs to VCF," GitHub, https://github.com/MatteoSchiavinato/all2vcf (accessed Oct. 9, 2025).