Compressing Genome and Metagenome Sequences

Badhan Das

September 16, 2022

As the research is expanding in Computational Biology and Bioinformatics field, the count of sequencing data from samples has increased drastically, including genome and metagenome sequences. The raw sequenced data are saved in fasta or fastq format. As this count has increased in the last few years, any project working with genome or metagenome sequences requires considerable space to do its experiments computationally. Again, the databases that keep these sequences need an enormous amount of space to keep these data. Since research works in this field will extend significantly, it is high time we focused on how to compress these sequence files more efficiently to organize those better for our research purpose.

Many file compression methods exist currently and are vastly used for many purposes, including zipping fasta/fastq files. When a file or a group of files is compressed, the resulting archive takes 50% to 90% less disk space than the original ones. Some common types of file compression are zip, gzip, tar, fz, rar, 7z, etc. To compress fasta/fastq files, typically, fz and gzip are used. Each of these compression methods uses a unique algorithm to compress the data. While each compression algorithm is different, they all work similarly. The goal is to remove redundant data in each file by replacing common patterns with smaller variables.

While these compressing tools similarly handle any possible file type to compress, it will be beneficial if a particular type of file can be compressed more from a granular level. In this regard, our interest is in the genome (nucleotide) and metagenome sequences that are typically saved as fasta files. This sequence has a unique format; the whole sequence consists of only four letters: A, T(U for RNA), C, and G. This essential yet very crucial information can help to think deeper about compressing sequence files at a granular level. Here we propose a new coding system to convert the genome sequence to one smaller. To better understand the proposed one, some existing coding techniques can be briefly discussed to understand better. The octal number system uses the numbers $0-7$. While converting a number from binary to octal, we take 3 bits and replace them with the equivalent octal digit. This can be viewed in such a way that we are reducing the length of the binary string by a factor of 3. It is the same logic for the hexadecimal number system, but instead of 3, it reduces the length of the binary string by a factor of 4.

Here propose a coding system that uses a set, S, of any 16 English alphabets skipping the nucleotides A, T, C, and G (for RNA, skipping A, U, C, and G, or for both, T and U can be skipped). Let $N = A, T, C, G, U$ be the set of nucleotides. For each nucleotide in the sequence, there can be 4 options: A, T(U), C, and G. So, for two adjacent nucleotides, there will be 16 possible permutations: AA, AT, AC, AG, TA, TT, TC, TG, CA, CT, CC, CG, GA, GT, GC, and GG. These 16 permutations can be encoded to each element of S. For clarity, let us consider $S = B, D, E, F, H, I, J, K, L, M, N, O, P, Q, R, S$. Due to this encoding, the sequence size will be reduced by 2. Although it is very straightforward for a sequence having an even length, for an odd length sequence, some other steps need to be done while encoding. For an odd-length sequence, if the encoding starts from the leftmost nucleotide of a sequence, then the rightmost one will be left alone. This alphabet can be left alone and will not create confusion since S does not include any elements of N. So, while decoding, an alphabet other than the elements of S, or to be more specific, an element of N, definitely will mean that the given sequence is of odd length, and the alphabet will remain as it is. This encoded version can be further compressed using compression methods, resulting in a space-efficient compressed file.

While the existing compression methods generically work for most file types, knowing fasta files' components helps compress them further. The encoding and decoding algorithm can be implemented on some fasta files to see whether it follows this theory empirically.