Data Structures: Cooperative Compression and Storage of Genomic Information

BI278 – Professor Noh
Ben Webb

Abstract

Genomic data that is being sequenced mandates thought towards the development of data structures and compression algorithms that can efficiently store genomes. Compression algorithms exploit redundancy in data to encode information more concisely but require a key to decode and decompress the information. The *de novo* genome compression algorithm, Genomic Squeeze pairs quality scores with nucleotide bases to create novel symbols based on homologous frequencies, which are more entropy efficient than the storage of the two bits separately. Reference based compression implicitly provide information that can be drawn from the alignment required for the compression. After mapping and compressing to a reference sequence, CRAM creates a compression framework with the unmapped reads which is used to map homologous variants. It is an effective reference-based compression for raw sequences within phylums lacking significant prior research and databases. GenomeZip achieves the maximum compression of all of the algorithms examined. It's compression efficiency largely reliant on significant public reference databases for reference-based compression of variants on to the public variant databases. Each compression algorithm is narrow minded in the scope to which solutions are provided but provides a solution to a specific problem or underutilized supplemental information. As the knowledge of which information is important to preserve and sequencing confidence of genomes increases, compression algorithms will adapt to answer new problems and utilize new resources to be as entropy efficient as possible.

Introduction

With the advent of sequencing genomes, biologists have been presented with a novel challenge in the efficient decoding and storing of genomic sequences. The genetic code is more complex than any system we have to store it in. Binary, the basis of all computer logic, relies on either a one or a zero at a fixed bit position along a long line of bits. Binary however, is relatively inefficient at ASCII encoding (A-z, 0-9, etc.) because it requires eight bits to represent a single character. Moreover, by requiring more than one bit to represent a datum, supplemental bits are also needed to indicate the start and stop of each unit of datum (Brandon et. al 2009). Quantum computing, which is still far from fully functional, presents the novel ability to add a third state to each bit (quanta spin), improving upon the storage capability of binary. Genetic code however, can be one of four states at each position, an efficiency that contemporary storage systems that we have control over cannot achieve. The effective storage of information degrades quickly when we can no longer store one unit of datum in one bit.

The effective storage of genomes is further confounded by the inability to perfectly read each base during sequencing. To keep this information requires a secondary table that includes the confidence for each positional read in the sequence as well as other potential deviations. Moreover, erroneous issues such as the phosphorylation of bases allows genetic code to periodically exist in even more than four states at a position (Tembe et. al 2010) requiring additional supplemental tables. These complication presents computer scientists and biologists alike with the same problem of storing and protecting the privacy of the information found in genomes efficiently and effectively while still allowing functional transmission and querying of this incredibly valuable data. The whole of genomic data is most efficiently stored in its truly raw physical state prior to sequencing. By sequencing a genome, we are processing the raw state of each nucleotide in a sequence, knowing that the observed sequenced form is may not be the true state.

It is important to ponder how we can use computers to store genomic information in the long term in an entropy efficient fashion. Moore's law estimates that we are approaching a point where our ability to store information will be surpassed by the accumulation of data. In 2005, the rate of sequencing doubled in just 18 months, meanwhile the rate of increase in disk space doubled over 24 months (Brandon et. al 2009). To prepare for the time when the quantity of data accumulated in the system exceeds the available storage, genomic databases have three options: throw away some of the data, add storage in hopes of staying ahead of the data of our future, or compress the stored data even more. Throwing away data is an easy answer because we have the ability to resequence nearly any sample at any time (Fritz M.H. et. al 2011) without necessarily first determining which data is worth keeping. Adding storage somewhat ignores the notion that erroneous data has been and will continue to be recorded. To compress data, there are a finite number of ways to store and query information and an optimal algorithm for any optimal compression method (Kolmogorov 1963). The storage and query are maximally optimized when they address the major issues of current knowledge while exploiting novel discoveries about genome composition and content.

There are two general groupings for the storage of genomes. The first is *de novo* compression which relies on the compression of the genome within itself. The second is reliant on storage of individual sequence reads against a reference genome. In this second approach, only the unique departures from the reference genome sequence are stored and all the redundant information among the sequences are stored only once in the reference sequence. The goal here is to identify current limitations and additional resources or known features regarding genomic sequences to track how those are ameliorated or utilized during genome compression. Below, I compare different compression structures for genomes by evaluating the success of the storage as a whole, that is the degree of compression of data is as important as the additional assumed resources available as well as speed and complexity of decoding it.

Methods

Genomic Squeeze (G-SQZ)

Genomic Squeeze created by Tembe et. al, is a *de novo* compression method focusing on improving two-bit compression without a reference genome by combining raw positional reads with the quality score for each position. This is done by first reading through the entire sequence and scores. Huffman codes are generated for each <base, quality> score pairing based on the frequency of successive pairs. The encoded pairs are then written in one merged table in order. Currently, the nucleotide base at a given position is as important information as the corresponding quality of the read. G-SQZ merges the coupled information which in turn creates more skewed variance allowing concise encoding. The paper by Tembe et. al was used on Scopus to find the number of citations to estimate the number of uses of G-SQZ there have been.

CRAM

CRAM is built upon the proof of concept principle reference-based alignment by Fritz et. al with the idea that compression should be focused such that a new read, identical to the reference sequence, will have minimal impact on storage regardless on their length or depth of coverage. To do this, reads are first aligned to the reference sequence. From there, the gaps between the unmapped reads are stored then assembled using *de-novo* assembly among all unmapped reads, creating a compression framework in which sequences that did not match the reference genome but are homologous within the framework can be mapped. Scopus citations were of the paper by Fritz, which is the paper that most CRAM was heavily based on as there is no formal paper for CRAM.

GenomeZip

GenomeZip, proposed in a paper by Pavlichin et. al takes a compression approach that attempts to compress all supplemental information from the sequencing reads to publicly accessible databases. After the alignment to a reference genome, the gaps between variations are stored using the delta positioning from the previous variation position. The collection and storage of all variations are all then subdivided. The Single Nucleotide Polymorphism (SNP) positions are encoded using delta positioning relative to the last SNP with the distance recorded using Huffman coding. SNPs are common base substitutions at a given position throughout a population. If the base is an alternate allele of the SNP, the base will be unmapped to the reference sequence. The SNPs are aligned to the NCBI dbSNP database which serves as a reference sequence of all known SNPs for another reference-based compression. The insertion and deletions positions are merged to result in a single distribution with delta distances between each indel. The length and content for both rely on Huffman coding with insertions also storing the sequence that was inserted in a concatenated sequence of all insertions in order of occurrence. GenomeZip usage was also estimated by finding the number of citations on Scopus using the paper by Pavlichin et. al.

Results

All of these compression methods work to parse the information that is deemed valuable. G-SQZ and GenomeZip both rely on Huffman coding which is entropy efficient encoding based on repeated sections of data (Huffman 1952). Both use Huffman coding for different purposes however. G-SQZ uses it to have the most common <base, quality> read with the shortest binary representation. This approach relies on the large number of possible combinations of base and quality score, rewarding more common combinations to be encoded with fewer bits. GenomeZip uses Huffman coding for the grouped together indels. It separately uses Huffman coding distances between non-haplotype SNPs. By compressing SNPs against a dbSNP reference sequence, GenomeZip decreases its size by another 15-35% depending on the sequence. CRAM, unlike GenomeZip uses Golomb coding to code for distances between SNPs.

	Number of Citations	Compression Method	Compression Ratio
Genomic Squeeze	59	*de novo*	2-fold

CRAM	160	Reference Based	50-fold
GenomeZip	15	Reference Based	1200-fold

Table 1. Citations of the primary paper for each compression algorithm as well as the general method that the algorithm uses and the upper threshold of compression level.

G-SQZ and GenomeZip were more often than not, cited among other compression methods as an introduction to the vast array of compression algorithms. That being said, Nicolae et. al looked at G-SQZ in the context of other compression methods that combine quality score with DNA bases to create new combined symbols. While they commended the value of creating domain specific compression algorithms, the lack of using more than a single position when creating <position, score> pairs or palindromic words meant that G-SQZ was not as domain specific as it could have been. The paper by Fritz, used to estimate the number of citations for CRAM, is commonly cited as the foundational concept behind reference-based compression thus citations are not necessarily specific to CRAM, in fact only 13 of the 160 papers mentioned CRAM. According to Cochran et al., the driving factor behind its prevalence of usage for CRAM is due to the interoperability between it and BAM. For that reason, it is the primary compression tool used by the European Nucleotide Archive and the most widely used format for raw sequence storage. Deorowicz et. al viewed GenomeZip as a hybrid between reference-based compression and variant call format based on the extreme compression methods due to the novel methods focusing on the compression of variants relative to known variance.

Discussion

A divergence between purely compression algorithms and a majority of other genomic algorithms is that compression does not need to be biologically correct or mimic a naturally occurring process, it simply needs to provide efficient compression of the information being stored. Each compression method works to parse and compress the information that is deemed valuable. Which information is categorized as valuable dictates the methods of the compression algorithm, therefore the direct comparison of size can be a misnomer with divergent information that is compressed. Each of the compression methods come in with different assumptions about the secondary information about the genome, as well as what information is important to store.

Genomic Squeeze is the only *de novo* compression method that was examined in this study. The primary solution it provides is viewing the base and the quality score as codependent and unequally distributed across all possible combinations of base and quality. This is a solution to current sequencing issues, where confidence of knowledge is bound more by sequencing error than by biological phenomena. It is known that there is not an equal distribution of nucleotide bases throughout a genome, nor is there an equal distribution of quality scores. By combining the two, the skew seen in both is exaggerated which allows more efficient compression of data. It also benefits by having the table to translate between the compressed and decompressed version local to the read, thereby reducing the chances of losing the ability to decompress the sequence. In general, *de novo* compression is most effective when there is a lack of metadata of reference phylogenetic knowledge.

Reference based compression assumes that referential knowledge is so widespread that the same reference sequence is implicitly available at both the sender and receiver (Patro 2015). For that to be true, someone must then be responsible for providing and maintaining the reference genome. The information being stored is only the deviations or anomalies relative to the sequences. This approach is extremely efficient by only storing the unique anomaly data; however, there is a risk in storing genomic sequences because if the reference genome is updated or changed in any way, all sequences that have been compressed using the reference this sequence will be lost forever if they are not decompressed and decoded first. While this method can compress alignments effectively, there is little compression of the original read. This creates another confounding issue when comparing effective size of compression because the size of the compression does not include the space taken up by the reference sequence, but all almost of the compression is in the context of the reference genome.

The ingenuity of reference-based compression is based on the concept that human genomes are 99% identical (Christley et. al 2009). By mapping and compressing more than one genome to a reference sequence, the dataspace consumed by the compression of the the mapped sequences without a reference genome would exceed that of the compression with a reference genome. 76% of Watson's genome dataspace was devoted to storing SNPs when compressed to the consensus genome. This is because we need to store the both the position of the nucleotide as well as the value of the polymorphism (Pavlichin et. al 2013).

GenomeZip addresses SNP compression by using the NCBI dbSNP database to align and compress using the database as a reference sequence. The ease of compressing to an dbSNP database is that a large majority of SNPs only exist in two separate forms which only take up one bit per SNP (Christley 2009). That being said, the cost of storing a reference genome and dbSNP database is amortized for other human genomes in large scale storage. When there is a catalogue of SNPs, GenomeZip is highly effective and can achieve up to 1200-fold compression. The only SNP database with enough coverage is the SNP database for humans. The 1000 genomes project was created with the goal of identifying 95% of all SNPs with a frequency over 1% across all human populations. For this reason, when the sequence is of a well-studied model organism with well curated databases, compression using GenomeZip is incredibly effective. The project also was able to identify 1.4 million biallelic indels as well as 14,000 large deletions. GenomeZip however does not use these identified indels as a reference framework for further compression.

The 1,000 genomes project genotyped 1,092 individuals. Even then, only 82% of the unmapped reads from Watson's genome were able to be mapped within the dbSNP (1000 GPC et. al 2012, Pavlichin 2013). CRAM builds upon the compression method BAM, through extended and unique compression of the unmapped reads and SNPs. By creating a compression framework of all unmapped reads, homologous unmapped reads are mapped within the compression framework, decreasing the final size of the file by 40% when compared to BAM. This method compression of the unmapped read then only requires a single reference sequence, or a single sequence to compress on, whereas GenomeZip relies heavily on the widespread accumulation of consensus genomes and SNPs found across many sequences.

Conclusion

Development of compression algorithms for genomic data is multifaceted in the benefits provided. Compression historically has been significantly an ameliorative action when thinking about dataspace usage. We are readily approaching an upper limit for the amount of data that is stored electronically. While there a multitude of lossless compression techniques, there is also a lot of extraneous data stored due the lack of distinction between valuable data and metadata stored to avoid erroneous conclusions. The development of novel compression techniques should take advantage of the increased reliability of the information that is documented by limiting the focus to information that is viewed as valuable.

Decreasing the amount of information stored through compression or discarding of data is important, but the compression of data in itself is not necessarily relevant or productive for genomic data at this moment. By first aligning a sequence, compression allows for easy comparison between sequences, as only the divergent part of referenced based compression are stored. One benefit is the knowledge gained during alignment-based compression. While slower than alignment free methods because they must first map sequences to a reference genome, the time taken to align is productive. Most of the conclusions drawn from novel sequences are first seen in alignment with known expression of aligned regions from previously studied sequences.

Due to the deep coupling of alignment and reference-based compression, evaluation of reference-based compression must also include evaluation of the success of the alignment method used. The common alignment algorithm ClustalW runs with a complexity of O(mn), meanwhile frequency-based algorithms run with a linear O(n) complexity (Oliver et. al 2014). It is currently unknown the true scope to which sequence order must be maintained when aligning, and in many ways the value depends on the goal of the alignment in the first place. As the benefits of different alignment methods are evaluated, corresponding compression algorithms can be developed in concert with the strengths of the alignment. Compression methods must acknowledge the symbiotic relationship between binary data and genetic data. The most effective compression of genomic data in binary is more than likely determined by the underlying biologically correct theme.

References

Brandon, M. C., Wallace, D. C., & Baldi, P., 2009, Data structures and compression algorithms for genomic sequence data. *Bioinformatics*, 25(14), 1731-8.

Christley, S., Lu, Y., Li, C., Xie, X., 2009, Human genomes as email attachments, *Bioinformatics*, 25(2), 274–275.

Cochrane, G., Alako, B., Amid, C., Bower, L., Cerdeño-Tárraga, A., Cleland, I., Gibson, R., Goodgame, N., Jang, M., Kay, S., Leinonen, R., Lin, X., Lopez, R., McWilliam, H., Oisel, A., Pakseresht, N., Pallreddy, S., Park, Y., Plaister, S., Radhakrishnan, R., Rivière, S., Rossello, M., Senf, A., Silvester, N., Smirnov, D., ten Hoopen, P., Toribio, A., Vaughan, D., Zalunin, V., 2013, Facing growth in the European Nucleotide Archive, *Nucleic Acids Research*, 41(1), D30–D35.

Deorowicz, S., Danek, A., Niemiec, M., 2015, GDC 2: Compression of large collections of genomes, *Scientific Reports*, 5

Fritz, M. H., Leinonen, R., Cochrane, G., & Birney, E., 2011, Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome research*, 21(5), 734-40.

Huffman, D., 1952, A method for the construction of minimum-redundancy codes, *Proc. IRE*, 40 1098-1102

Kolmogorov, A.,1963, On Tables of Random Numbers, Sankhyā Ser. A., 25, 369–375.

Li, H., Bob Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., 2009. The Sequence Alignment/Map format and SAMtools. *Bioinformatics* 25(16): 2078-2079.

Moore, G. E., 1965, Cramming more components onto integrated circuits. Electronics, 38(8).

Nicolae, M., Pathak, S., Rajasekaran, S. 2015. LFQC: a lossless compression algorithm for FASTQ files. *Bioinformatics*, 31(20), 3276-81.

Oliver Bonham-Carter, Joe Steele, Dhundy Bastola, 2014, Alignment-free genetic sequence comparisons: a review of recent approaches by word analysis, *Briefings in Bioinformatics*, 890–905

Patro, R., Kingsford, C., 2015, Data-dependent bucketing improves reference-free compression of sequencing reads, *Bioinformatics*, 31(17), 2770–2777.

Pavlichin, D. S., Weissman, T., Yona, G., 2013, The human genome contracts again, *Bioinformatics*, 29(17), 2199–2202.

Tembe, W., Lowey, J., Suh, E., 2010, G-SQZ: compact encoding of genomic sequence and quality data, *Bioinformatics,* 26(17), 2192–2194.

Wheeler, D. A., et al., 2008, The complete genome of an individual by massively parallel DNA sequencing, *Nature*, 452, 872-876

1000 Genomes Project Consortium et al., 2012, An integrated map of genetic variation from 1,092 human genomes, *Nature*, 491, 56-65.