BIOINFORMATICS

Vol. 00 no. 00 2005
Pages 1–8

Data Structures and Compression Algorithms for Genomic Sequence Data

Marty C. Brandon[1,2,4] Douglas C. Wallace[2,3,4] and Pierre Baldi[1,2,3*]

[1]Department of Computer Science, UCI, Irvine, CA 92697
[2] Institute for Genomics and Bioinformatics, UCI, Irvine, CA 92697
[3] Department of Biological Chemistry, UCI, Irvine, CA 92697
[4] Center for Molecular and Mitochondrial Medicine and Genetics, UCI, Irvine, CA 92697

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Motivation: The continuing exponential accumulation of full genome data, including full diploid human genomes, creates new challenges not only for understanding genomic structure, function, and evolution, but also for the storage, navigation, and privacy of genomic data. Here we develop data structures and algorithms for the efficient storage of genomic and other sequence data that may also facilitate querying and protecting the data.

Results: The general idea is to encode only the differences between a genome sequence and a reference sequence, using absolute or relative coordinates for the location of the differences. These locations and the corresponding differential variants can be encoded into binary strings using various entropy coding methods, from fixed codes such as Golomb and Elias codes, to variables codes, such as Huffman codes. We demonstrate the approach and various tradeoffs using highly variables human mitochondrial genome sequences as a testbed. With only a partial level of optimization, 3,615 genome sequences occupying 56 Megabytes in GenBank are compressed down to only 167 Kilobytes, achieving a 345-fold compression rate, using the revised Cambridge Reference Sequence as the reference sequence. Using the consensus sequence as the reference sequence, the data can be stored using only 133 Kilobytes, corresponding to a 433-fold level of compression, roughly a 23% improvement. Extensions to nuclear genomes and high-throughput sequencing data are discussed.

Availability: Data is publicly available from GenBank, the HapMap Web site, and the MITOMAP database. Supplementary materials with additional results, statistics, and software implementations are available from `http://mammag.web.uci.edu/bin/view/Mitowiki/ProjectDNACompression`.

Contact: pfbaldi@uci.com,mbrandon@uci.edu

1 INTRODUCTION

As high-throughput genome sequencing technologies continue to improve, genome sequence data continue to accumulate at an exponential pace. Not only do we already have the genome sequence of thousands of viruses and bacteria and dozens of multicellular organisms from plants to humans, but we are rapidly approaching the stage where sequencing individual diploid human genomes will be economically affordable. The first diploid human genome sequences were recently produced (Levy *et al.*, 2007; Wheeler *et al.*, 2008; Wang *et al.*, 2008) and a project to sequence 1,000 human genomes in the next few years is under way (Kaiser, 2008). And so is the race for the capability to sequence an individual human genome for less than $1,000 within a few years (Service, 2006). Millions of human genome sequences could be generated within a decade or two.

In addition to the obvious challenges to understand the structure, function, and evolution of genomes, modern high-throughput sequencing methods also raise questions about how to efficiently represent, store, transmit, query, and protect the privacy of sequence information. These questions are further reinforced if one takes into account also progress in synthetic biology and our ability to bioengineer new sequences.

Currently, publicly available genomes are typically stored as flat text files in GenBank, but this approach is unlikely to scale up in many ways. The storage of the diploid genomes of all currently living humans using this simple approach would take "GenBank", without counting headers or any additional annotations, on the order of 36×10^{18} bytes, or 36M Terabytes, an amount difficult to store or download over the Internet, even using standard compression technologies (e.g gzip). And even with the progress that can be expected with Moore's law for storage and networking in the coming years, it is likely that security and privacy issues will require additional layers of protection around genomic data.

Here we develop data structures and algorithms to begin addressing these problems. These data structures allow the compression of genome and other sequences while facilitating certain classes of sequence queries by bypassing classical sequence alignments and dynamic programming algorithms. The approach is demonstrated primarily using a benchmark dataset comprising a few thousand of individual mitochondrial genome sequences. Human mitochondrial sequences provide an excellent testbed for developing and testing efficient data structures and algorithms because, unlike nuclear genome sequences, many thousands of fully sequenced mitochondrial genomes are already available, from a diverse population of individuals. In addition, mitochondrial genome sequences pose unique challenges due to their greater

[*]to whom correspondence should be addressed

variability, as compared to SNP data, resulting from their higher mutation rates.

2 GENERAL APPROACH

In the case of multiple genomes from the same species, associated with "resequencing" technologies, the flat text file approach is clearly wasteful since for the most part the sequences are identical. Thus a simple approach is to store a reference sequence, and then for each other sequence, encode only the differences (or "deltas") with respect to the original sequence. More precisely, consider first the sequences AACGACTAGTAATTTG and CACGTCTAGTAATGTG which are identical, except for a substitution in position 1 (A→C), 5 (A→T), and 14 (T→G). Each SNP can be encoded by a pair (i, X), where i is an integer encoding the position and X represents the value of the substitution relative to the reference. Thus given the first sequence as a reference, the second one can be encoded by the string "1C5T14G", concatenating the coordinates of the locations at which the variations occur and the SNP values at these locations. Note that with this data representation, the questions "Is this sequence different from the reference sequence at position i? And if so how?" are easy to answer. Thus the same data structure that facilitates compact representation, facilitates also efficient information retrieval.

Other events such as deletions and insertions can easily be accommodated in the same general scheme. For a deletion, imagine using two integers (i, l) where the first integer denotes the position where the deletion occurs, and the second integer represents the length of the deletion. Likewise, for an insertion of length l, one can use the encoding $i, X_1 \ldots X_l$ to denote the insertion of $X_1 \ldots X_l$ at position i with respect to the reference sequence.

Although the basic idea is easy to understand, and not new, a precise implementation requires addressing a number of important technical issues. A first observation is that one can use local relative addresses, i.e. intervals, rather than absolute addresses. Using intervals, the above example"1C5T14G" becomes "0C4T9G". With intervals the dynamic range of the integers to be encoded may be considerably smaller than with absolute addresses. The relatively modest price to pay is that intervals must be added to recover absolute coordinates.

A second observation is that if the positions at which variations occur in the population are fixed and form a relatively small subset of all possible positions, then additional savings may result by focusing only on those positions. If in the same schematic example as above, one knew that in the population substitutions can occur only at positions 1, 5, and 14, then one could, for instance, encode "1C5T14G" simply by "CTG", at the cost of keeping an additional table storing the coordinates where the variants occur, and using the letter in the reference sequences at positions where the reference sequence and the sequence under consideration are identical. This approach could be suitable, for instance, for the SNP HapMap data (HapMap Consortium, 2003, 2007), but may not be suitable in other situations, where either the location of all possible variations occurring in the population under consideration is not known in advance, or the number of such locations is very large across the population, but not very large in a typical sequence. This is the case, for instance, of mitochondrial DNA which is characterized by much higher mutation rates than nuclear DNA. Thus different situations may lead to different variations of the basic idea.

An additional technical consideration is the choice of the reference sequence. In particular, the reference sequence does not need to be an actual genome but can, for instance, correspond to a consensus genome. While the resequencing case is of primary interest here due to the medical implication associated with resequencing human genomes, the same general ideas can be applied also to the case of de novo sequencing by using, for instance, the genome of the closest available species as the reference genome.

However, no matter what the detailed scenario is, all applications of the basic ideas hinge on a fundamental technical problem: how encode integers, representing for instance absolute or relative genomic addresses or read lengths, into binary strings. It is essential to understand that the naive idea of converting integers to their binary value, that is converting a "5" to"101" does not work at all since with this encoding one does not know where an integer ends and the next one begins. There are no spaces, tabs, or commas available to separate consecutive integers in the ultimate binary format of any computer where only the symbols 0 and 1 are available. Thus the encoding itself must somehow contain the information necessary to uniquely determine the beginning and end of each information item. In addition, the plain conversion of integers to binary does not take into account any entropy considerations. similarly, a general purpose compression scheme for text data, such as Lempel- Ziv (gzip), is likely to be far from optimal for genome and HTS data. In short, we are interested in binary encoding schemes for sequences of integers that can be parsed automatically and that, consistently with information theory, are entropy efficient, in the sense that fewer bits are used to encode more frequent events. The goal here is not to prescribe a single strategy to achieve this end, but rather to present a family of related coding strategies and some of the tradeoffs that would have to be optimized in a practical application, and illustrate the approach using highly variable mitochondrial DNA.

3 SPECIFIC ENCODING STRATEGIES

To begin with, we illustrate these issues here by considering how the integer positions i are ultimately encoded into a binary string. From Shannon's entropy coding theory (McEliece, 1977; Cover and Thomas, 1991), optimal encoding of these integers from a compression standpoint depends on their distribution in order to assign shorter binary codes to more probable symbols (integers). For simplicity, we distinguish two broad classes of codes: fixed codes, such as Golomb (Golomb, 1965) and Elias codes (Elias, 1975), and variable codes, such as Huffman codes (Huffman, 1952). In a fixed code, the integer i is always encoded in the same way, whereas in a variable code the encoding changes.

3.1 Fixed Codes: Golomb and Golomb-Rice Codes

Both Golomb codes and Elias codes encode an integer j by catenating two bit strings: a preamble $p(j)$, that encodes j's scale, and a mantissa. Golomb codes were specifically developed to encode stationary coin flips with $p \neq 0.5$. Thus they are known to be optimal and asymptotically approach the Shannon limit if the data is generated by random coin flips or, equivalently, if the distribution over the integers is geometric, although they can be used for any other distribution. The more skewed the probability p is (towards 0 or 1) the greater the level of compression that can be achieved.

Golomb codes have one integer parameter m. Given m, any positive integer j can be written using its quotient and remainder modulo m as $j = \lfloor j/m \rfloor + (j \mod m)$. To encode j, the Golomb code with parameter m (Table 1) encodes the quotient and remainder by using:

- $\lfloor j/m \rfloor$ 1-bits for the quotient;
- followed by a 0, as a delimiter (unary encoding of $\lfloor j/m \rfloor$);
- followed by the phased-in binary code for $j \mod m$ for the remainder (described below).

The encoding of integers $0, \ldots, m-1$ normally requires $B = \lceil \log m \rceil$ bits. If m is not a power of two, then one can sometimes use $B - 1$ bits. More specifically, in the "phased-in" approach:

- if $i < 2^B - m$, then encode i in binary, using $(B-1)$ bits;
- if $i \geq 2^B - m$, then encode i by $i + 2^B - m$ in binary, using B bits.

For instance, for $m = 5$, $i = 2$ is encoded as "10" using 2 $(= B-1)$ bits, and $i = 4$ is encode as "111" using 3 $(= B)$ bits (see Table 1). Thus the encoding of j requires in total $\lfloor j/m \rfloor + 1 + \lfloor \log m \rfloor$ or $\lfloor j/m \rfloor + 1 + \lceil \log m \rceil$ bits (Table 1) and the codeword for the integer $j + m$ has one more bit than the codeword for the integer j. Unless otherwise specified, all logarithms are taken to base 2. We use also "$[\log m]$" to denote "$\lfloor \log m \rfloor$ or $\lceil \log m \rceil$".

j	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$
0	00	00	000	000	000
1	01	010	001	001	001
2	100	011	010	010	0100
3	101	100	011	0110	0101
4	1100	1010	1000	0111	0110
5	1101	1011	1001	1000	0111
6	11100	1100	1010	1001	1000
7	11101	11010	1011	1010	1001
8	1111100	11011	11000	10110	10100

Table 1. Golomb encoding of the integers $j = 0$ to 8, for different values of the parameter m.

The entropy of the geometric distribution of the coin flip run-lengths is given by (using $q = 1 - p$):

$$H(geometric) = -\sum_{j=0}^{\infty} q^j p \log(q^j p) \quad (1)$$

and provides the optimal Shannon coding lower bound on the expected encoding length l per integer

$$E(l) \approx \sum_{j=0}^{\infty} q^j p \left(\lfloor j/m \rfloor + 1 + [\log m]\right) \quad (2)$$

under the coin flip model. Thus the Golomb code approaches the Shannon limit when $q^m = 0.5$. In particular, this ensures that for each integer j

$$-\log P(j) = \log(q^j p) \approx \lfloor j/m \rfloor + 1 + [\log m] \quad (3)$$

where $P(j)$ is the probability associated with the integer j.

Number	Encoding ($k=2$)	Number	Encoding ($k=3$)
0-3	0xx	0-7	0xxx
4-7	10xx	8-15	10xxx
8-11	110xx	16-31	110xxx
33	11111111001	33	11110001

Table 2. Golomb-Rice encoding of integers $j = 0 - 33$ with $k = 2$ ($m = 4$) and $k = 3$ ($m = 8$). Integer j is encoded by concatenating $\lfloor j/2^k \rfloor$ 1-bits, one 0-bit, and the k least significant bits of j.

Number	Encoding
1	1
2-3	01x
4-7	001xx
8-15	0001xxx
16-31	00001xxxx

Table 3. Elias Gamma encoding. Each integer j is encoded by concatenating $\lfloor \log j \rfloor$ 0's with the binary value of j.

Finally, Golomb-Rice codes are a particularly convenient sub-family of Golomb codes, when $m = 2^k$ (Table 2). To encode j, we concatenate $\lfloor j/2^k \rfloor$ 1-bits, one 0-bit, and the k least significant bits of j. The length of the encoding of j is thus $\lceil j/2^k \rceil + k + 1$. The decoding of Golomb-Rice codes is particularly simple, the position of the 0-bit gives the value of the prefix to be followed by the next k bits.

3.2 Elias Codes

In the Elias Gamma coding scheme, the preamble $p(m)$ is a string of zeroes of length $\lfloor \log j \rfloor$, and the mantissa $m(j)$ is the binary encoding of j. More precisely, to encode the scale and value of j:

- write $\lfloor \log j \rfloor$ 0-bits;
- followed by the binary value of j beginning with its most significant 1-bit.

The length of the encoding of j is $2\lfloor \log j \rfloor + 1$ (Table 3). The decoding is obvious: 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

$$-\log P(j) \approx 2\lfloor \log j \rfloor + 1 \quad (4)$$

to the integer probabilities, shows that Elias Gamma encoding asymptotically approaches the Shannon limit for $P(j) \approx Cj^{-2}$. This is a power law relationship with exponent -2 and C is a normalizing constant. Note that for both Golomb (Equation 3) and Elias Gamma codes (Equation 4), several different consecutive integers can be encoded into a bit vector with the same length, hence the relationships $-\log P(j) \approx \mathrm{length}(j)$ is only approximate with respect to geometric or power-law distributions over the integers. To be more precise, the optimal distribution associated with the Elias Gamma code can be separated into the product of a probability distribution over the length l given by $P(l) = 2^{-l}$ and a uniform distribution over the integers having an encoding of length l given by $P(j|l) = 2^{-l+1}$.

More recently, new families of efficient fixed codes for integers have been developed (Moffat and Stuiver, 2000; Moffat and Anh, 2006; Baldi *et al.*, 2007; Hirschberg and Baldi, 2008), for instance in the case of increasing or quasi increasing sequences of integers, by encoding only the deltas of the preambles. For sequence data, the absolute addresses are increasing, and the relative addresses could be made quasi-increasing if one were to apply a fixed permutation to all the sequences to be stored, at the cost of storing and using this permutation (Baldi *et al.*, 2007).

3.3 Decoding and Byte Arithmetic

While the degree of compression achieved is an important criteria, the complexity and speed of decoding is also important in all the applications to be considered. For all the encoding algorithms described above, we have also described corresponding simple and fast decoding algorithms. Direct implementations of the decoding algorithms process the compressed representations bit-by-bit; however, it is possible to implement even faster decoders, which decode the compressed data byte-by- byte. These faster decoders work by looking up information from pre-computed tables. These tables are indexed by: (1) all possible bytes B (ranging from 0 to 255); and (2) a bit-index i (ranging from 0 to 7) which marks the position of the decoder within the byte. These tables may store quantities such as the binary value of byte B starting from bit i, the number of bits turned on in byte B starting from bit i, and the unary value of byte B starting from bit i. The exact quantities stored depend on the details of a particular decoder implementation. In practice, byte arithmetic considerably increases decoding speed, sometimes approaching as much as an eight-fold improvement over the corresponding bit-by-bit implementation. The exact value of the speedup depends on several factors including the characteristic of the data, the exact compression scheme, and the hardware used.

3.4 Variable Codes

In genomic applications, in general the integers may not have a well defined distribution, in which case it is always possible to use a general entropy encoding scheme, such as Huffman coding (Huffman, 1952; McEliece, 1977; Cover and Thomas, 1991) which essentially builds a prefix code by using a binary hierarchical clustering algorithm starting from the events (integers) with the lowest probability. While Huffman coding achieves compression close to the entropy limit, the price to pay over fixed coding schemes such as Golomb and Elias Gamma, or the more recent codes mentioned above, is the storage of the Huffman table which can be quite large in some applications. However this is a fixed cost with respect to the database size, and therefore whether this cost is acceptable or not depends on the specific application. Small gains in compression over Huffman coding may be obtained using arithmetic coding (Rissanen and Langdonr, 1979; Witten *et al.*, 1987), but at a non-trivial price in the complexity of computations.

4 RESULTS

4.1 Data Extraction

To demonstrate the general approach, 3,615 human mitochondrial sequences were downloaded from a recent version of GenBank. We focused on the sequences alone, ignoring any header and any other exogenous information. We first use the the Revised Cambridge Reference Sequence (rCRS) sequence (GenBank accession number: AC_000021) as the reference sequence (Brandon *et al.*, 2005; Ruiz-Pesini *et al.*, 2007). The reference sequence is 16,568 bp long. Among the other sequences, 2,671 correspond to complete genomes, while the remaining 944 correspond only to the coding region sequence, which is about $\sim 1,100$ bp shorter than the full genome sequence, and extends from position 577 to 16,023 of the reference sequence. 80 sequences contained ambiguous symbols which, for simplicity, were replaced by the corresponding value in the reference sequence. This replacement is without much loss of generality since ambiguous symbols could easily be accommodated into the coding schemes, for instance as additional variation types.

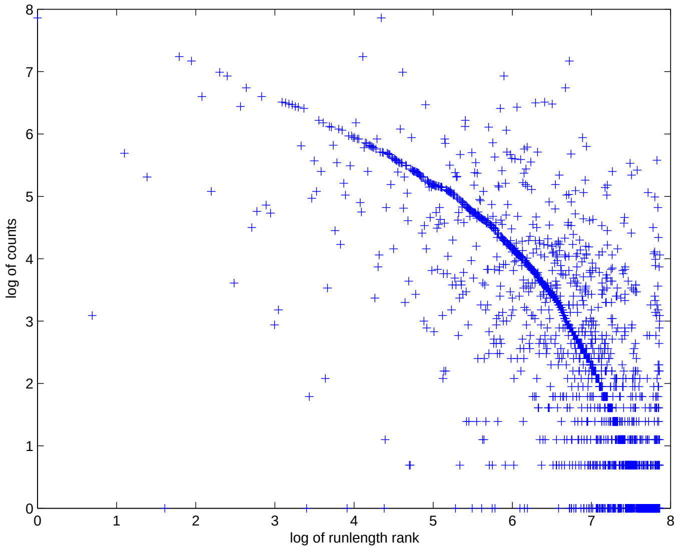

Fig. 1. Distribution of intervals between variations using a log rank-log frequency plot. x-axis represents the logarithm of the rank associated with decreasing interval frequencies. y axis represents the logarithm of the corresponding counts.

4.2 General Statistics

There are 4,577 positions along the reference sequence where at least one of the other sequence deviates from the reference. In aggregate, there are 122,131 bp that deviate from the reference sequence. Besides substitutions, the total number of insertion and deletion events across all the sequences is 7,119, the most frequent one being 1 bp insertions (4,615 occurrences), followed by 2 bp deletions (901). Some well known variants, such as the "Asian-specific 9 bp deletion" (Harihara *et al.*, 1992; Thomas *et al.*, 1998), also occur frequently (255 occurrences). In total, there are 43 different kinds of variation events (see Tables 6 and 7). On average, a given sequence deviates from the reference sequence in 33.8 bp

	Intervals			Variants		
	Huffman	Golomb	Elias Gamma	Huffman	Golomb	Elias Gamma
Cambridge	9.21	11.10	12.93	2.66	2.44	2.77
Consensus	9.75	12.03	13.86	2.44	2.59	2.97

Table 4. Comparison of the average bit cost of encoding intervals and events for Huffman, Golomb, and Elias Gamma encoding schemes using the revised Cambridge Reference Sequence and the consensus sequence.

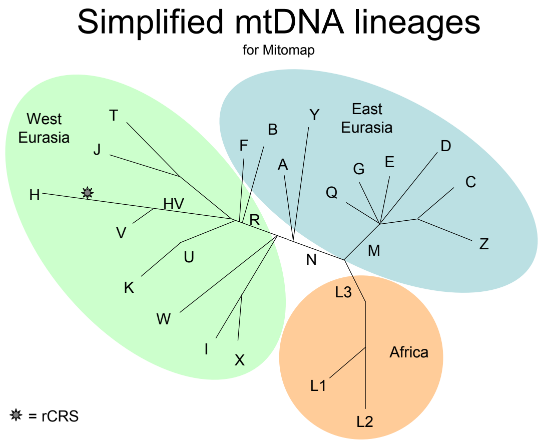

Fig. 2. Simplified Haplotype classification used in Brandon *et al.* (2009). rCRS = revised Cambridge Reference Sequence.

with a standard deviation of 13.43 bp. The average number of substitutions (transition/transversions) per sequence is 30.69 bp. The average number of insertions per sequence is 1.69 bp, the average number of deletions is 1.37 bp.

The distribution of the raw intervals using the rCRS as the reference sequence is represented in Figure 1 displaying the logarithm of the counts versus the logarithm of the rank (in decreasing order of frequency). Observed intervals vary from 0 to 14,997 bp, the most frequent one being an interval of 72 (2,579 occurrences) (see interpretation in next section), followed by 687 (2,418 occurrences), and followed by 5 (2,130 occurrences). Overall this distribution is not strongly structured.

	H+H	G+G	E+E
Cambridge	167 (345)	195 (295)	226 (254)
Consensus	133 (433)	159 (361)	183 (314)

Table 5. Total file size comparison using the rRCS and the consensus sequence, with Huffman encoding for both intervals and variants (H+H), or Golomb encoding for both interval and variants (G+G), or Elias Gamma encoding for both interval and variants (E+E). Numbers are given in Kilobytes (1024 × 8 bits). In comparison, the raw data takes 56 Megabytes (57439.05 Kilobytes). Compression factor are given in parenthesis.

Table 6. Huffman encoding for the event types using the revised Cambridge Reference Sequence.

variant	count	binary code
G	42839	11
C	24753	01
T	22345	00
A	21003	101
InsC	3980	1001
Del2bp	901	100011
Del1bp	757	100001
InsCC	360	1000100
InsT	313	1000001
Del9bp	255	1000000
InsA	222	10001011
InsCCC	34	1000101000
InsCCCC	30	10001010110
InsG	29	10001010100
InsCCCCC	16	100010101110
InsACA	15	100010101011
InsCCCCCC	12	100010100110
InsCCCCCCC	8	1000101010101
InsAC	6	1000101001011
InsCCT	5	1000101001010
Del6bp	4	10001010111100
Del8bp	4	10001010111101
InsCCCCCTCTA	3	10001010011111
Del3bp	3	10001010101001
InsGC	3	10001010011101
InsCCCCCCCC	3	10001010101000
InsTT	3	10001010011110
Del4bp	3	10001010011100
InsACAC	2	100010101111100
InsACACA	1	100010100100000
InsCCCCCCCCC	1	100010100100111
InsTA	1	1000101011111110
InsGA	1	1000101011111101
InsGG	1	1000101011111100
InsCA	1	1000101011111011
InsAG	1	1000101011111010
InsGATCACAG	1	100010100100011
Del10bp	1	100010100100010
InsTCTCTGTTCTTTCAT	1	100010100100001
InsACACAC	1	100010100100101
InsAGAA	1	100010100100100
InsCACA	1	1000101011111111
Del5bp	1	100010100100110

Deletions (Del) are followed by their length. Insertions (Ins) by their content.

Table 7. Huffman encoding for the event types using the consensus sequence.

variant	count	binary code
C	26164	11
A	19576	01
G	18002	00
T	16528	101
InsC	3980	1001
Del2bp	901	100011
Del1bp	757	100001
InsCC	360	1000100
InsT	313	1000001
Del9bp	255	1000000
InsA	222	10001011
InsCCC	34	1000101000
InsCCCC	30	10001010110
InsG	29	10001010100
InsCCCCC	16	100010101110
InsACA	15	100010101011
InsCCCCCC	12	100010100110
InsCCCCCCC	8	1000101010101
InsAC	6	1000101001011
InsCCT	5	1000101001010
Del6bp	4	10001010111100
Del8bp	4	10001010111101
InsCCCCCTCTA	3	10001010011111
Del3bp	3	10001010101001
InsGC	3	10001010011101
InsCCCCCCCC	3	10001010101000
InsTT	3	10001010011110
Del4bp	3	10001010011100
InsACAC	2	100010101111100
InsACACA	1	100010100100000
InsCCCCCCCCC	1	100010100100111
InsTA	1	1000101011111110
InsGA	1	1000101011111101
InsGG	1	1000101011111100
InsCA	1	1000101011111011
InsAG	1	1000101011111010
InsGATCACAG	1	100010100100011
Del10bp	1	100010100100010
InsTCTCTGTTCTTTCAT	1	100010100100001
InsACACAC	1	100010100100101
InsAGAA	1	100010100100100
InsCACA	1	1000101011111111
Del5bp	1	100010100100110

Deletions (Del) are followed by their length. Insertions (Ins) by their content.

4.3 Changing the Reference Sequence

There are no particular reasons, beyond standardization and tradition, for using rRCS as the reference sequence. Furthermore, purely from a compression standpoint, the rCRS may not be optimal due to biases in data. To illustrate this point we computed the haplotype distribution of the data using the simplified haplotype classification described in Figure 2 (see also (Brandon *et al.*, 2009; Mishmar *et al.*, 2003)). We find the following skewed distribution: 11.2% African (405 sequences), 26.3% Asian (950 sequences), and 62.5% EurAsian (2,260 sequences). In addition, it is well known that the original Cambridge Reference sequence contains a number of errors and has been revised over the years (Anderson *et al.*, 1981; Andrews *et al.*, 1999) (The revisions to the original sequence are described at: http://www.mitomap.org/mitoseq.html). This alone, for instance, explains why the interval 72 is so frequent with respect to the rCRS: the rCRS sequence has a G in the corresponding position, which is a very rare variant, most likely an error.

Thus it is clear that other reference sequences could be used to improve compression rates and minimize the total number of variants. Furthermore, the reference sequence does not need to be a sequence from an actual individual, but could be designed using purely statistical considerations. Note that the design of the reference sequence impacts not only the variants to be recorded, but also the intervals, and therefore it must also take into consideration any constraints a particular implementation may place on the intervals and their encodings. A reasonable choice adopted here to try to further improve the compression rate, is to use the consensus sequence, derived by computing the consensus at each position, as the reference sequence.

Using the consensus sequence, observed intervals vary from 0 to 11,717 bp, the most frequent one being an interval of 5 (2,104 occurrences), followed by 1 (1,251 occurrences), and followed by 259 (895 occurrences).

4.4 Encoding and Compression

We explored and compared different encoding schemes using both fixed and variable codes. The main sample of results is given in Tables 4 and 5 giving the average number of bits required to encode an interval or a variant, using Huffman, Golomb, or Elias Gamma codes, with the rCRS or the consensus sequence, as well as the total number of bits required to encode the entire data. The Huffman coding tables for the events are given in Tables 6 and 7 for the rCRS and consensus sequence respectively.

As can be seen in Table 5, Huffman coding achieves slightly better compression rates than Golomb or Elias Gamma coding, with a table storage cost that may be manageable in this case. The raw data takes 56 Megabyte (58,817,584 bytes) of space. By concatenating the Huffman codes for the intervals and the variants (H+H), the encoded data requires only 167 Kilobytes of space, corresponding to a 345-fold level of compression. Using, for instance, Golomb codes for both the intervals and the variants (G+G) requires instead 195 Kilobytes. The choice of the reference sequence has a noticeable effect. Although the average number of bits required to encode an interval or a variant is slightly higher for the consensus sequence (Table 4), this is compensated by a considerable decrease in the total number of variants to be encoded. This is true here even with a consensus sequence that differs from the rRCS sequence by only 11 nucleotides. As shown in Table 5, the same encoding method based on using two Huffman codes (H+H), applied with the consensus sequence, requires only 133 Kilobytes to store the entire data. This corresponds to a 433-fold level of compression, roughly a 23% improvement.

5 DISCUSSION

A simple but general data structure and data encoding approach has been developed for the efficient storage of genomic data.

Encoding	DataSet 1	DataSet 2	DataSet 3
Raw Sequence	133,366,560	353,182,128	8,869,613,600
Flat File	75,525,168	185,536,864	8,396,646,344
Elias Gamma Absolute	358,402 (210.73)	79,281,140 (2.34)	1,373,892,116 (6.11)
Elias Gamma Relative	185,542 (407.05)	27,741,238 (**6.69**)	340,764,564 (**24.64**)
Monotone Value (MOV)	169,664 (**445.15**)	39,528,754 (4.69)	834,672,652 (10.06)

Table 8. Compression of the read addresses information from three HTS experiments (see text). The size in bits for the raw sequence data, the corresponding flat text file format for the corresponding addresses, and the compressed files for different compression algorithms. Elias Gamma coding is applied both to the absolute and relative addresses. Compression factors with respect to the flat text file format are given in parentheses, with top compression factors in bold. MOV is a coding algorithm specifically designed for increasing sequences of integers described in Baldi *et al.* (2007).

The approach specifically leverages homology between sequences and is different from general compression algorithms for text, or compression algorithms for single genome data (Williams and Zobel, 1997; Chen *et al.*, 2002; Behzadi and Fessant, 2005). The approach has been demonstrated on the mitochondrial genomes, where it leads to 2-3 orders of magnitude improvement in data storage. From these compact representations, full sequences can be recovered rapidly using the reference sequence. Furthermore, queries regarding the existence and nature of variants at particular coordinate positions, such as those arising in a variety of applications from medicine to forensics, can be answered efficiently. Additional encryption methods may be applied to these representations to protect the security of both the genomic data and the queries.

The approach has been used for lossless compression, however it could be used also in lossy compression, for instance by ignoring variants that are not medically relevant. The approach is also applicable to other kinds of sequences, such as RNA or protein sequences. While for demonstration purposes we have used a single reference sequence, it is clear that one could cluster the data and use different reference sequences for different subgroups. In the case of mitochondria genomes, for instance, Figure 2 would suggest using at least three different reference sequences. Whether the gain in compression that can be expected for each subgroup, akin to the gain achieved by going from the rCRS to the consensus sequence, is worth the cost of having multiple reference sequences rather than a single one, cannot be answered in generality and depends on the details of a particular application, the number of genomes to be stored coming from each group, and so forth. For future work, the same idea of multiple reference sequences can be extended beyond the storage of genomes within a given species, to the storage of genomes from multiple species by using a phylogenetic hierarchy of reference sequences.

Finally, the approach can be extended to human nuclear genomes and to high-throughput sequencing (HTS) from different technologies and different kinds of experiments. For human SNP variation, data and statistics are readily available (Hinds *et al.*, 2005; Goldstein and Cavalleri, 2005; HapMap Consortium, 2007). A comprehensive list of human SNPs is available from the dbSNP database maintained by NCBI. The current release (version 129) contains about 15 million SNPs. This data can readily be compressed using the techniques described here and additional gains in compression can be achieved by storing separately a fixed table recording the location of all the SNPs and leveraging the skewed distribution of some of the SNP variants. In preliminary experiments, we have achieved compression factors of over 1,000 on the raw HapMap sequence data. Although SNPs account for most of genetic variation events between individuals, a much larger fraction of the genome (in terms of the total number of bases) is involved in larger structural variation events, such as copy number variations (CNV). While there have been studies attempting to derive a preliminary assessment of large-scale genomic complexity and variation (Feschotte and Pritham, 2007; Tuzun *et al.*, 2005), statistics on the frequencies and location of these more complex structural variations in the human genome are still at an earlier stage of development. For instance, comparative analysis of the single diploid genome described in (Levy *et al.*, 2007) “revealed more than 4.1 million DNA variants, encompassing 12.3 Mb. These variants (of which 1,288,319 were novel) included 3,213,401 single nucleotide polymorphisms (SNPs), 53,823 block substitutions (2206 bp), 292,102 heterozygous insertion/deletion events (indels)(1571 bp), 559,473 homozygous indels (182,711 bp), 90 inversions, as well as numerous segmental duplications and copy number variation regions. Non-SNP DNA variation accounts for 22% of all events identified in the donor, however they involve 74% of all variant bases. This suggests an important role for non-SNP genetic alterations in defining the diploid genome structure.” A better statistical understanding of the coding constraints posed by these complex events, and how to encode them, should become possible as more full human genome sequences become available in the coming years (`www.1000genomes.org`).

Regarding HTS data, for illustration purposes here we consider the problem of storing the genomic addresses of the reads from three HTS datasets associated with different HTS technologies. The first data set is obtained from the laboratory of Dr. S. Sandmeyer at UCI and comes from an experiment aimed at mapping retrotransposon Ty3 insertion sites in the yeast genome. It consists of 833,541 sequence reads, all of length 19 bp. The second data set comes from a chromatin immunoprecipitation assay (ChIP-Seq) used to map the in vivo binding site locations of the neuron-restrictive silencer factor (NRSF) in humans (Johnson *et al.*, 2007). It consists of 1,697,991 sequence reads, all of length 25 bp and mapped to the most recent human genome sequence (hg18). The third data set corresponds to a full diploid human genome sequencing experiment for an Asian individual (Wang *et al.*, 2008). This is a very large data set with enough reads to provide 36-fold average coverage, and we utilize the existing mapping of the reads provided by the YH database (Li *et al.*, 2009) to the human reference genome. For illustrative purposes, we report only the results corresponding to the reads associated with chromosome 22. For chromosome 22, there are 31,118,532 reads that vary in length from 30 to 40 bp for a total of 1,108,701,700 bp of sequence data. While complete details of these experiments will be reported elsewhere, Table 8 shows the resulting compression factors which are again in the range of one to three orders of magnitude, depending on the statistical properties of the data sets. The same techniques described here can readily be applied to storing also the length of the reads, the content of the reads, where they differ from the reference genome, their quality, and so forth. Statistical properties of the reads and the underlying HTS technologies, e.g increasing error rates towards the end of the read, can also be exploited to achieve efficient compression. Thus the data structures and compression algorithms described here provide a framework for the management of HTS and genomic data that can be flexibly applied in different environments.

FUNDING

Work supported by NIH Biomedical Informatics Training grant (LM-07443-01), NSF MRI grant (EIA-0321390), and NSF grant 0513376 to PB. NIH grants AG24373, DK73691, AG13154, and NS21378 to DW. We thank K. Daily, S. Sandmeyer, P. Rigor for useful discussions.

ACKNOWLEDGEMENT

REFERENCES

Anderson,S., Bankier,A., Barrell,B., de Bruijn,M., amd J. Drouin,A.C., Eperon,I., Nierlich,D., B.A.Roe, Sanger,F., P.H.Schreier, Smith,A., Staden,R. and Young,I. (1981) Sequence and organization of the human mitochondrial genome. *Nature,* **290** (5806), 457–465.

Andrews,R., Kubacka,I., Chinnery,P., Lightowlers,R., Turnbull,D. and Howell,N. (1999) Reanalysis and revision of the cambridge reference sequence for human mitochondrial dna. *Nature Genetics,* **2**, 147.

Baldi,P., Benz,R.W., Hirschberg,D. and Swamidass,S. (2007) Lossless compression of chemical fingerprints using integer entropy codes improves storage and retrieval. *Journal of Chemical Information and Modeling,* **47** (6), 2098–2109.

Behzadi,B. and Fessant,F. (2005) DNA compression challenge revisited: a dynamic programming approach. *Lecture Notes in Computer Science,* **3537**, 190–200.

Brandon,M., Baldi,P. and Wallace,D. (2009) MITOMASTER: a bioinformatics tool for the analysis of mitochondrial DNA sequences. *Human Mutation, Database Issue,* **30** (1), 1–6.

Brandon,M., Lott,M., Nguyen,K., Spolim,S., Navathe,S., Baldi,P. and Wallace,D. (2005) MITOMAP: a human mitochondrial genome database - 2004 update. *Nucleic Acids Research,* **33**, D611–D613. Database Issue.

Chen,X., Li,M., Ma,B. and Tromp,J. (2002) DNACompress: fast and effective DNA sequence compression. *Bioinformatics,* **18**, 1696–1698.

Cover,T.M. and Thomas,J.A. (1991) *Elements of Information Theory*. John Wiley, New York.

Elias,P. (1975) Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory,* **21** (2), 194–203.

Feschotte,C. and Pritham,E. (2007) DNA Transposons and the Evolution of Eukaryotic Genomes. *ANNUAL REVIEW OF GENETICS,* **41**, 331.

Goldstein,D. and Cavalleri,G. (2005) Genomics: understanding human diversity. *Nature,* **437**, 1241–1242.

Golomb,S.W. (1965) Run-length encodings. *IEEE Transactions on Information Theory,* **12** (3), 399–401.

HapMap Consortium,T.I. (2003) The international HapMap project. *Nature,* **426**, 789–796.

HapMap Consortium,T.I. (2007) A second generation human haplotype map of over 3.1 million SNPs. *Nature,* **449**, 851–861.

Harihara,S., Hirai,M., Suutou,Y., Shimizu,K. and Omoto,K. (1992) Frequency of a 9-bp deletion in the mitochondrial DNA among Asian populations. *Human Biology,* **64** (2), 161–166.

Hinds,D.A., Stuve,L.L., Nilsen,G., Eskin,E., Ballinger,D., Frazer,K. and Cox,D. (2005) Whole genome patterns of common DNA variation in three human populations. *Science,* **307**, 1072–1079.

Hirschberg,D.S. and Baldi,P. (2008) Effective compression of monotone and quasi-monotone sequences of integers. In *Proceedings of the 2008 Data Compression Conference (DCC 08)*. IEEE Computer Society Press Los Alamitos, CA In press.

Huffman,D. (1952) A method for the construction of minimum redundancy codes. *Proc. IRE,* **40**, 1098–1101.

Johnson,D.S., Mortazavi,A., Myers,R.M. and Wold,B. (2007) Genome-wide mapping of in vivo protein-DNA interactions. *Science,* **316**, 1497–1502.

Kaiser,J. (2008) A plan to capture human diversity in 1000 genomes. *Science,* **319**, 395.

Levy,S., Sutton,G., Ng,P.C., Feuk,L., Halpern,A.L., Walenz,B.P., Axelrod,N., Huang,J., Kirkness,E.F., Denisov,G., Lin,Y., MacDonald,J.R., Pang,A.W.C., Shago,M., Stockwell,T.B., Tsiamouri,A., Bafna,V., Bansal,V., Kravitz,S.A., Busam,D.A., Beeson,K.Y., McIntosh,T.C., Remington,K.A., Abril,J.F., Gill,J., Borman,J., Rogers,Y.H., Frazier,M.E., Scherer,S.W., Strausberg,R.L. and Venter,J.C. (2007) The diploid genome sequence of an individual human. *PLoS Biol,* **5** (10), e254.

Li,G., Ma,L., Song,C., Yang,Z., Wang,X., Huang,H., Li,Y., Li,R., Zhang,X., Yang,H., Wang,J. and Wang,J. (2009) The YH database: the first Asian diploid genome database. *Nucleic Acids Res,* **37**, D1025–8.

McEliece,R.J. (1977) *The Theory of Information and Coding*. Addison-Wesley Publishing Company, Reading, MA.

Mishmar,D., Ruiz-Pesini,E., Golik,P., Macaulay,V., Clark,A., Hosseini,S., Brandon,M., Easley,K., Chen,E., Brown,M., Sukernik,R., Olckers,A. and Wallace,D. (2003) Natural selection shaped regional mtDNA variation in humans. *Proc Natl Acad Sci USA,* **100** (1), 171–176.

Moffat,A. and Anh,V. (2006) Binary codes for locally homogeneous sequences. *Information Processing Letters,* **99**, 175–180.

Moffat,A. and Stuiver,L. (2000) Binary interpolative coding for effective index compression. *Inf. Retr.,* **3** (1), 25–47.

Rissanen,J.J. and Langdonr,G.G. (1979) Arithmetic coding. *IBM Journal of Research and Development,* **23** (2), 149162.

Ruiz-Pesini,E., Lott,M.T., Procaccio,V., Poole,J., Brandon,M.C., Mishmar,D., Yi,C., Kreuziger,J., Baldi,P. and Wallace,D.C. (2007) An enhanced MITOMAP with a global mtDNA mutational philogeny. *Nucleic Acids Research,* **35**, D823–D828. Database Issue.

Service,R.F. (2006) The race for the $1000 genome. *Science,* **311**, 1544–1546.

Thomas,M., Cook,C., Miller,K., Waring,M. and Hagelberg,E. (1998) Molecular instability in the COII-tRNA(lys) intergenic region of the human mitochondrial genome: multiple origins of the 9-bp deletion and heteroplasmy for expanded repeats. *Philosophical Transactions of the Royal Society of London - Series B: Biological Sciences,* **353** (1371), 955–965.

Tuzun,E., Sharp,A., Bailey,J., Kaul,R., Morrison,V., Pertz,L., Haugen,E., Hayden,H., Albertson,D., Pinkel,D. *et al.* (2005) Fine-scale structural variation of the human genome. *Nature Genetics,* **37**, 727–732.

Wang,J., Wang,W., Li,R., Li,Y., Tian,G., Goodman,L., Fan,W., Zhang,J., Li,J., Zhang,J., Guo,Y., Feng,B., Li,H., Lu,Y., Fang,X., Liang,H., Du,Z., Li,D., Zhao,Y., Hu,Y., Yang,Z., Zheng,H., Hellmann,I., Inouye,M., Pool,J., Yi,X., Zhao,J., Duan,J., Zhou,Y., Qin,J., Ma,L., Li,G., Yang,Z., Zhang,G., Yang,B., Yu,C., Liang,F., Li,W., Li,S., Li,D., Ni,P., Ruan,J., Li,Q., Zhu,H., Liu,D., Lu,Z., Li,N., Guo,G., Zhang,J., Ye,J., Fang,L., Hao,Q., Chen,Q., Liang,Y., Su,Y., San,A., Ping,C., Yang,S., Chen,F., Li,L., Zhou,K., Zheng,H., Ren,Y., Yang,L., Gao,Y., Yang,G., Li,Z., Feng,X., Kristiansen,K., Wong,G.K.S., Nielsen,R., Durbin,R., Bolund,L., Zhang,X., Li,S., Yang,H. and Wang,J. (2008) The diploid genome sequence of an Asian individual. *Nature,* **456** (7218), 60–5.

Wheeler,D.A., Srinivasan,M., Egholm,M., Shen,Y., Chen,L., McGuire,A., He,W., Chen,Y.J., Makhijani,V., Roth,G.T., Gomes,X., Tartaro,K., Niazi,F., Turcotte,C.L., Irzyk,G.P., Lupski,J.R., Chinault,C., Song,X.Z., Liu,Y., Yuan,Y., Nazareth,L., Qin,X., Muzny,D.M., Margulies,M., Weinstock,G.M., Gibbs,R.A. and Rothberg,J.M. (2008) The complete genome of an individual by massively parallel DNA sequencing. *Nature,* **452** (7189), 872–6.

Williams,H. and Zobel,J. (1997) Compression of nucleotide databases for fast searching. *Bioinformatics,* **13** (5), 549–554.

Witten,I.H., Neal,R.M. and Clearly,J.G. (1987) Arithmetic coding for data compression. *Communications of the ACM,* **30** (6), 520540.