Samantha Woodward
BIOC 218
Winter 2011-2012

A Critical Analysis of DNA Data Compression Methods

Data compression is an area of research that has many applications spanning different technological fields; among the most important of these fields is that of DNA compression. With whole genome sequencing becoming increasingly practical and cost-effective, the need for a superior space-saving algorithm is obvious. There are many research projects and feasible future applications that necessitate having many different sequenced genomes readily available in a form that can be easily decompressed and examined.

A single, uncompressed human genome in the form that is generated by companies such as Helicos Biosciences, Pacific Biosciences, 454, and Illumina can take up as much as 285 GB [6]. This amount of hard drive space usage makes keeping even a few fully-sequenced, uncompressed genomes on the same machine unrealistic, while transferring them between machines over a small bandwidth is out of the question. Data compression is not a new topic, and at first several general-purpose compression algorithms were used to store the DNA in a more efficient manner. Lempel-Ziv, or gzip compression, is a universal compression algorithm that has been used to store genetic data using an adaptive dictionary [10]. Standard encoding like gzip can reduce the space needed per nucleotide in human DNA asymptotically to two bits, since we only have four possible nucleotides. Eventually, algorithms were implemented that took characteristics of DNA into consideration, such as reverse complementation or point mutation, which could compress DNA beyond this previous gold standard. Several algorithms employing this kind of tactic are GenCompress [3] and BioCompress [5].

Hundreds of algorithms have been proposed for DNA compression; this paper aims to explore several innovative advances made in DNA data compression, and the applications of these improved methods. Most of these algorithms operate with one of two purposes: either to simply achieve the tightest compression possible and save storage space on a computer, or compress the data in a fashion that allows for biological inference about either a single genome or multiple genomes. While there can be overlap between the two paradigms, in terms of influencing design they are quite different, requiring separate examination and evaluation.

Standard Compression Goals

Saving disk and network space by compression is undeniably a worthwhile endeavor. This goal was instated long before the Human Genome Project was completed, and has continued to be a popular research topic for statisticians and biologists alike. While Lempel-Ziv provided a huge contribution that is still in use [10], advancements have been made since that have further simplified the task of storing and transferring genetic data. There is also evidence to suggest that while the major leaps in compression have already been made, there is room for improvement, even in the most sophisticated of algorithms.

coil, 2007

There are two distinct ways of looking at DNA compression; the first of these is to attempt to compress individual biological sequences. This is what previously mentioned algorithms in this paper, GenCompress [3] and BioCompress [5], aimed to accomplish. However once the HGP was completed, advents in sequencing made the need for full genome databases valid. At this point some researchers shifted their focus to compressing entire databases worth of genome data. A paper published by Timothy White and Michael Hendy in *BMC Bioinformatics* presents a software tool, “coil”, which exploits the idea of edit-tree coding to compress an entire database worth of data to a transmittable size [9].

Method

coil revolves around a scheme that creates a series of trees of similar sequences, encoded in a fashion that saves as much space as possible. This is achieved through the following four-step process:

1. Using a similarity measure related to Levenshtein distance computation, coil counts the number of length-k substrings that each sequence has in common with all the other sequences and groups sequences of high similarity. These groups are used to create a representative map with nodes symbolizing sequences and edges between nodes weighted differently according to strength of similarity.
2. coil then constructs an encoding graph from the similarity graph computed in step one. This graph is made up of rooted trees with directed arcs representing similarity.
3. Each tree must be encoded as well. The root of each tree is stored as the full sequence written out, and then each other sequence down the tree is “delta-encoded” from its parents. That is, only differences from the parent sequence are recorded as opposed to the entire sequence.
4. gzip or bzip is run on the entire graph to provide further compression [9].

Analysis

coil was tested against four other compression programs: bz2, nrdb+bz2, PPMdi, and 7z. These are all variants of general compression algorithms (meaning they are not specific to DNA, with the exception of nrdb+bz2, which eliminates duplicate sequences). coil and 7z significantly outperformed all the other algorithms, with 7z edging coil on small datasets and coil proving optimal for large datasets. [9]

coil adopts a paradigm of "one off investment" which means at least one sequence in the original database must be stored in entirety to transmit the entire data set. The larger the dataset, the better the investment as the compression of more sequences far outweighs the cost of transmitting one or several in full. This also means that for smaller databases there are other methods that will, and did in the study's published results, perform better than coil. Another limitation arises from the strategy employed in making the encoding trees. Because coil operates on a delta scheme for the encoding, this scheme works best on databases with highly similar, shorter sequences. In a potential situation where a database has been constructed to look for overarching genomic themes across species (such as one mentioned later), this method would theoretically do poorly and have problems other than space after compression. This is because the program attempts to pseudo-align all the different genomes (using k-tuple indexing) to construct a similarity graph. Using variable-length, dissimilar fragments could take either quadratic time or GB of RAM, both of which are impractical on a normal machine.

Modified reference genome, 2008

There are a variety of different compression methods that have been researched and developed to produce results optimized for a specific need. Already mentioned are the attempt to compress a single genome (GenCompress , BioCompress) and the attempt to compress a very large database of genomes (coil). An algorithm developed by Christley, et al shows research that was tailored to meet the need of transmitting a smaller number of genomes in a way that is extremely quick and easy. This algorithm assumes that SNPs and indels have been provided for the genomes being analyzed, and seeks to compress that variation data.

Method

The algorithm designed in this paper makes small improvements upon the concept of using a common reference genome and storing additional genomes as variations from this one common reference [1]. There are four important novel details employed that enhance compression impressively beyond using a reference alone.

The first of these details is the use of variable-sized integers to store the position of SNP variations along the chromosome. SNPs that occur in very early positions in the chromosome do not require the full 4 bytes that are allotted to a standard integer data type, and can therefore be represented by a one, two or three-byte integer.

The second detail states that positions can be stored relative to the last variation, known as “DELTA” positions. Because variations tend to happen more frequently in close groups, using DELTA positioning reduces the size needed to represent position further.

The third technique employed uses a reference SNP map to decide whether a SNP is one of the common bi-alleles. In this case, a bitmap is used to indicate if a SNP is common or not. Depending on this answer, a variation can be stored using even less space than the normal two bits required to represent a nucleotide.

Finally, K-mer partitioning is used to encode repeat sequences. Huffman encoding is used to represent common substrings of a computed "size k" in the sequence in the most efficient form possible [9].

Analysis

The modifications made to the well-known idea of using a reference genome and storing the differences may seem small, but they are extremely well thought-out and have a large impact on the factor of reduction. Compared to using gzip on the original variation map, this method provides a 4.5x better compression, resulting in a data size of 4.1MB – small enough, as they make their title, to send as an email attachment.

Human genomes as email attachments

Table 1.

Data sizes for compression techniques

Compression	**SNPs**	**Deletions**	**Insertions**	**Total**
	(KB)	**(KB)**	**(KB)**	**(KB)**
Entire genome				3 169 831
Map to ref genome	68 519	1741	14 274	84 534
Map to ref genome + gzip	14 803	588	2687	18 078
VINT	13 733	765	803	15 301
VINT/DELTA	6475	507	712	7694
VINT/DELTA/DBSNP	3292	507	712	4511
VINT/DELTA/DBSNP/KMER	3292	507	302	4101

[9]

Using variable-sized integers (data type int) instead of regular integers is a bit of a no-brainer from a technological standpoint, but saving the positions in reference to each other is smart, considering the relative distances will always be smaller than the distance from the beginning of each chromosome. The only potential difficulty with this method is that the entire chromosome must be decoded to look for a SNP variation; storing the absolute positions enables an easier lookup.

It's also clever to use a bitmap with a SNP variation map. Most SNPs are bi-allele, which means they can easily be represented as one of two common values using a single bit instead of the usual two bits used to represent a nucleotide. A potential improvement would be to maintain several different haplotypes instead of a single SNP map. Using simple machine learning techniques, the program, based on previous "tag" SNPs it had encountered, could accurately predict and store later SNPs with a diminutive margin of error. This could be done without having to examine and potentially store an "uncommon" version of every single SNP [8].

Many of the advancements moving forward in compression are going to be small and detail-oriented, and must make carefully calculated decisions about storage, as this study did.

Biological Inference

Though standard compression is important, several studies have attempted to reach a slightly different goal: storing data in a manner that is not only efficient, but is encoded in a manner that is a particularly useful form for biological applications. This can include phylogenetic sequence analysis, repeat element detection, structural variation analysis, and biological complexity evaluation.

Life domain map, 2008

Menconi, et al conducted a different kind study with the goal of using compressibility of fragments as a measure to make inferences about the biological complexity of different organisms, and even distinguish between different domains. They've used the information content of a sequence, as defined by a lossless data compression algorithm [7], to define complexity per nucleotide of that sequence and use the relative complexity of sequences to compare whole genomes.

Method

The methodology of this study focuses more on the interpretation of results than the actual algorithm used. The research team used a relatively simple compression algorithm based on LZ77 and LZ78 called CASToRe [7]. This method calculated the compressibility (and therefore complexity by the above logic) of genomes across different biological domains. This allowed the team to keep track of differences in fragments (exons, introns, and intergenic regions), while maintaining averages of the three fragment groups within each genome.

Analysis

Using two metrics known as the "curtosis coefficient" and "skewness coefficient", researchers manipulated the complexity data to show that there are indeed differences in the compressibility of genomes from Eukaryotes, Bacteria, and Archaea, which can be identified without ambiguity [7]. While this is important for classifying different genomes on a life map, it is important to note that this method is not yet able to distinguish evolutionary paths.

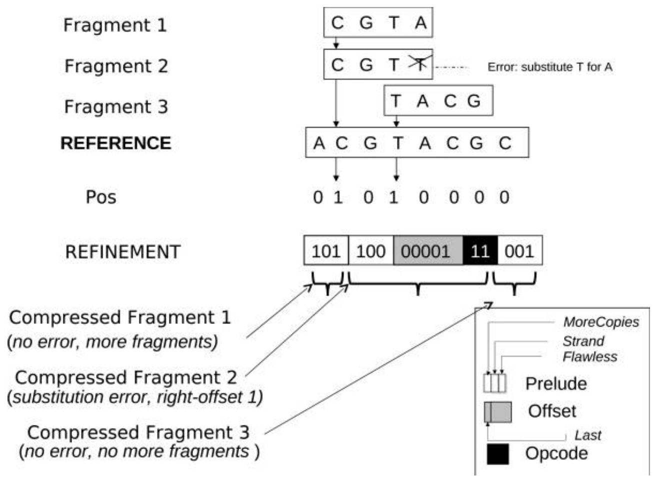
However, it seems that by using the features specific to genomes of distinct domains collected from this study, it could be possible to create an “intelligently randomized” algorithm that could determine a likely evolutionary link between the genomes. Finding overarching differences in complexity is an important first step. The next should be attempting to generate these differences in complexity in an experimental setting that can be verified by repeated trials. Using hypothetical knowledge of how selection forces acted differently on eukaryotes and prokaryotes, as briefly mentioned in the conclusion of Menconi, et al could help limit the sample space of possible evolutionary paths.

SlimGene, 2011

While most of the methods presented thus far have focused on sequence-level compression, Kozanitis, et al chose to focus on fragment-level compression. Kozanitis, et al states that current technologies output sequences as a series of fragments, and there are several widely conducted applications that do not require a costly intermediate step of constructing a complete sequence from these fragments (identifying SNPs, structural variation, etc). This algorithm provides a shortcut for research and should be considered a method tailored specifically to these applications. The program employs a complicated series of bit vectors to align each fragment to a reference and record errors in the alignment.

Method

SlimGene uses two bit vectors (arrays mapping to one bit per fragment) to compare fragments against a reference genome. The first of these is called the "position vector". This vector has one bit for every possible position in the genome, and each bit is set to 1 if some fragment maps to it, and 0 otherwise. The "refinement vector" accounts for the fact that the position vector is not perfect; it records divergences from the reference as a series of bits broken into two parts: the "Prelude", which is always 3 bits, and the "ErrorInstruction" record, which is a variable number of bits. In short terms, the Prelude uses its three bits to indicate how many copies of that fragment there are, whether it aligns with a forward or backward strand in the reference genome, and whether or not it aligns flawlessly. If it does not align perfectly, the ErrorInstruction, through a set of codes, designates the change [6].



[6]

Analysis

SlimGene manages to compress a fairly large 124.7GB file down to 3.2GB, a 40x compression [6]. Though the percent compression is not quite as impressive as the above "email attachment" compression, it must be noted that this algorithm is tailored to techniques that do not require a complete genome to be constructed. Skipping this step entirely will be of value to researchers who only need a way to analyze characteristics of fragments.

The obvious optimization bottleneck on this algorithm's compression is the refinement vector. The error encoding is clever enough that changing the representation would make minimal difference. However, minimizing the number of errors overall could make an impact. Maintaining several genomes with SNP variations is one way to effectively reduce the errors. These could be stored in an edit tree as variants of the first at the cost of a small space addition. An additional bit could be maintained for each fragment denoting which genome the fragment aligned with. The additional space for the extra genomes would be outweighed by the errors that could then be aligned. Another idea could be based off of the idea that since alleles in different haplotypes tend to vary together, a set of haplotypes (again, maintained as an edit tree) could be included with the data. For any errors, a bit could be maintained denoting if a certain SNP was consistent with the haplotype of the previous error. If so, no additional information would be needed. Implemented correctly, these features could save on the storage of many errors.

Expert, 2011

Expert is a probabilistic compression model that was developed by Cao, et al, in an effort to achieve superior performance without the use of a reference genome. The method of calculation is extremely convenient for repeat element detection, pattern recognition, local alignment and phylogenetic analysis.

Method

Expert maintains a set of "experts" that, through machine learning, determine a probability distribution for each symbol and encode that distribution in a "code word". There are different types of experts, such as a Markov expert that uses a Markov model based on previous patterns of symbols and "repeat experts" that predicts based on specific repeated sequences. Experts are weighted differently based on accuracy, and their accuracy is reviewed each iteration [2].

Analysis

Expert effectively compressed each of the genomes to well under 2 bits per symbol, as few as 1.3 for several of them [2]. This model is competitive with the other compression algorithms in use, and lends itself well to information extraction. Because the repeat experts are developed as the algorithm runs, a genome with a repeat expert that outperformed the Markov expert on part of the sequence points strongly to a repeat element. Sequence alignment can be performed using the Markov expert from one sequence, and the repeat expert from a separate sequence, and seeing if adding the repeat expert can increase compression. Finally, a competitively accurate phylogenetic tree can be produced by training a set of experts on one genome, and then compressing a different genome using the same experts to obtain their "mutual information" [2].

These applications are a striking example of the effectiveness of using statistics to analyze biological properties. It is slower than some of the other discussed strategies, so if compression is the only goal then it may be more prudent to use a different method.

However the fact that it does not need to refer to a reference genome makes it an optimal choice for a single, longer genome if time is not an issue.

Undoubtedly there is a lot of potential for further research in the interdisciplinary field of machine learning and biology that this method employs. This same method is likely to apply well to protein compression, and also would probably excel at predicting the function of a sequence. This could be done by examining what training set, with unknown function, allowed the best compression of a sequence whose function is known.

Summary

The field of DNA data compression has been vastly improved since Lempel and Ziv made their first contributions to the field of encoding. Several different novel techniques have been proposed, some of which are simply geared toward effective storage, while others are more application-driven to present information in a useful manner.

Methods that refer to a genome and record the differences seem to be the most effective in saving space. The best of these methods appears to be the optimized reference genome method presented by Christley et al. There is a bound on compression of sequences, and it appears that we are getting quite close to as good as we can do with the technology that exists today. Until the next breakthrough in technology or methods becomes apparent and available, small modifications are going to be where gains in compression are made. Specific projects, such as transferring a whole database or skipping the intermediate step of creating a full sequence can lead to specifically optimized methods as well, like the coil or SlimGene programs.

Statistical methods using machine learning, like Expert, provide exciting new opportunities for information extraction, and may replace existing programs for various applications. Though often sub-optimal in time, they are unprecedented in terms of performing tasks with minimal outside knowledge.

References

[1] M.C. Brandon, D.C. Wallace, P. Baldi. "Data structures and compression algorithms for genomic sequence data". *Bioinformatics*. (2009) 25:1731-1738.

[2] M.D. Cao, T. Dix, L. Allison. "A Biological Compression Model and Its Applications". *Advances in Experimental Medicine and Biology*. (2011) 696: 657-666.

[3] X. Chen, S. Kwong, M. Li. "A compression algorithm for DNA sequences and its applications in genome comparison". *Genome Informatics*. (1999) 10:52-61.

[4] S. Christley, Y. Lu, C. Li, et al. "Human genomes as email attachments". *Bioinformatics*. (2009) 25:274-275.

[5] S. Grumbach, F. Tahi. "A new challenge for compression algorithms: genetic sequences". *Information Processing and Management*. (1994) 30:875-886.

[6] C. Kozanitis, C. Saunders, S. Kruglyak, V. Bafna, G. Varghese. "Compressing Genomic Sequence Fragments Using SlimGene". *Journal of Computation Biology*. (2011) 18:401-413.

[7] G. Menconi, V. Benci, M. Buiatti. "Data compression and genomes: A two-dimensional life domain map". *Journal of Theoretical Biology*. (2008) 253:281-288.

[8] "What Is the HapMap?". *International HapMap Project*. HapMap Data Coordination Center. Web. 10 Mar 2012. < http://hapmap.ncbi.nlm.nih.gov/whatishapmap.html.en>

[9] W. White, M. Hendy. "Compressing DNA sequence databases with coil". *BMC Bioinformatics*. (2008) 9:242

[10] J. Ziv, A. Lempel. "Compression of individual sequences via variable-rate encoding". *Information Theory, IEEE Transactions*. (1978) 24:530-536.