Published online 27 July 2012 *Nucleic Acids Research, 2012, Vol. 40, No. 20* ***e159***
doi:10.1093/nar/gks709

HapZipper: sharing HapMap populations just got easier

Pritam Chanda[1,2], Eran Elhaik[3,4] and Joel S. Bader[1,2,*]

[1]Department of Biomedical Engineering, Johns Hopkins University, [2]High Throughput Biology Center, Johns Hopkins University School of Medicine, [3]McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University School of Medicine and [4]Department of Mental Health, Johns Hopkins University Bloomberg School of Public Health, Baltimore, MD 21205, USA

Received January 2, 2011; Revised June 28, 2012; Accepted July 1, 2012

ABSTRACT

The rapidly growing amount of genomic sequence data being generated and made publicly available necessitate the development of new data storage and archiving methods. The vast amount of data being shared and manipulated also create new challenges for network resources. Thus, developing advanced data compression techniques is becoming an integral part of data production and analysis. The HapMap project is one of the largest public resources of human single-nucleotide polymorphisms (SNPs), characterizing over 3 million SNPs genotyped in over 1000 individuals. The standard format and biological properties of HapMap data suggest that a dedicated genetic compression method can outperform generic compression tools. We propose a compression methodology for genetic data by introducing HapZipper, a lossless compression tool tailored to compress HapMap data beyond benchmarks defined by generic tools such as gzip, bzip2 and lzma. We demonstrate the usefulness of HapZipper by compressing HapMap 3 populations to <5% of their original sizes. HapZipper is freely downloadable from https://bitbucket.org/pchanda/hapzipper/downloads/HapZipper.tar.bz2.

INTRODUCTION

Continuous improvements of high-throughput sequencing technologies yield genomic datasets that are accumulating at an exponentially increasing rate, including both complete genomes and population genotypes.

Next-generation sequencing technologies ushered in a new era in which the cost of sequencing a complete human genome became affordable and is poised to go <$1000 within a few years (1). Population genomic sequences are frequently published (2–5), and a project to sequence over 1000 human genomes is currently underway (6).

Publicly available genomic datasets are typically stored as flat text files with increasing burdens for electronic storage and transmission (7). The storage of polymorphic markers for all currently living humans is estimated to be around million terabytes. Projected storage requirements must also consider additional plant and animal populations. Storing, sharing or downloading genetic information remotely is laborious and nearly impossible for institutions in parts of the world lacking high-speed Internet access. Even today, large storage sites such as the Broad Institute and the European Bioinformatics Institute, spend millions of dollars on storage (7,8) and massive data transfer remains an imposing burden on servers and Internet networks (9).

For researchers facing the daunting challenge of interpreting vast genomic datasets, data storage is a central issue. Collaborations necessitate handling a huge influx of datasets and require strategies for data transfer and storage. Often data can only be shared by sending physical storage devices to distant collaborators, which is a slow and expensive process. Unfortunately, general-purpose compression tools such as gzip, bzip2 and lzma offer limited compression abilities.

Improved compression can be achieved only with dedicated tools that consider the special properties of genetic data. This insight motivated the development of compression tools specialized for mitochondrial and nuclear genome sequence (10,11).

Our new proposed bit-level compression approach is suitable for polymorphic genetic data. Given the importance of HapMap data (12), we used phased haplotypes to demonstrate the usefulness of our approach. We emphasize that the framework provided by HapZipper can

*To whom correspondence should be addressed. Tel: +1 410 516 7417; Fax: +1 410 516 6240; Email: joel.bader@jhu.edu

The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.

© The Author(s) 2012. Published by Oxford University Press.
This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0), which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

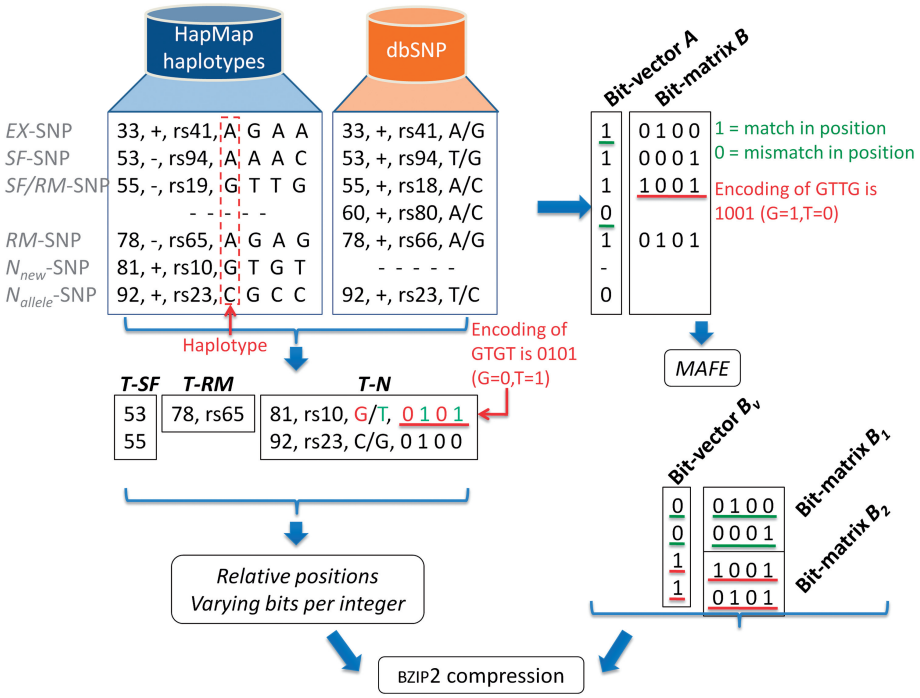

Figure 1. HAPZIPPER compression diagram provided a dbSNP database. Phased haplotypes are compared with dbSNP SNPs and classified into four categories: *EX*-SNPs, *SF*-SNPs, *RM*-SNPs and *N*-SNPs that include novel SNPs (N_{new}-SNP) and SNPs that are similar to dbSNP except the alleles (N_{allele}-SNP). Using dbSNP, the haplotype data are encoded into a bit-vector *A*, which corresponds dbSNP SNPs, and a bit-matrix *B*, which encodes all the alleles marked as 1 in bit-vector *A*. The bit-matrix *B* is then divided into two bit-matrices B_1 and B_2, based on the MAF of the SNPs. The binary bit-vector B_v represents the bit-matrices order in the original matrix. The information for categorical SNPs is stored in three tables: T-*SF*, T-*RM* and T-*N* that are further compressed using relative positions and varying bits per integer. In the absence of a dbSNP database only the table T-*N* is used.

be easily modified to compress other genetic datasets. We further note that the proposed method is not a compression algorithm in the conventional computer science meaning, but it rather uses a specific encoding scheme and applies existing and novel compression algorithms to efficiently and effectively compress the HapMap data.

HAPZIPPER is a lossless compression tool tailored for HapMap data and achieves over 20-fold compression (95% reduction in file size), providing 2- to 4-fold better compression than the leading general-purpose compression utilities. We apply HAPZIPPER to HapMap Phase III (release 2)-phased haplotypes of entire populations and compress them into several megabases, small enough for easy sharing.

MATERIALS AND METHODS

Reference-based compression scheme

High compression of genetic variation data for entire populations is accomplished by mapping the data to an existing reference map (dbSNP), National Center for Biotechnology Information's free public database of single-nucleotide polymorphism (SNP) available at http://www.ncbi.nlm.nih.gov/projects/SNP/. This resource is intended to contain all known genomic variations in human and other species. Variation data comprise SNPs and insertions/deletions of multiple nucleotides. Upon identification, SNPs are commonly submitted to a public database such as dbSNP. These SNPs are later used to generate haplotype maps (HapMap). Our method assumes that both HapMap (12) and dbSNP will remain publicly available.

The compression scheme of HAPZIPPER consists of three stages. First, HapMap haplotype data are encoded into a bit-sequence using dbSNP as a reference. Second, taking advantage of the genetic redundancy in population data, the encoded dataset is split into two datasets of high- and low-minor allele frequencies (MAFs). Finally, the two parts are compressed separately using bzip2 (Figure 1). We first describe the encoding scheme that generates the bit-sequence.

HapMap encoding

HapMap haplotypes (12) consist of biallelic SNPs on a single chromatid that are also recorded in the dbSNP database. Thus, each HapMap haplotype can be encoded as a bit-sequence in reference to dbSNP. More specifically, for each chromosome, each HapMap SNP is classified according to the following four categories (Figure 1):

(1) Exact match SNPs (or *EX*-SNPs): A SNP that matches a dbSNP SNP in base pair position, alleles and SNP reference number (i.e. SNP name).

(2) Strand flipped SNPs (or *SF*-SNPs): A SNP that matches a dbSNP SNP in base pair position, SNP reference number and alleles from the opposite strand.
(3) Reference number mismatch SNPs (or *RM*-SNPs): A SNP that matches a dbSNP SNP in base pair position and alleles, but not in SNP reference number.
(4) Novel SNPs (or N-SNPs) consist of two types:
   - N_{allele}-SNP: a SNP that matches a dbSNP SNP in base pair position, but does not match the alleles of the corresponding dbSNP.
   - N_{new}-SNP: a SNP that does not exist in dbSNP in the same base pair position.

SNPs that have both a mismatch in the reference SNP number compared with the dbSNP and that have matching alleles with dbSNP only after a strand inversion belong to both the *SF*-SNP and *RM*-SNP categories (*SF*/*RM*-SNPs). All other SNPs belong to a single category.

In our compression scheme, for each HapMap chromosome, a bit-vector A is created in reference to dbSNP with '1' indicating an *EX*-SNP, *SF*-SNP or *RM*-SNP and '0' indicating a N_{allele}-SNP. Novel HapMap SNPs (N_{new}-SNPs) will not have any entry in the correspondence bit-vector A. Finally, a dbSNP SNP that does not have a corresponding HapMap SNP is also indicated with 0 in the bit-vector A (see rs80 in Figure 1).

The bit-matrix B encodes the genotypes in reference to dbSNP alleles. For each HapMap individual, the two alleles of the *EX*-, *SF*- and *RM*-SNPs are stored as a bit-sequence of 1 for the dbSNP common variant and 0 if otherwise (Figure 1). Assume that there are M dbSNP SNPs out of which K SNPs belong to categories 1, 2 and 3. Let each HapMap SNP consists of $2N$ nucleotides (for N individuals) for each of the K SNPs occupying $2NK$ bits. Therefore, bit-vector A and bit-matrix B would consist of M bits (in correspondence with dbSNP SNPs) plus $2NK$ bits from the SNP nucleotides so that the overall bit-sequence size is $M+2NK$ bits. The information for the categorical SNPs is stored by three tables (Figure 1):

(1) Table *T-SF* stores the base pair position with the mismatch strands for each *SF*-SNP.
(2) Table *T-RM* stores the base pair position and reference SNP number for each *RM*-SNP.
(3) Table *T-N* stores for N_{new}- and N_{allele}-SNPs the base pair position, reference SNP number and the two alleles of each HapMap individual. Each haplotype is encoded as a bit-sequence of 1's and 0's based on the two HapMap alleles. Clearly, the *T-N* table stores only novel HapMap SNPs that are absent from dbSNP. When dbSNP is not used, this table stores all the SNPs. We term this mode HAPZIPPERMINUS.

Thus for each chromosome, the bit-vector A, bit-matrix B and the three tables encode the complete information for each HapMap haplotype. We next describe HAPZIPPER compression techniques.

Compression of encoded HapMap

The human genome is non-random and highly repetitive. Many HapMap alleles are rare, indicating a large abundance of monomorphic SNPs in most populations. Note that either the reference or the alternative alleles can be rare. Using these genomic properties, we considered the following compression techniques to reduce the size of bit-matrix B and of the associated tables *T-SF*, *T-RM* and *T-N*.

Relative positions with varying bits per integer

The positions stored in the tables *T-SF*, *T-RM* and *T-N* are strictly increasing because the SNPs are ordered according to their base pair positions in both HapMap and dbSNP. As chromosome sizes can be as large as 247 Mb, storing the full position is expensive. Therefore, instead of storing absolute positions in a table, each position is stored as the difference with respect to the absolute position of the previous entry of the table. For example, SNP positions 103456, 144588, 183256, ... will be stored as 103456, 41132, 3868, ..., using the offset from the previous position. Moreover, the 'relative positions' are stored as integers with varying numbers of bits; a single bit at the most significant position in a byte when set to 1 is used to indicate the most significant byte of an integer which also indicates the boundary between two integers. Thus, we do not require a fixed number of bytes to store each relative position. Instead, several integer values can be compactly concatenated together to save space. As a result, relative position values ranging from 0 to 127 are stored using 1 byte of storage, 128–16 383 consumes 2 bytes and so on. Figure 2 gives an example of the encodeing scheme of relative positions for values 127, 128, 16 383 and 16 384.

MAF based encoding (MAFE)

The bit-matrix B, encoding the genotypes, is a highly complex matrix exhibiting a mosaic of sparsity (0's) and density (1's) due to the high frequency of rare alleles and their random positioning. This complexity complicates the compression process. To reduce the heterogeneity of this matrix, we calculate the mean MAF for each SNP and divide the bit-matrix B into two bit-matrices B_1 and B_2 of low- and high-MAF SNPs, respectively. A binary bit-vector B_v of size K identifies their original positions (Figure 1). As most of the SNPs are rare, the coded alleles will have either high or low frequencies and thus the compression of the two bit-matrices (each consists mostly of 0's or 1's) is more efficient than the compression of bit-matrix B. Finally, we use BZIP2 to compress the four files: bit-vector A, bit-matrix B_1, bit-matrix B_2 and bit-vector B_v. A similar procedure is followed for encoding the *T-N* table, which requires a similar bit-vector A and bit-matrix B for its encoding.

Compressing HapMap III population data

HAPZIPPER effectiveness was demonstrated on phased data of 18 HapMap (Phase III draft 2) populations downloaded from http://hapmap.ncbi.nlm.nih.gov/downloads/phasing/2009-02_phaseIII/HapMap3_r2/. The reference

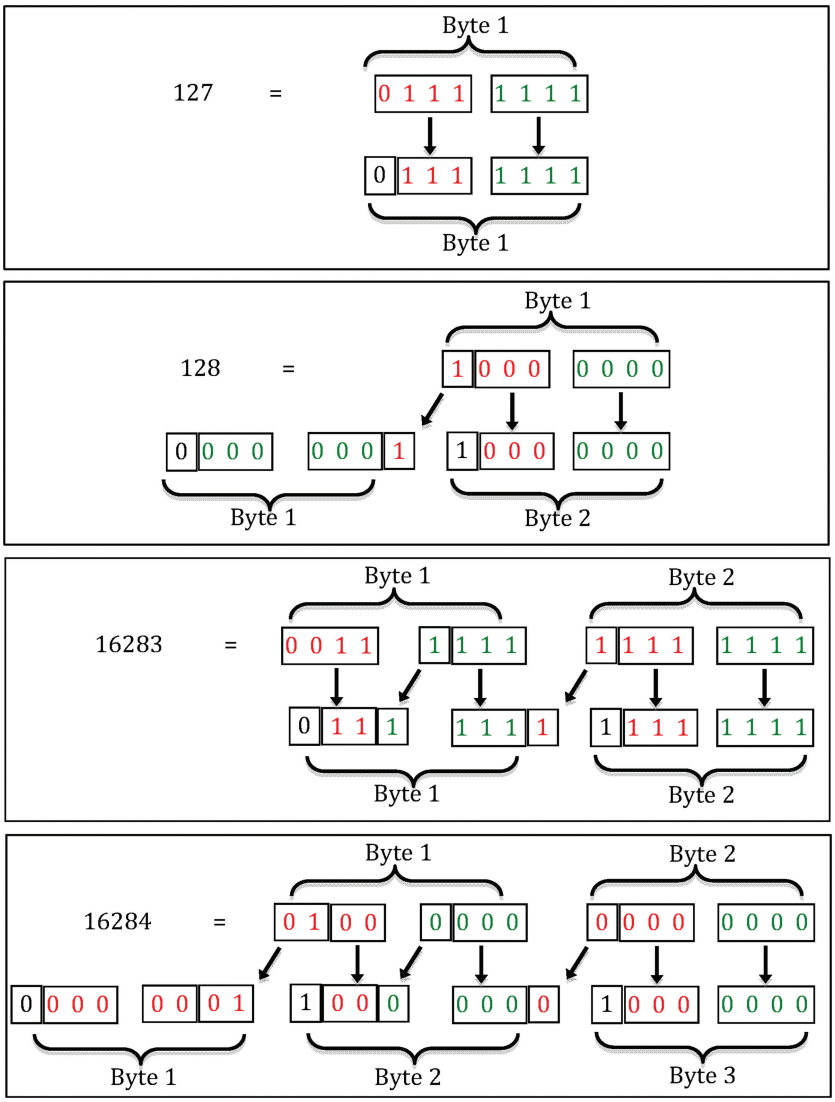

Figure 2. Storing relative positions using varying number of bits. The binary representation of each relative position (top row) and the corresponding encoded binary value (bottom row) are shown for boundary values 127, 128, 16 383 and 16 384. A bit in black shows the flag bit within an encoded byte that indicates beginning of a relative position value with a 0.

was dbSNP (build 129). The populations consists of unrelated individuals, sibling duos and parent–child trios.

Performance is assessed both with dbSNP (HAPZIPPER) and without dbSNP (HAPZIPPERMINUS.) as a reference and is compared with GZIP, BZIP2 and LZMA. GZIP is based on Lempel–Ziv encoding (LZ77) (13). BZIP2 is based on the Burrows–Wheeler transform (14) and is considered more effective than GZIP, but slower. LZMA (Lempel–Ziv–Markov chain algorithm) uses a dictionary compression scheme similar to LZ77 and is considered more effective and faster than BZIP2. We tested different options for compression flags and found that the default flags for GZIP, BZIP2 and LZMA all provided optimal compression time and file size (Supplementary Table S4).

The compression effectiveness is compared using 'fold-compression', defined as the ratio between the file-size before compression and the file-size after compression (e.g. compressing 100 Mb file to 10 Mb would be 10×, that is 10-fold compression). CPU timings are reported for dual-core AMD opteron processors with 3 GHz CPU and 8 GB memory and may vary for different hardware, but the fold-compression is expected to be similar.

Our tool is implemented in C++ and is available under a BSD open source license. Updated versions are available at https://bitbucket.org/pchanda/hapzipper/downloads/HapZipper.tar.bz2 under GNU (version 3 or later) general public license with sample data for testing. Further details about the compressed file structure can be found in our user guide available with the tool. We also provide example files to demonstrate how to use the tool.

RESULTS

We applied HAPZIPPER and HAPZIPPERMINUS to the HapMap-phased haplotypes in 18 populations, containing

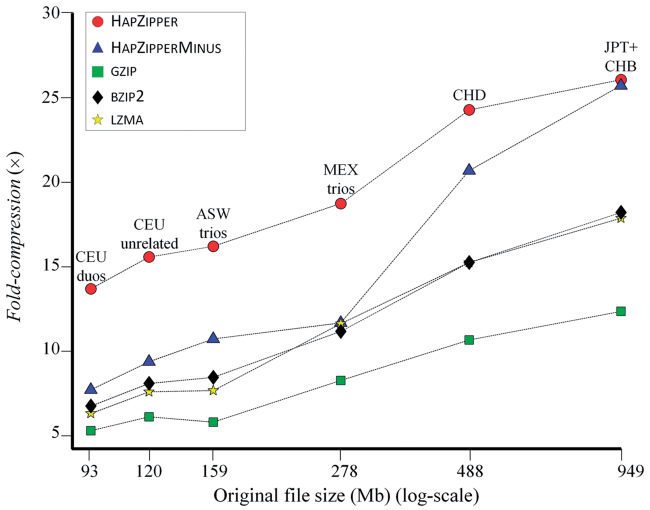

Figure 3. A comparison of the compression tool fold-compressions on six HapMap population datasets of various sizes.

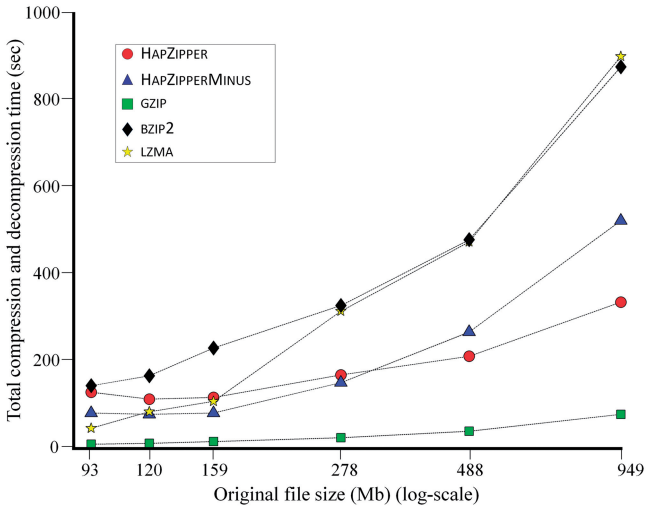

Figure 4. A comparison of the compression tool total run-time (compression + decompression) using six HapMap population datasets of various sizes.

1 387 466 SNPs across 22 autosomal chromosomes. The smallest populations consisted of 12 haplotypes (YRI-duos and MEX-duos) and the largest population consisted of 340 haplotypes (JPT + CHB). Population datasets ranged from 60 Mb to ~1 Gb in size (Supplementary Table S1). We compared the performance of HAPZIPPER with GZIP, BZIP2 and LZMA using the fold-compression defined as the ratio (file size before compression)/(file size after compression).

Mapping the HapMap SNPs to dbSNP, we found ~219 000 (15.8%) strand flipped SNPs (*SF*-SNPs), ~2500 (0.2%) reference number mismatch SNPs (*RM*-SNPs) and ~2300 (0.2%) novel SNPs (*N*-SNPs) (Supplementary Table S2).

Using GZIP, fold-compression ranged from 4× (YRI-duos and MEX-duos) to 12× (JPT + CHB), with the compression effectiveness improving for larger datasets (Figure 3). BZIP2 and LZMA performed similarly to one another with a relatively low fold-compression of 5× for small populations that improves for larger families until reaching 18×. By comparison, HAPZIPPER reduced the entire dataset size by 95% with a fold-compression of 13–26×. Using GZIP to compress the files already compressed with HAPZIPPER resulted in a difference of ~3.5% in the size of the compressed files, demonstrating the higher compression achieved by HAPZIPPER (Figure 3).

We also measured the performances of HAPZIPPER when dbSNP was not used as a reference during compression (HAPZIPPERMINUS) (Supplementary Table S1). In such cases, all SNPs in the population are considered novel SNPs and stored in the *T*-*N* table. HAPZIPPERMINUS outperformed all other algorithms reducing the entire dataset size by 94% with a fold-compression of 6–26× (Figure 3). Remarkably, HAPZIPPERMINUS performances approached those of HAPZIPPER for large data files and matched them on the largest dataset of 949 Mb (Supplementary Table S1).

There are several reasons to how HAPZIPPER achieves such high compression compared with the generic compression methods. First, the inability of GZIP to encode long SNP positions with 'relative positions' and 'varying bits per integer' degrades its performance. By comparison, both HAPZIPPER and HAPZIPPERMINUS encode each haplotype as a bit-sequence, whereas GZIP does not attempt to encode the haplotypes. The savings from the different compression techniques (relative positions and varying bits per integer) successfully overcome the complex compression schemes of BZIP2 and the compression that GZIP and LZMA achieve using *LZ77*. These key enhancements, however, affect mostly HAPZIPPER, but not HAPZIPPERMINUS that, like the generic algorithms, does not take advantage of dbSNP data. For that we implemented a novel compression scheme using our knowledge on the SNP MAFs. The MAF encoding was designed to increase the homogeneity of the sequence data and enhances its compression. This step boosts the compression of HAPZIPPERMINUS by 50% and is responsible for its high performances.

Overall, HAPZIPPER compression is up to four times better than the GZIP compression and up to three times better than BZIP2 and LZMA compressions. HAPZIPPERMINUS compression is nearly two times better than generic compression tools. Applying HAPZIPPER to the HapMap population dataset, originally ranging from 60 to 949 Mb, results in 95% reduction in file size for population data ranging from 4.5 (YRI-duos) to 36.2 Mb (JPT + CHB).

The total run-time (compression + decompression) varies between all the tools (Figure 4 and Supplementary Table S3a and b). The fastest compression is with GZIP, followed by HAPZIPPER and HAPZIPPERMINUS, which are 8 and 7 times slower, and BZIP2 and LZMA which are 9 and 14 times slower than GZIP, respectively. The HAPZIPPER implementation was not optimized for CPU performance, and performance gains should be possible.

The default settings of GZIP, BZIP2 and LZMA are set to provide the most optimal compression in terms of both

time and file size (Supplementary Table S4). Even when set to optimal compression, at the cost of increasing computation time, HAPZIPPER and HAPZIPPERMINUS outperformed all other algorithms, with the exception of GZIP that performed better than HAPZIPPERMINUS for small populations (<100 phased haplotypes) (Supplementary Tables S1 and S4). We therefore concluded that HAPZIPPER outperforms other methods in practical settings, at their respective maximum compression setting.

DISCUSSION

To demonstrate the power of dedicated genetic compression approaches, we developed a compression scheme that reduces the size of the HapMap dataset by >20-fold (over 95% reduction in file size). Our compression scheme is scalable with the expected growth of databases; that is the more complete the reference SNP map, the smaller the compressed file will be. There are two reasons for that: first, position data (insertions) are the most expensive to compress. In other words, new variants added to dbSNP from the 1000 Genomes project would reduce the cost of compressing the HapMap data presented here. Second, most of the new SNPs are expected to be rare and create complex allele matrices that are well handled by our algorithm.

As compression tools specific to DNA sequence data (11) lose the phased haplotype information in HapMap, we compared HAPZIPPER with general-purpose lossless compression tools, such as GZIP, BZIP2 and LZMA. HAPZIPPER compresses HapMap data better than any of these tools for all population sizes tested. For example, compressed by GZIP, the HapMap 3 dataset consumes 687 Mb of disk space, whereas HAPZIPPER and HAPZIPPERMINUS require only 302 and 384 Mb, respectively, to store the compressed files. HAPZIPPER high compression effectiveness comes at the cost of slower compression and decompression run-times, relative to GZIP (Figure 4 and Supplementary Tables S3a and b). However, it is possible to improve the HAPZIPPER run-time using low-level disk access to read/write large HapMap files. Moreover, the compression and decompression times are on the order of the time to transmit the file itself electronically.

HAPZIPPER performs more effectively and efficiently if the sender and receiver both have a copy of the reference SNP map (of the same version number), although this is not essential. We have shown that HAPZIPPER and HAPZIPPERMINUS outperform other compression tools even in the absence of any reference dataset (Figure 3). Providing HAPZIPPER with a copy of the reference SNP data (dbSNP) will optimize the compression and require a smaller disk space because only the SNP reference number, alleles and base pair positions are stored. Effectively storing the dbSNP dataset reduces its size to only 165 Mb using variable number of bytes for both the 'relative positions' and SNP reference numbers and by encoding the four possible nucleotides with 2 bits. Overall, HAPZIPPER (302 Mb) and dbSNP (165 Mb) use only 467 Mb of disk space for all populations used in this analysis, less than BZIP2 (496 Mb), LZMA (548 Mb) and GZIP (687 Mb).

HAPZIPPER is easy to use, and it runs in time proportional to the number of individuals to be compressed (Figure 4). The method described here cannot be used for imputed data consisting of numerical values (e.g. dosage data), but it can be easily adapted to compress unphased genotype data from HapMap or other datasets, such as the 1000 Genomes project data. We hope that HAPZIPPER will become a viable tool for data compression.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online: Supplementary Tables 1–4.

ACKNOWLEDGMENTS

We thank Scott Christley for his helpful answers regarding the DNAZip tool. We thank Emily Foster, Vasyl Pihur and Niv Sabath for their helpful comments in this article. We thank two anonymous reviewers for their helpful suggestions.

FUNDING

Funding for open access charge: Robert J. Kleberg Jr and Helen C. Kleberg Foundation [to J.S.B. and P.C.]; National Institutes of Mental Health [T32MH014592 to E.E.].

Conflict of interest statement. None declared.

REFERENCES

1. Service,R.F. (2006) The race for the $1000 genome. *Science*, **311**, 1544–1546.
2. Ahn,S.M., Kim,T.H., Lee,S., Kim,D., Ghang,H., Kim,D.S., Kim,B.C., Kim,S.Y., Kim,W.Y., Kim,C. *et al.* (2009) The first Korean genome sequence and analysis: full genome sequencing for a socio-ethnic group. *Genome Res.*, **19**, 1622–1629.
3. Levy,S., Sutton,G., Ng,P.C., Feuk,L., Halpern,A.L., Walenz,B.P., Axelrod,N., Huang,J., Kirkness,E.F., Denisov,G. *et al.* (2007) The diploid genome sequence of an individual human. *PLoS Biol.*, **5**, e254.
4. Wang,J., Wang,W., Li,R., Li,Y., Tian,G., Goodman,L., Fan,W., Zhang,J., Li,J., Guo,Y. *et al.* (2008) The diploid genome sequence of an Asian individual. *Nature*, **456**, 60–65.
5. Schuster,S.C., Miller,W., Ratan,A., Tomsho,L.P., Giardine,B., Kasson,L.R., Harris,R.S., Petersen,D.C., Zhao,F., Qi,J. *et al.* (2010) Complete Khoisan and Bantu genomes from southern Africa. *Nature*, **463**, 943–947.
6. The 1000 Genomes Project Consortium. (2010) A map of human genome variation from population-scale sequencing. *Nature*, **467**, 1061–1073.
7. Willyard,C. (2010) Tech teams try to curate genetic data for future use. *Nat. Med.*, **16**, 733–733.
8. Dublin,M. (2009) So Long, Data Depression. *Genome Technology*, http://www.genomeweb.com/informatics/so-long-data-depression.
9. Sansom,C. (2010) Up in a cloud? *Nat. Biotechnol.*, **28**, 13–15.
10. Brandon,M.C., Wallace,D.C. and Baldi,P. (2009) Data structures and compression algorithms for genomic sequence data. *Bioinformatics*, **25**, 1731–1738.

11. Christley,S., Lu,Y., Li,C. and Xie,X. (2009) Human genomes as email attachments. *Bioinformatics*, **25**, 274–275.
12. Altshuler,D.M., Gibbs,R.A., Peltonen,L., Dermitzakis,E., Schaffner,S.F., Yu,F., Bonnen,P.E., de Bakker,P.I., Deloukas,P., Gabriel,S.B. *et al.* (2010) Integrating common and rare genetic variation in diverse human populations. *Nature*, **467**, 52–58.
13. Ziv,J. and Lempel,A. (1977) A universal algorithm for sequential data compression. *IEEE Trans. Inform. Theory*, **23**, 337–343.
14. Burrows,W. and Wheeler,D.J. (1994) A block-sorting lossless data compression algorithm. *Technical Report 124*. Digital Equipment Corp., Maynard, MA.